Bull

AIX Commands Reference Vol.5
ruutry to uux

AIX

ORDER REFERENCE
86 A2 42JX 02

Bull

AIX Commands Reference Vol.5
ruutry to uux

AIX

Software

April 2000

BULL ELECTRONICS ANGERS
CEDOC
34 Rue du Nid de Pie – BP 428
49004 ANGERS CEDEX 01
FRANCE

ORDER REFERENCE
86 A2 42JX 02

Printed in France

Suggestions and criticisms concerning the form, content, and presentation of this book are invited. A form is provided at the end of this book for this purpose.

To order additional copies of this book or other Bull Technical Publications, you are invited to use the Ordering Form also provided at the end of this book.

Trademarks and Acknowledgements

We acknowledge the right of proprietors of trademarks mentioned in this book.

AIX® is a registered trademark of International Business Machines Corporation, and is being used under licence.

UNIX is a registered trademark in the United States of America and other countries licensed exclusively through the Open Group.

Year 2000

The product documented in this manual is Year 2000 Ready.

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Commands Reference, Volume 5

First Edition (October 1997)

This edition of the *AIX Version 4.3 Commands Reference, Volume 5* applies to the AIX Version 4.3, 3270 Host Connection Program 2.1 and 1.3.3 for AIX, and Distributed SMIT 2.2 for AIX licensed programs, and to all subsequent releases of these products until otherwise indicated in new releases or technical newsletters.

A reader's comment form is provided at the back of this publication. If the form has been removed, address comments to Publications Department, Internal Zip 9561, 11400 Burnet Road, Austin, Texas 78758−3493. To send comments electronically, use this commercial internet address: `aix6kpub@austin.ibm.com`. Any information that you supply may be used without incurring any obligation to you.

Trademarks and Acknowledgements

The following trademarks and acknowledgements apply to this book:

ADM is a trademark of Lear Siegler, Inc.

AIX is a registered trademark of International Business Machines Corporation.

Connect is a trademark of INTERACTIVE Systems Corporation.

DEC is a trademark of Digital Equipment Corporation.

DEC VT100, VT220, VT320, and VT330 are trademarks of Digital Equipment Corporation.

GL is a trademark of Silicon Graphics, Inc.

HP is a trademark of Hewlett−Packard Company.

IBM is a registered trademark of International Business Machines Corporation.

INed is a trademark of INTERACTIVE Systems Corporation.

InfoExplorer is a trademark of International Business Machines Corporation.

Intel is a trademark of Intel Corporation.

Interleaf is a trademark of Interleaf, Inc.

LaserJet Series II is a trademark of Hewlett−Packard Company.

Micro Channel is a registered trademark of International Business Machines Corporation.

NetView is a trademark of International Business Machines Corporation.

Network Computing System is a trademark of Apollo Computer, Inc.

OSF and OSF/Motif are trademarks of Open Software Foundation, Inc.

Personal Computer AT and AT is a registered trademark of International Business Machines Corporation.

Personal System/2 is a registered trademark of International Business Machines Corporation.

PS/2 is a registered trademark of International Business Machines Corporation.

POSIX is a trademark of the Institute of Electrical and Electronic Engineers (IEEE).

PostScript is a trademark of Adobe Systems Incorporated.

Proprinter is a registered trademark of International Business Machines Corporation.

Quickwriter is a registered trademark of International Business Machines Corporation.

Quiet is a trademark of International Business Machines Corporation.

RS/6000 is a trademark of International Business Machines Corporation.

RT is a registered trademark of International Business Machines Corporation.

Sun is a trademark of Sun Microsystems, Inc.

Tektronix is a trademark of Tektronix, Inc.

Televideo is a trademark of Televideo, Inc.

The Source is a service mark of Source Telecomputing Corp., a subsidiary of The Reader's Digest Assn., Inc.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

WY−50 is a trademark of the WYSE Corporation.

WYSE is a trademark of WYSE Corporation.

About This Book

This book is Volume 5 of the six−volume *AIX Version 4.3 Commands Reference*, SBOF−1877, which contains reference information on Advanced Interactive Executive (AIX) Operating System commands. It describes the tasks each command performs, how commands can be modified, how they handle input and output, who can run them and provides a master index for all six volumes.

For a quick reference list of commands arranged in functional groups, see Volume 6.

Who Should Use This Book

This book is intended for users of AIX commands.

How to Use This Book

A command is a request to perform an operation or run a program. You use commands to tell the AIX Operating System what task you want it to perform. When commands are entered, they are deciphered by a command interpreter (also known as a shell) and that task is processed.

Some commands can be entered simply by typing one word. It is also possible to combine commands so that the output from one command becomes the input for another command. This is known as pipelining.

Flags further define the actions of commands. A flag is a modifier used with the command name on the command line, usually preceded by a dash.

Commands can also be grouped together and stored in a file. These are known as shell procedures or shell scripts. Instead of executing the commands individually, you execute the file that contains the commands.

Some commands can be constructed using Web−based System Manager applications or the System Management Interface Tool (SMIT).

Highlighting

The following highlighting conventions are used in this book:

Bold	Identifies commands, subroutines, keywords, files, structures, directories, and other items whose names are predefined by the system. Also identifies graphical objects such as buttons, labels, and icons that the user selects.
Italics	Identifies parameters whose actual names or values are to be supplied by the user.
`Monospace`	Identifies examples of specific data values, examples of text similar to what you might see displayed, examples of portions of program code similar to what you might write as a programmer, messages from the system, or information you should actually type.

Format

Each command may include any of the following sections:

Purpose	A description of the major function of each command.
Syntax	A syntax diagram showing command line options.
Description	A discussion of the command describing in detail its function and use.

Flags	A list of command line flags and associated variables with an explanation of how the flags modify the action of the command.
Parameters	A list of command line parameters and their descriptions.
Subcommands	A list of subcommands (for interactive commands) that explains their use.
Exit Status	A description of the exit values the command returns.
Security	Specifies any permissions needed to run the command.
Examples	Specific examples of how you can use the command.
Files	A list of files used by the command.
Related Information	A list of related commands in this book and related discussions in other books.

Implementation Specifics

To list the installable software package (fileset) of an individual command use the **lslpp** command with the **−w** flag. For example, to list the fileset that owns the **installp** command, enter:

```
lslpp −w /usr/sbin/installp
```

Output similar to the following displays:

```
File                            Fileset                 Type
-----------------------------------------------------------------------
/usr/sbin/installp                bos.rte.install           File
```

To list the fileset that owns all file names that contain `installp`, enter:

```
lslpp −w "*installp*"
```

Output similar to the following displays:

```
File                            Fileset                 Type
-----------------------------------------------------------------------
/usr/sbin/installp                bos.rte.install           File
/usr/clvm/sbin/linstallpv         prpq.clvm                 File
/usr/lpp/bos.sysmgt/nim/methods/c_installp
                                  bos.sysmgt.nim.client     File
```

Syntax Diagrams

AIX command syntax is represented by syntax diagrams and usage statements.

Syntax diagrams are designed to provide information about how to enter the command on the command line. A syntax diagram can tell you:

- Which flags can be entered on the command line
- Which flags must take a parameter
- Which flags have optional parameters
- Default values of flags and parameters, if any
- Which flags can and cannot be entered together
- Which flags and parameters are optional
- When you can repeat flag and parameter sequences.

AIX commands use the following conventions in their syntax diagrams:

- Diagram items that must be entered literally on the command line are in **bold**. These items include

the command name, flags, and literal characters.
- Diagram items representing variables that must be replaced by a name are in *italics*. These items include parameters that follow flags and parameters that the command reads, such as *Files* and *Directories*.
- Default values that do not have to be entered are in the normal font on a **bold** path.

The Sample Syntax Diagram illustrates the conventions used in syntax diagrams. Each part of the diagram is labeled. An explanation of the labels follows the diagram.

You interpret the example diagram as follows.

0 PATH LINE	The path line begins the syntax diagram.
1 COMMAND NAME	This item in the diagram is the name of the command you want to invoke. It is in bold, which indicates that it must be entered exactly as it appears in the diagram. In the example diagram, the path branches into two paths after the command name. You can follow either the lower path (discussed in item 2) or the upper path (discussed in item 3).
2 SINGLE CHOICE BOX	If you follow the lower path, you encounter a box with the words *one of* over it. You can choose only one item from this box.
3 DEFAULT LINE	If you follow the upper path, you bypass the single choice box, and enter nothing. The bold line around the box is a default line, which means that you do not have to enter anything from that part of the diagram. Exceptions are usually explained under "Description." One important exception, the blank default line around input and output files, is explained in item 10.
4 REPEAT ARROW	When you follow a path that takes you to a box with an arrow around it, you must choose at least one item from the box. Then you can either follow the arrow back around and continue to choose items from the box, or you can continue along the path. When following an arrow that goes around a box (rather than an arrow that includes several branches in the diagram), do not choose the same item more than once.
5 REQUIRED ITEM	Following the branch with the repeat arrow is a branch with three choices and no default line around them. This means that you must choose one of A, B, or C.
6 GO TO NEXT LINE	If a diagram is too long to fit on one line, this character tells you to go to the next line of the diagram to continue entering your command. Remember, the diagram does not end until you reach the vertical mark.
7 CONTINUE DIAGRAM	This character shows you where to continue with the diagram after it breaks on the previous line.
8 OPTIONAL PARAMETER	If a flag can (but does not have to) take a parameter, the path branches after the flag. If you cannot enter a space between the flag and parameter, you are told in a footnote.
9 DEFAULT VALUE	Often, a command has default values or actions that it will follow if you do not enter a specific item. These default values are indicated in normal font in the default line if they are equivalent to something you could enter on the command line (for example, a flag with a value). If the default is not something you can enter on the command line, it is not indicated in the diagram. **Note:** Default values are included in the diagram for your information. It is not necessary to enter them on the command line.
10 INPUT OR OUTPUT	A command that can read either input files or standard input has an empty

	default line above the file parameter. If the command can write its output to either an output file or to standard output, it is also shown with an empty default line above the output file parameter. If a command can read only from standard input, an input file is not shown in the diagram, and standard input is assumed. If a command writes only to standard output, an output file is not shown in the diagram, and standard output is assumed. When you must supply a file name for input or output, the file parameter is included in the diagram without an empty default line above it.
11 FOOTNOTE	If a command has special requirements or restrictions, a footnote calls attention to these differences.
12 VERTICAL MARK	This ends the syntax diagram.

Running Commands in the Background

If you are going to run a command that takes a long time to process, you can specify that the command run in the background. Background processing is a useful way to run programs that process slowly. To run a command in the background, you use the & (ampersand) operator at the end of the command:

```
Command&
```

Once the process is running in the background, you can continue to work and enter other commands on your system.

At times, you might want to run a command at a specified time or on a specific date. Using the **cron** daemon, you can schedule commands to run automatically. Or, using the **at** and **batch** commands, you can run commands at a later time or when the system load level permits.

Entering Commands

When you work with AIX, you typically enter commands following the shell prompt on the command line. The shell prompt can vary. In the following examples, $ is the prompt.

To display a list of the contents of your current directory, you would type **ls** and press the Enter key:

```
$ ls
```

When you enter a command and it is running, the operating system does not display the shell prompt. When the command completes its action, the system displays the prompt again. This indicates that you can enter another command.

The general format for entering AIX commands is:

```
Command Flag(s) Parameter
```

The flag alters the way a command works. Many commands have several flags. For example, if you type the **−l** (long) flag following the **ls** command, the system provides additional information about the contents of the current directory. The following example shows how to use the **−l** flag with the **ls** command:

```
$ ls −l
```

A parameter consists of a string of characters that follows a command or a flag. It specifies data, such as the

name of a file or directory, or values. In the following example, the directory named **/usr/bin** is a parameter:

```
$ ls −l /usr/bin
```

When entering commands in AIX, it is important to remember the following:

- Commands are usually entered in lowercase.
- Flags are usually prefixed with a − (minus sign).
- More than one command can be typed on the command line if the commands are separated by a ; (semicolon).
- Long sequences of commands can be continued on the next line by using the \ (backslash). The backslash is placed at the end of the first line. The following example shows the placement of the backslash:

  ```
  $ cat /usr/ust/mydir/mydata > \
  /usr/usts/yourdir/yourdata
  ```

When certain commands are entered, the shell prompt changes. Because some commands are actually programs (such as the **telnet** command), the prompt changes when you are operating within the command. Any command that you issue within a program is known as a subcommand. When you exit the program, the prompt returns to your shell prompt.

AIX can operate with different shells (for example, Bourne, C, or Korn) and the commands that you enter are interpreted by the shell. Therefore, you must know what shell you are using so that you can enter the commands in the correct format.

Stopping Commands

If you enter a command and then decide to stop that command from running, you can halt the command from processing any further. To stop a command from processing, press the Interrupt key sequence (usually Ctrl−C or Alt−Pause). When the process is stopped, your shell prompt returns and you can then enter another command.

ISO 9000

ISO 9000 registered quality systems were used in the development and manufacturing of this product.

AIX 32−Bit Support for the X/Open UNIX95 Specification

Beginning with AIX Version 4.2, the operating system is designed to support the X/Open UNIX95 Specification for portability of UNIX−based operating systems. Many new interfaces, and some current ones, have been added or enhanced to meet this specification. Beginning with Version 4.2, AIX is even more open and portable for applications.

At the same time, compatibility with previous AIX releases is preserved. This is accomplished by the creation of a new environment variable, which can be used to set the system environment on a per−system, per−user, or per−process basis.

To determine the proper way to develop a UNIX95−portable application, you may need to refer to the X/Open UNIX95 Specification, which can be obtained on a CD−ROM by ordering the printed copy of *AIX Version 4.3 Commands Reference*, order number SBOF−1877, or by ordering *Go Solo: How to Implement and Go Solo with the Single Unix Specification*, order number SR28−5705, a book which includes the X/Open UNIX95 Specification on a CD−ROM.

AIX 32−Bit and 64−Bit Support for the UNIX98 Specification

Beginning with AIX Version 4.3, the operating system is designed to support the X/Open UNIX98 Specification for portability of UNIX−based operating systems. Many new interfaces, and some current ones, have been added or enhanced to meet this specification. Making AIX Version 4.3 even more open and portable for applications.

At the same time, compatibility with previous AIX releases is preserved. This is accomplished by the creation of a new environment variable, which can be used to set the system environment on a per−system, per−user, or per−process basis.

To determine the proper way to develop a UNIX98−portable application, you may need to refer to the X/Open UNIX98 Specification, which can be obtained on a CD−ROM by ordering the printed copy of *AIX Version 4.3 Commands Reference*, order number SBOF−1877, or by ordering *Go Solo: How to Implement and Go Solo with the Single Unix Specification*, order number SR28−5705, a book which includes the X/Open UNIX98 Specification on a CD−ROM.

Related Information

The following books contain information about or related to commands:

- *AIX and Related Products Documentation Overview*, Order Number SC23−2456.
- *AIX Version 4.3 Files Reference*, Order Number SC23−4168.
- *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*, Order Number SC23−4128.
- *AIX Version 4.3 Problem Solving Guide and Reference*, Order Number SC23−4123.
- *AIX Version 4.3 System Management Guide: Communications and Networks*, Order Number SC23−4127.
- *AIX Version 4.3 System Management Guide: Operating System and Devices*, Order Number SC23−4126.
- *AIX Version 4.3 System User's Guide: Operating System and Devices*, Order Number SC23−4121.
- *AIX Version 4.3 System User's Guide: Communications and Networks*, Order Number SC23−4122.
- *AIX Versions 3.2 and 4 Performance Tuning Guide*, Order Number SC23−2365.
- *AIX Version 4.3 Guide to Printers and Printing*, Order Number SC23−4130.
- *AIX Version 4.3 Kernel Extensions and Device Support Programming Concepts*, Order Number SC23−4125.
- *5080 Graphics System Installation, Operation, and Problem Determination*, Order Number GA23−2063.
- *AIX Version 4.3 Technical Reference: Base Operating System and Extensions Volume 1* Order Number SC23−4159
- *AIX Version 4.3 Technical Reference: Base Operating System and Extensions Volume 2*, Order Number SC23−4160.
- *AIX Version 4.3 Technical Reference: Communications Volume 1*, Order Number SC23−4161.
- *AIX Version 4.3 Technical Reference: Communications Volume 2*, Order Number SC23−4162
- *AIX Version 4.3 Technical Reference: Kernel and Subsystems Volume 1*, Order Number SC23−4163.
- *AIX Version 4.3 Technical Reference: Kernel and Subsystems Volume 2*, Order Number SC23−4164.
- *AIX Version 4 Keyboard Technical Reference*, Order Number SC23−2631.
- *Distributed SMIT 2.2 for AIX: Guide and Reference*, Order Number SC23−2667.
- *3270 Host Connection Program 2.1 and 1.3.3 for AIX: Guide and Reference*, Order Number SC23−2563.

The following books also may be helpful:

- Lamb, Linda. *Learning the vi Editor*. Sebastopol, CA: O'Reilly & Associates, 1990. Order Number SR28−4966.

- Dougherty, Dale. *sed & awk*. Sebastopol, CA: O'Reilly & Associates, 1990. Order Number SR28−4968.
- Hunt, Craig. *TCP/IP Network Administration*. Sebastopol, CA: O'Reilly & Associates, 1992. Order Number SR23−7422.

Ordering Publications

You can order publications from your sales representative or from your point of sale.

To order additional copies of this book, use order number SC23−4119.

To order additional copies of all six volumes of *AIX Version 4.3 Commands Reference*, use Order Number SBOF−1877.

Use *AIX and Related Products Documentation Overview* for information on related publications and how to obtain them.

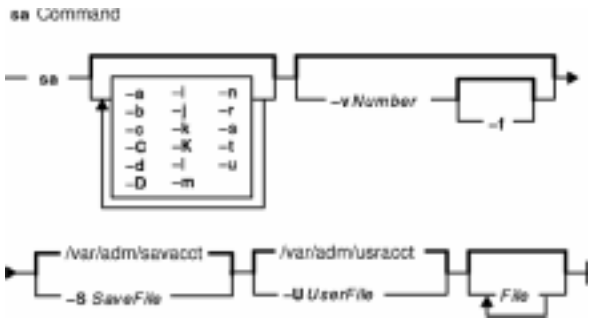
Alphabetical Listing of Commands

sa Command

Purpose

Summarizes accounting records.

Syntax



/usr/sbin/sa [**−a**] [**−b**] [**−c**] [**−C**][**−d**] [**−D**] [**−i**] [**−j**] [**−k**] [**−K**] [**−l**] [**−m**] [**−n**] [**−r**] [**−s**] [**−t**] [**−u**] [**−v***Number* [**−f**]] [**−S***SaveFile*] [**−U***UserFile*] [*File ...*]

Description

The **sa** command summarizes the information in the file that collects the raw accounting data, either the **/var/adm/pacct** file or the file specified by the *File* parameter, and writes a usage summary report to the **/var/adm/savacct** file. Then, the **sa** command deletes the data in the **/var/adm/pacct** file so it can collect new accounting information. The next time the **sa** command executes, it reads the usage summary and the new data and incorporates all the information in its report.

The flags used with the **sa** command vary the type of information that is reported. The reports can contain the following fields:

`avio`	Indicates the average number of I/O operations per execution.
`cpu`	Indicates the sum of user and system time (in minutes).
`k`	Indicates the average K−blocks of CPU−time per execution.
`k*sec`	Indicates the CPU storage integral in kilo−core seconds.
`re`	Indicates the minutes of real time.
`s`	Indicates the minutes of system CPU time.
`tio`	Indicates the total number of I/O operations.
`u`	Indicates the minutes of user CPU time.

If you run the **sa** command without specifying any flags, the summary report includes the number of times each command was called as well as the `re`, `cpu`, `avio`, and `k` fields.

> **Note:** The **−b**, **−d**, **−D**, **−k**, **−K**, and **−n** flags determine how output is sorted. If you specify more than one of these flags on the command line, only the last one specified will take effect.

Summary files created under this release of the base operating system are saved in a format that supports large user IDs (8 characters or longer). Summary files created under previous releases may be in the old format that supports only user IDs of up to 7 characters. The **sa** command recognizes and supports both

formats of the summary file. If you need to convert old format summary files to the new format, use the **−C** flag instead of the **−s** flag. You need to do this conversion only once. After converting you can use either the **−s** or the **−C** flag.

Flags

−a	Prints all command names, including those with unprintable characters. Commands that were used once are placed under the `other` category.
−b	Sorts output by the sum of user and system time divided by the number of calls. Otherwise, output is the sum of user and system time.
−c	Prints the time used by each command as a percentage of the time used by all the commands. This is in addition to the user, system and real time.
−C	Merges the accounting file into the summary file. If the summary file is in the old format, it is converted into the new format.
−d	Sorts the output by the average number of disk I/O operations.
−D	Sorts and prints the output by the total number of disk I/O operations.
−f	Does not force interactive threshold compression. This flag must be used with the **−v** flag.
−i	Reads only the raw data, not the summary file.
−j	Prints the number of seconds per call instead of the total minutes per category.
−k	Sorts the output by the average CPU time.
−K	Sorts and prints the output by the CPU−storage integral.
−l	Separates system and user time, instead of combining them.
−m	Prints the number of processes and the number of CPU minutes for each user.
−n	Sorts output by the number of calls.
−r	Reverses the order of the sort.
−s	Merges the accounting file into the summary file.
−S *SaveFile*	Uses the specified saved file as the command summary file, instead of the **/var/adm/savacct** file.
−t	Prints the ratio of real time to the sum of user and system time for each command.
−u	Suspends all other flags and prints the user's numeric ID and the command name for each command.
−U *UserFile*	Uses the specified file instead of the **/var/adm/usracct** file to accumulate the per−user statistics printed by the **−m** flag.
−v *Number*	Types the name of each command used the specified number times or fewer. When queried, if you type `y` (yes), the command is added to the `junk` category and appears in future summaries as part of that category.

Examples

1. To summarize accounting records for all the commands in the **/var/adm/pacct** file, enter:

   ```
   sa −a
   ```

   Commands used only once are placed under the `other` field.

2. To summarize accounting records by average CPU time, enter:

   ```
   sa −k
   ```

Files

/usr/sbin/sa	Contains the **sa** command.
/etc/sa	Contains the symbolic link to the **sa** command.
/var/adm/pacct	Contains raw accounting records.
/var/adm/savacct	Contains summary accounting records.
/var/adm/usracct	Contains summary accounting records by user.

Related Information

The **acctcms** command, **acctcom** command, **acctcon1** or **acctcon2** command, **acctmerg** command, **acctprc1**, **acctprc2**, or **accton** command, **fwtmp** command, **runacct** command.

For more information about the Accounting System, the preparation of daily and monthly reports, and the accounting files, see the Accounting Overview in the *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

Setting Up an Accounting System in the *AIX Version 4.3 System Management Guide: Operating System and Devices* describes the steps you must take to establish an accounting system.

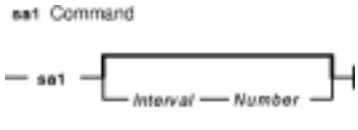
See the Accounting Commands in the *AIX Version 4.3 System Management Concepts: Operating System and Devices* for a list of accounting commands that can be run automatically or entered from the keyboard.

sa1 Command

Purpose

Collects and stores binary data in the **/var/adm/sa/sa***dd* file.

Syntax



/usr/lib/sa/sa1 [*Interval Number*]

Description

The **sa1** command is a shell procedure variant of the **sadc** command and handles all of the flags and parameters of that command. The **sa1** command collects and stores binary data in the **/var/adm/sa/sa***dd* file, where *dd* is the day of the month. The *Interval* and *Number* parameters specify that the record should be written *Number* times at *Interval* seconds. If you do not specify these parameters, a single record is written. You must have permission to write in the **/var/adm/sa** directory to use this command.

The **sa1** command is designed to be started automatically by the **cron** command. If the **sa1** command is not run daily from the **cron** command, the **sar** command displays a message about the nonexistence of the **/usr/lib/sa/sa1** data file.

Examples

To create a daily record of **sar** activities, place the following entry in your adm **crontab** file:

```
0 8-17 * * 1-5 /usr/lib/sa/sa1 1200 3 &
```

Files

/var/adm/sa	Specifies the directory containing the daily data files.
/var/adm/sa/sa*dd*	Contains the daily data file, where the *dd* parameter is a number representing the day of the month.
/usr/lib/sa/sa1	Contains the **sa1** command.

Related Information

The **sadc** command, **sar** command, **sa2** command.

The **cron** daemon.

For more information about the Accounting System, the preparation of daily and monthly reports, and the accounting files, see the Accounting Overview in the *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

Setting Up an Accounting System in the *AIX Version 4.3 System Management Guide: Operating System and Devices* describes the steps you must take to establish an accounting system.

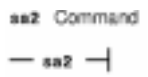
See the Accounting Commands in the *AIX Version 4.3 System Management Concepts: Operating System and Devices* for a list of accounting commands that can be run automatically or entered from the keyboard.

sa2 Command

Purpose

Writes a daily report in the **/var/adm/sa/sar***dd* file.

Syntax



/usr/lib/sa/sa2

Description

The **sa2** command is a variant shell procedure of the **sar** command, which writes a daily report in the **/var/adm/sa/sar***dd* file, where *dd* is the day of the month. The **sa2** command handles all of the flags and parameters of the **sar** command.

The **sa2** command is designed to be run automatically by the **cron** command and run concurrently with the **sa1** command.

Examples

To run the **sa2** command daily, place the following entry in the root **crontab** file:

```
5 18 * * 1-5 /usr/lib/sa/sa2 -s 8:00 -e 18:01 -i 3600 -ubcwyaqvm &
```

This will generate a daily report called **/var/adm/sa/sar***dd*. It will also remove a report more than one week old.

Files

/var/adm/sa	Specifies the directory containing the daily data files.
/var/adm/sa/sar*dd*	Contains daily data file, where the *dd* parameter is a number representing the day of the month.
/usr/lib/sa/sa2	The path to the shell script of the **sa2** command.

Related Information

The **sa1** command, **sadc** command, **sar** command.

The **cron** daemon.

Accounting Overview, Setting Up an Accounting System in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

Accounting Commands in *AIX Version 4.3 System Management Concepts: Operating System and*

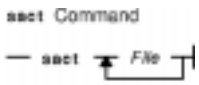
Devices lists accounting commands that can be run automatically or entered from the keyboard.

sact Command

Purpose

Displays current SCCS file−editing status.

Syntax



sact*File* ...

Description

The **sact** command reads Source Code Control System (SCCS) files and writes to standard output the contents, if any, of the p−file associated with the specified value of the *File* variable. The p−file is created by the **get −e** command. If a **−** (minus sign) is specified for the *File* value, the **sact** command reads standard input and interprets each line as the name of an SCCS file. If the *File* value is a directory, the **sact** command performs its actions on all SCCS files.

Exit Status

This command returns the following exit values:

0 Successful completion.
>0 An error occurred.

Examples

To display the contents of a p−file, enter:

```
sact File
```

Files

/usr/bin/sact Contains the path to the SCCS **sact** command.

Related Information

The **delta** command, **get**command, **sccs** command, **unget** command.

The **sccsfile** file format.

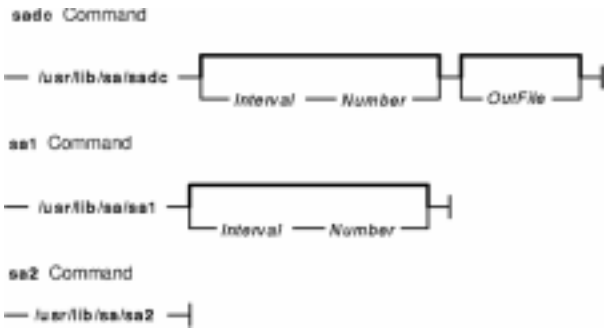
List of SCCS Commands, Source Code Control System (SCCS) Overview in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

sadc Command

Purpose

Provides a system data collector report.

Syntax



/usr/lib/sa/sadc [*Interval Number*] [*Outfile*]

/usr/lib/sa/sa1 [*Interval Number*]

/usr/lib/sa/sa2

Description

The **sadc** command, the data collector, samples system data a specified number of times (*Number*) at a specified interval measured in seconds (*Interval*). It writes in binary format to the specified outfile or to the standard output. When both *Interval* and *Number* are not specified, a dummy record, which is used at system startup to mark the time when the counter restarts from 0, will be written. The **sadc** command is intended to be used as a backend to the **sar** command.

The operating system contains a number of counters that are incremented as various system actions occur. The various system actions include:

- System unit utilization counters
- Buffer usage counters
- Disk and tape I/O activity counters
- Tty device activity counters
- Switching and subroutine counters
- File access counters
- Queue activity counters
- Interprocess communication counters

Note: The **sadc** command reports only local activity.

Security

Access Control: These commands should grant execute (x) access only to members of the **adm** group.

Examples

To write 10 records of one second intervals to the **/tmp/rpt** binary file, enter:

```
sadc 1 10 /tmp/rpt
```

Files

/var/adm/sa/sa*dd*	Contains the daily data file, *dd* represents the day of the month.
/var/adm/sa/sar*dd*	Contains the daily report file, *dd* represents the day of the month.
/tmp/rpt	Contains the binary file used for input by the **sar** command.
/tmp/sa.adrf1	Contains the address file.

Related Information

The **sar** command, **sa1** command, **sa2** command, **timex** command.

The **cron** daemon.

Accounting Overview, Setting Up an Accounting System in the *AIX Version 4.3 System Management Guide: Operating System and Devices*.

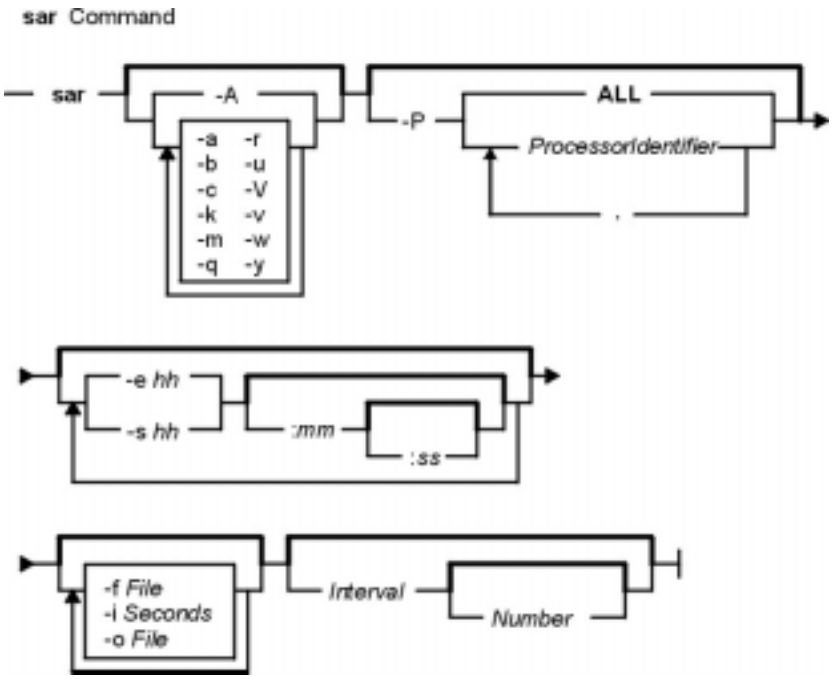
Accounting Commands in the *AIX Version 4.3 System Management Concepts: Operating System and Devices* lists accounting commands that can be run automatically or entered from the keyboard.

sar Command

Purpose

Collects, reports, or saves system activity information.

Syntax



/usr/sbin/sar [{ **−A** | [**−a**] [**−b**] [**−c**] [**−k**] [**−m**] [**−q**] [**−r**] [**−u**] [**−V**] [**−v**] [**−w**] [**−y**] }] [**−P** *ProcessorIdentifier***,** *...* | **ALL**] [**−e***hh* [**:***mm* [**:***ss*]]] [**−f***File*] [**−i***Seconds*] [**−o***File*] [**−s***hh* [**:***mm* [**:***ss*]]] [*Interval* [*Number*]]

Description

The **sar** command writes to standard output the contents of selected cumulative activity counters in the operating system. The accounting system, based on the values in the *Number* and *Interval* parameters, writes information the specified number of times spaced at the specified intervals in seconds. The default sampling interval for the *Number* parameter is 1 second. The collected data can also be saved in the file specified by the **−o***File* flag.

The **sar** command extracts and writes to standard output records previously saved in a file. This file can be either the one specified by the **−f** flag or, by default, the standard system activity daily data file, the **/var/adm/sa/sa***dd* file, where the *dd* parameter indicates the current day.

Without the **−P** flag, the **sar** command reports system−wide (global among all processors) statistics, which are calculated as averages for values expressed as percentages, and as sums otherwise. If the **−P** flag is given, the **sar** command reports activity which relates to the specified processor or processors. If **−P ALL** is given, the **sar** command reports statistics for each individual processor, followed by system−wide statistics.

You can select information about specific system activities using flags. Not specifying any flags selects only system unit activity. Specifying the **−A** flag selects all activities.

The default version of the **sar** command (CPU utilization report) might be one of the first facilities the user runs to begin system activity investigation, because it monitors major system resources. If CPU utilization is near 100 percent (user + system), the workload sampled is CPU−bound. If a considerable percentage of time is spent in I/O wait, it implies that CPU execution is blocked waiting for disk I/O. The I/O may be required file accesses or it may be I/O associated with paging due to a lack of sufficient memory.

> **Note:** The time the system spends waiting for *remote* file access is *not* accumulated in the I/O wait time. If CPU utilization and I/O wait time for a task are relatively low, and the response time is not satisfactory, consider investigating how much time is being spent waiting for remote I/O. Since no high−level command provides statistics on remote I/O wait, trace data may be useful in observing this.

If multiple samples and multiple reports are desired, it is convenient to specify an output file for the **sar** command. Direct the standard output data from the **sar** command to `/dev/null` and run the **sar** command as a background process. The syntax for this is:

```
sar −A −o data.file interval count > /dev/null &
```

All data is captured in binary form and saved to a file (`data.file`). The data can then be selectively displayed with the **sar** command using the **−f** option.

The **sar** command calls a process named **sadc** to access system data. Two shell scripts (`/usr/lib/sa/sa1` and `/usr/lib/sa/sa2`) are structured to be run by the **cron** command and provide daily statistics and reports. Sample stanzas are included (but commented out) in the **/var/spool/cron/crontab/adm** crontab file to specify when the **cron** daemon should run the shell scripts. Collection of data in this manner is useful to characterize system usage over a period of time and determine peak usage hours.

You can insert a dummy record into the standard system activity daily data file at the time of system start by uncommenting corresponding lines in the **/etc/rc** script. The sar command reports `time change not positive` for any record where CPU times are less than the previous record. This occurs if you reboot the system with the dummy record insertion lines in **/etc/rc** commented out.

> **Note:** The **sar** command only reports on local activities.

You can use the Web−based System Manager System application (**wsm system** fast path) to run this command. You could also use the System Management Interface Tool (SMIT) **smit sar** fast path to run this command.

Flags

−A Without the **−P** flag, using the **−A** flag is equivalent to specifying **−abckmqruvwy**. When used with the **−P** flag, the **−A** is equivalent to specifying **−acmuw**.

−a Reports use of file access system routines specifying how many times per second several of the system file access routines have been called. When used with the **−P** flag, the information is provided for each specified processor; otherwise, it is provided only system−wide. The following values are displayed:

- **dirblk/s** Number of 512−byte blocks read by the directory search routine to locate a directory entry for a specific file.
- **iget/s** Calls to any of several i−node lookup routines that support multiple file system types. The **iget** routines return a pointer to the i−node structure of a file or device.

lookuppn/s Calls to the directory search routine that finds the address of a v−node given a path name.

−b Reports buffer activity for transfers, accesses, and cache (kernel block buffer cache) hit ratios per second. Access to most files in AIX Version 3 bypasses kernel block buffering and therefore does not generate these statistics. However, if a program opens a block device or a raw character device for I/O, traditional access mechanisms are used making the generated statistics meaningful. The following values are displayed:

bread/s, bwrit/s Reports the number of block I/O operations. These I/Os are generally performed by the kernel to manage the block buffer cache area, as discussed in the description of the **lread/s** value.

lread/s, lwrit/s Reports the number of logical I/O requests. When a logical read or write to a block device is performed, a logical transfer size of less than a full block size may be requested. The system accesses the physical device units of complete blocks and buffers these blocks in the kernel buffers that have been set aside for this purpose (the block I/O cache area). This cache area is managed by the kernel, so that multiple logical reads and writes to the block device can access previously buffered data from the cache and require no real I/O to the device. Application read and write requests to the block device are reported statistically as logical reads and writes. The block I/O performed by the kernel to the block device in management of the cache area is reported as block reads and block writes.

pread/s, pwrit/s Reports the number of I/O operations on raw devices. Requested I/O to raw character devices is not buffered as it is for block devices. The I/O is performed to the device directly.

%rcache, %wcache Reports caching effectiveness (cache hit percentage). This percentage is calculated as: [(100)x(lreads − breads)/ (lreads)].

−c Reports system calls. When used with the **−P** flag, the information is provided for each specified processor; otherwise, it is provided only system−wide. The following values are displayed:

exec/s, fork/s Reports the total number of **fork** and **exec** system calls.

sread/s, swrit/s Reports the total number of read/write system calls.

rchar/s, wchar/s Reports the total number of characters transferred by read/write system calls.

scall/s Reports the total number of system calls.

Note: The **sar** command itself can generate a considerable number of reads and writes depending on the interval at which it is run. Run the **sar** statistics without the workload to understand the **sar** command's contribution to your total statistics.

−e *hh*[:*mm*[:*ss*]] Sets the ending time of the report. The default ending time is 18:00.

−f *File* Extracts records from *File* (created by **−o***File* flag). The default value of the *File* parameter is the current daily data file, the **/var/adm/sa/sa***dd* file.

−i *Seconds* Selects data records at seconds as close as possible to the number specified by the *Seconds* parameter. Otherwise, the **sar** command reports all seconds

found in the data file.

−k Reports kernel process activity. The following values are displayed:

- **kexit/s** Reports the number of kernel processes terminating per second.
- **kproc−ov/s** Reports the number of times kernel processes could not be created because of enforcement of process threshold limit.
- **ksched/s** Reports the number of kernel processes assigned to tasks per second.

−m Reports message (sending and receiving) and semaphore (creating, using, or destroying) activities per second. When used with the **−P** flag, the information is provided for each specified processor; otherwise, it is provided only system−wide. The following values are displayed:

- **msg/s** Reports the number of IPC message primitives.
- **sema/s** Reports the number of IPC semaphore primitives.

−o *File* Saves the readings in the file in binary form. Each reading is in a separate record and each record contains a tag identifying the time of the reading.

−P*ProcessorIdentifier,...* | **ALL** Reports per−processor statistics for the specified processor or processors. Specifying the **ALL** keyword reports statistics for each individual processor, and globally for all processors . Of the flags which specify the statistics to be reported, only the **−a**, **−c**, **−m**, **−u**, and **−w** flags are meaningful with the **−P** flag.

−q Reports queue statistics. The following values are displayed:

- **runq−sz** Reports the average number of kernel threads in the run queue.
- **%runocc** Reports the percentage of the time the run queue is occupied.
- **swpq−sz** Reports the average number of kernel threads waiting to be paged in.
- **%swpocc** Reports the percentage of the time the swap queue is occupied.

  Note: A blank value in any column indicates that the associated queue is empty.

−r Reports paging statistics. The following values are displayed:

- **cycle/s** Reports the number of page replacement cycles per second.
- **fault/s** Reports the number of page faults per second. This is not a count of page faults that generate I/O, because some page faults can be resolved without I/O.
- **slots** Reports the number of free pages on the paging spaces.
- **odio/s** Reports the number of nonpaging disk I/Os per second.

−s *hh*[:*mm*[:*ss*]] Sets the starting time of the data, causing the **sar** command to extract records time−tagged at, or following, the time specified. The default starting time is 08:00.

−u Reports per processor or system−wide statistics. When used with the **−P** flag, the information is provided for each specified processor; otherwise, it is provided only system−wide. Because the **−u** flag information is expressed as percentages, the system−wide information is simply the average of each individual processor's statistics. Also, the I/O wait state is defined system−wide and not per processor. The following values are displayed:

- **%idle** Reports the percentage of time the cpu or cpus were idle with no outstanding disk I/O requests.
- **%sys** Reports the percentage of time the cpu or cpus spent in execution at the system (or kernel) level.

%usr Reports the percentage of time the cpu or cpus spent in execution at the user (or application) level.

%wio Reports the percentage of time the cpu or cpus were idle waiting for disk I/O to complete. For system−wide statistics, this value may be slightly inflated if several processors are idling at the same time, an unusual occurence.

Note: The **sar** command reports system unit activity if no other specific content options are requested.

−V Reads the sar files created on previous versions of AIX. This flag can only be used with the **−f** flag.

−v Reports status of the process, kernel−thread, i−node, and file tables. The following values are displayed:

file−sz, inod−sz, **proc−sz, thrd−sz**

Reports the number of entries in use for each table.

−w Reports system switching activity. When used with the **−P** flag, the information is provided for each specified processor; otherwise, it is provided only system−wide. The following value is displayed:

pswch/s Reports the number of context switches per second.

−y Reports tty device activity per second.

canch/s Reports tty canonical input queue characters. This field is always 0 (zero) for AIX Version 4 and later versions.

mdmin/s Reports tty modem interrupts.

outch/s Reports tty output queue characters.

rawch/s Reports tty input queue characters.

revin/s Reports tty receive interrupts.

xmtin/s Reports tty transmit interrupts.

Security

Access Control: These commands should grant execute (x) access only to members of the **adm** group.

Examples

1. To report system unit activity, enter:

   ```
   sar
   ```

2. To report current tty activity for each 2 seconds for the next 20 seconds, enter:

   ```
   sar −y −r 2 20
   ```

3. To watch system unit for 10 minutes and sort data, enter:

   ```
   sar −o temp 60 10
   ```

4. To report cpu activity for the first two processors, enter:

   ```
   sar −u −P 0,1
   ```

   This produces output similar to the following:

   ```
   cpu  %usr  %sys  %wio  %idle
   0      45    45     5      5
   1      27    65     3      5
   ```

5. To report message, semaphore, and cpu activity for all processors and system−wide, enter:

   ```
   sar −mu −P ALL
   ```

   On a four−processor system, this produces output similar to the following (the last line indicates system−wide statistics for all processors):

   ```
   cpu  msgs/s  sema/s  %usr  %sys  %wio  %idle
   0      7       2       45    45     5     5
   1      5       0       27    65     3     5
   2      3       0       55    40     1     4
   3      4       1       48    41     4     7
   −     19       3       44    48     3     5
   ```

6. To read the system activity file called `File` generated on previous versions of AIX, enter:
   ```
   sar −V −f File
   ```

Files

/usr/sbin/sar	Contains the **sar** command.
/bin/sar	Indicates the symbolic link to the **sar** command.
/var/adm/sa/sa*dd*	Indicates the daily data file, where the *dd* parameter is a number representing the day of the month.

Related Information

The **sadc** command, **sa1** command, **sa2** command.

Accounting Overview in the *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

Setting Up an Accounting System in the *AIX Version 4.3 System Management Guide: Operating System and Devices*.

Accounting Commands Overview in the *AIX Version 4.3 System Management Concepts: Operating System and Devices* lists accounting commands that can be run automatically or entered from the keyboard.

AIX Performance Monitoring and Tuning Commands in *AIX Versions 3.2 and 4 Performance Tuning Guide*.

Setting up and running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

savebase Command

Purpose

Saves information about base−customized devices in the Device Configuration database onto the boot device.

Syntax

savebase [**−o** *Path*] [**−d** *File*] [**−v**]

Description

The **savebase** command stores customized information for base devices for use during phase 1 of system boot. By default, the **savebase** command retrieves this information from the **/etc/objrepos** directory. However, you can override this action by using the **−o** flag to specify an ODM directory. By default, the **savebase** command writes the information it retrieves to the boot disk. Alternatively, you can use the **−d** flag to specify a destination file or a device, such as the **/dev/hdisk0** device file.

The **savebase** command determines what device information to save using the `PdDv.base` field corresponding to each entry in the **CuDv** object class. Specifically, the `PdDv.base` field is a bit mask which represents the type of boot for which this device is a base device. The **savebase** command determines the current type of boot by accessing the **boot_mask** attribute in the **CuAt** object class. The value of this attribute is the bit mask to apply to the `PdDv.base` field to determine which devices are base.

> **Note:** Base devices are those devices that get configured during phase 1 boot; they may vary depending on the type of boot (mask). For example, if the mask is **NETWORK_BOOT**, network devices are considered base; for **DISK_BOOT**, disk devices are considered base. The type−of−boot masks are defined in the **/usr/include/sys/cfgdb.h** file.

Flags

−d *File* Specifies the destination file or device to which the base information will be written.

−o *Path* Specifies a directory containing the Device Configuration database.

−v Causes verbose output to be written to standard input.

Examples

1. To save the base customized information and see verbose output, enter:

   ```
   savebase −v
   ```

2. To specify an ODM directory other than the **/usr/lib/objrepos** directory, enter:

   ```
   savebase −o /tmp/objrepos
   ```

3. To save the base customized information to the **/dev/hdisk0** device file instead of to the boot disk, enter:

```
savebase −d /dev/hdisk0
```

Files

/usr/include/sys/cfgdb.h	Defines the type of boot mask for base devices.
/usr/lib/objrepos/PdDv	Contains entries for all known device types supported by the system.
/etc/objrepos/CuDv	Contains entries for all device instances defined in the system.
/etc/objrepos/CuAt	Contains customized device−specific attribute information.
/etc/objrepos/CuDep	Describes device instances that depend on other device instances.
/etc/objrepos/CuDvDr	Stores information about critical resources that need concurrency management through the use of the Device Configuration Library routines.

Related Information

The **bosboot** command, **restbase** command.

Object Data Manager (ODM) Overview for Programmers in *AIX General Programming Concepts: Writing and Debugging Programs*.

Device Configuration Subsystem: Programming Introduction, List of Device Configuration Commands in *AIX Kernel Extensions and Device Support Programming Concepts*.

savecore Command

Purpose

Saves a system dump.

Syntax

savecore { [**−c**] [**−d**] [**−f**] } *DirectoryName SystemName*

Description

The function of the **savecore** command is to save a system dump and is usually run at system startup.

The **savecore** command checks to see that you have a recent dump and that there is enough space to save it. The system dump is saved in the *DirectoryName*/**vmcore.***n* file, and the system is saved in the *DirectoryName*/**vmunix.***n* file. The *n* variable is specified in the *DirectoryName*/**bounds** file. If this file does not exist, it is created with a default of **0**, and the *n* variable uses this value. With each subsequent dump, the *n* variable is increased by 1.

The **savecore** command also checks to see if the current dump was compressed. If so, then it is copied to a file named *DirectoryName*/**vmcore**. *n*.**Z**, where .**Z** is the standard indication that a file is compressed.

> **Note**: This applies to AIX 4.3.2 and later.

If the system dump was from a system other than */unix*, the name of the system must be supplied as *SystemName*.

> **Note:** The **savecore** command saves only the current dump and the dump prior to the current one.

Flags

−c Marks the dump invalid (not recent), but does not copy it.
−d Copies only the dump. It does not copy the system.
−f Copies the dump even if it appears to be invalid.

Examples

1. To copy the dump (not the system) to *DirectoryName*, enter:

   ```
   savecore −d DirectoryName
   ```

2. To copy the dump even if it is invalid, enter:

   ```
   savecore −f −d DirectoryName
   ```

3. To mark the dump invalid, enter:

   ```
   savecore -c
   ```

4. To copy the dump and the system, enter:

   ```
   savecore -d DirectoryName SystemName
   ```

Related Information

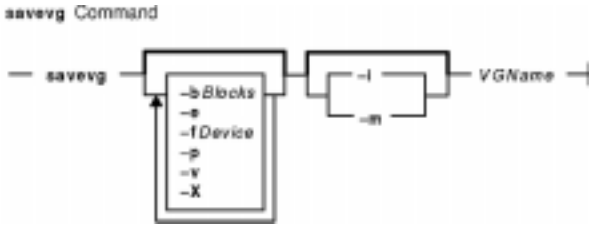
The **crash** command, **sysdumpdev** command, and **sysdumpstart** command.

savevg Command

Purpose

Finds and backs up all files belonging to a specified volume group.

Syntax



savevg [**−b** *Blocks*] [**−e**] [**−f** *Device*] [**−i** | **−m**] [**−p**] [**−v**] [**−X**] *VGName*

Description

> **Attention:** The **savevg** command will not generate a bootable tape if the volume group is the root volume group. Although the tape is not bootable, the first three images on the tape are dummy replacements for the images normally found on a bootable tape. The actual system backup is the fourth image. For more information about images on bootable tapes, please see Troubleshooting an Installation from a System Backup, General Information Regarding Mksysb System Backups in the *AIX Installation Guide*.

The **savevg** command finds and backs up all files belonging to a specified volume group. A volume group must be varied−on, and the file systems must be mounted. The **savevg** command uses the data file created by the **mkvgdata** command. This file can be one of the following:

/image.data	Contains information about the root volume group (**rootvg**). The **savevg** command uses this file to create a backup image that can be used by Network Installation Management (NIM) to reinstall the volume group to the current system or to a new system.
/tmp/vgdata/*vgname*/*vgname***.data**	Contains information about a user volume group. The *vgname* variable reflects the name of the volume group. The **savevg** command uses this file to create a backup image that can be used by the **restvg** command to remake the user volume group.

To create a backup of the operating system to CD, please refer to the **mkcd** command.

Flags

−b*Blocks* Specifies the number of 512−byte blocks to write in a single output operation. If this parameter is not specified, the **backup** command uses a default value appropriate for the physical device selected. Larger values result in larger physical transfers to tape devices. The value specified must be a multiple of the physical block size of the device being used.

−e Excludes files specified in the **/etc/exclude.***vgname* file from being backed up by this command.

> **Note:** If you want to exclude certain files from the backup, create the **/etc/exclude.rootvg** file, with an ASCII editor, and enter the patterns of file names that you do not want included in your system backup image. The patterns in this

file are input to the pattern matching conventions of the **grep** command to determine which files will be excluded from the backup. If you want to exclude files listed in the **/etc/exclude.rootvg** file, select the Exclude Files field and press the Tab key once to change the default value to yes.

For example, to exclude all the contents of the directory called scratch, edit the exclude file to read as follows:

```
/scratch/
```

For example, to exclude the contents of the directory called **/tmp**, and avoid excluding any other directories that have **/tmp** in the pathname, edit the exclude file to read as follows:

```
^./tmp/
```

All files are backed up relative to . (current working directory). To exclude any file or directory for which it is important to have the search match the string at the beginning of the line, use ^ (caret character) as the first character in the search string, followed by **.** (dot character), followed by the filename or directory to be excluded.

If the filename or directory being excluded is a substring of another filename or directory, use **^.** (caret character followed by dot character) to indicate that the search should begin at the beginning of the line and/or use **$** (dollar sign character) to indicate that the search should end at the end of the line.

−f*Device* Specifies the device or file name on which the image is to be stored. The default is the **/dev/rmt0** device.

−i Creates the data file by calling the **mkvgdata** command.

−m Creates the data file with map files by calling the **mkvgdata** command with the **−m** flag.

−p Disables software packing of the files as they are backed up. Some tape drives use their own packing or compression algorithms. This flag only applies to AIX Version 4.2 or later.

−v Verbose mode. Lists files as they are backed up. This flag only applies to AIX Version 4.2 or later.

−X Specifies to automatically expand the **/tmp** file system if necessary. The **/tmp** file system may need to be extended to make room for the boot image when creating a bootable backup to tape. This flag only applies to AIX Version 4.2 or later.

Parameters

VGName Specifies the name of the volume group to be backed up.

Examples

1. To backup the root volume group (operating system image) to the **/mysys/myvg/myroot** backup file and create an **/image.data** file, enter:

   ```
   savevg −i −f/mysys/myvg/myroot rootvg
   ```

2. To backup the **uservg** volume group to the default tape drive (**dev/rmt0**) and create a new **uservg.data** file, enter:

   ```
   savevg −i uservg
   ```

3. To backup the **data2** volume group and create map files along with a new **data2.data** file on **rmt1** device, enter:

```
savevg -mf/dev/rmt1 data2
```

4. To backup the **data2** volume group, excluding the files listed in the **/etc/exclude.data2** file, enter:

   ```
   savevg -ief/dev/rmt1 data2
   ```

Files

/image.data	Used when the volume group is **rootvg**.
/tmp/vgdata/*vgname***/***vgname***.data**	Used when the volume group is not **rootvg** and where *vgname* is the name of the volume group.

Related Information

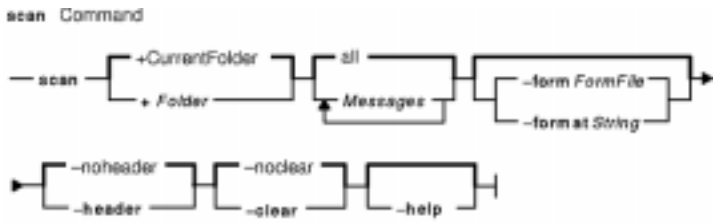
The **backup** command, **bosboot** command, **mkcd** command, **mkszfile** command.

scan Command

Purpose

Produces a one line per message scan listing.

Syntax



scan [*+Folder*] [*Messages*] [**−form** *FormFile* | **−format** *String*] [**−noheader** | **−header**] [**−clear** | **−noclear**] [**−help**]

Description

The **scan** command displays a line of information about the messages in a specified folder. Each line gives the message number, date, sender, subject, and as much of the message body as possible. By default, the **scan** command displays information about all of the messages in the current folder.

If a + (plus sign) is displayed after the message number, the message is the current message in the folder. If a − (minus sign) is displayed, you have replied to the message. If an * (asterisk) is displayed after the date, the `Date:` field was not present and the displayed date is the last date the message was changed.

Flags

−clear Clears the display after sending output. The **scan** command uses the values of the **$TERM** environment variable to determine how to clear the display. If standard output is not a display, the **scan** command sends a form feed character after sending the output.

+Folder Specifies which folder to scan. The default is the current folder.

−form *FormFile* Displays the **scan** command output in the alternate format described by the *FormFile* variable.

−format *String* Displays the **scan** command output in the alternate format described by the *String* variable.

−header Displays a heading that lists the folder name and the current date and time.

−help Lists the command syntax, available switches (toggles), and version information.
Note: For Message Handler (MH), the name of this flag must be fully spelled out.

Messages Displays information about each specified message in the specified folder. You can use the following references when specifying messages:

Number Specifies the number of the message.

Sequence Specifies a group of messages specified by the user. Recognized values include:

all	All messages in a folder. This is the default.
cur or **.** (period)	Current message.
first	First message in a folder.
last	Last message in a folder.

next	Message following the current message.
prev	Message preceding the current message.

−noclear Prevents clearing of the terminal after sending output. This is the default.

−noheader Prevents display of a heading. This is the default.

−width *Number* Sets the number of columns in the **scan** command output. The default is the width of the display.

Profile Entries

The following entries are entered in the *UserMhDirectory*/**.mh_profile** file:

`Alternate−Mailboxes:`	Specifies the mailboxes.
`Current−Folder:`	Sets the default current folder.
`Path:`	Specifies the *UserMhDirectory*.

Examples

1. To get a one−line list of all the messages in the current folder, enter:

   ```
   scan
   ```

   The system responds with a message similar to the following:

   ```
   3   04/17 dale@athena Status meeting <<The weekly status meeting
   5   04/20 tom@venus   Due Dates      <<Your project is due to
   6   04/21 dawn@tech   Writing Clas   <<There will be a writing
   ```

2. To get a one−line list of messages `11` through `15` in the `test` folder, enter:

   ```
   scan +test 11−15
   ```

   The system responds with a message similar to the following:

   ```
   11  04/16 karen@anchor Meeting    <<Today's meeting is at 2 p.m.
   12  04/18 tom@venus    Luncheon   <<There will be a luncheon to
   14  04/20 dale@athena  First Draft  <<First drafts are due
   15  04/21 geo@gtwn     Examples   <<The examples will be written
   ```

Files

$HOME/.mh_profile	Contains the MH user profile.
/etc/mh/scan.size	Contains a sample scan format string.
/etc/mh/scan.time	Contains a sample scan format string.
/etc/mh/scan.timely	Contains a sample scan format string.
/usr/bin/scan	Contains the executable form of the **scan** command.

Related Information

The **inc** command, **pick** command, **show** command.

The **.mh_alias** file format, **.mh_profile** file format.

Mail Overview in *AIX Version 4.3 System User's Guide: Communications and Networks*.

Peek, Jerry. *MH and xmh: E−mail for Users and Programmers.* Sebastopol, CA: O'Reilly & Associates, 1992.

sccs Command

Purpose

Administration program for SCCS commands.

Syntax

sccs [**−r**] [**−d***Path*] [**−p***Path*] *Command* [*CommandFlags*] *File ...*

Description

The **sccs** command is an administration program that incorporates the set of Source Code Control System (SCCS) commands into the operating system. Additionally, the **sccs** command can be used to assign or reassign file ownership (see the **−r** flag).

The **sccs** command activates a specified *Command* having the specified flags and arguments. Each file is normally placed in a directory named SCCS and named **s.filename**. The directory SCCS is assumed to exist relative to the working directory (unless the **−p** flag is used).

Two types of commands can be used in the **sccs** command syntax sentence. The first type consists of 14 **sccs** commands that can be entered at the prompt. The second type, pseudo−commands, can be used only as part of the **sccs** command syntax. There are 12 pseudo−commands, which perform the following actions:

edit Equivalent to the **get −e** command.

delget Performs a **delta** command on the named files and then gets new versions. The new versions of the files have expanded identification keywords and are not editable.

Flags:

−m, **−p**, **−r**, **−s**, **−y**
Can be passed to the **delta** command.
−b, **−c**, **−i**, **−l**, **−s**, **−x**
Can be passed to the **get** command.

deledit Equivalent to the **delget** pseudo−command, except that the **get** portion of the sentence includes the **−e** flag. The **deledit** pseudo−command is useful for creating a checkpoint in your current editing session.

Flags:

−m, **−p**, **−r**, **−s**, **−y**
Can be passed to the **delta** command.
−b, **−c**, **−i**, **−l**, **−s**, **−x**
Can be passed to the **get** command.

create Creates an SCCS file, copying the initial contents from a file of the same name. If the file is successfully created, the original file is renamed with a comma on the front. You do not have to move or remove the original file as with the **admin** command.

Flags:

Accepts the same flags as the **admin** command. The **−i** flag is implied.

fix Removes a named delta, but leaves a copy of the delta with changes intact. This pseudo−command is useful for fixing small compiler errors. This pseudo−command does not keep a record of changes made to the file.

Flags:

−r*SID* Indicates a required flag.

clean Removes all files from the current directory or from the designated directory that can be recreated from SCCS files. Does not remove files that are in the process of being edited.

Flags:

−b Ignores branches when determining which files are being edited. Branches being edited in the same directory can be lost.

unedit Equivalent to the **unget** command. Any changes made since the **get** command was used are lost.

info Lists all files being edited.

Flags:

−b Ignores branches when determining which files are being edited.

−u [*Argument*] Lists only the files being edited by you or the user named by the *Argument* parameter.

check Prints all files being edited. Returns a nonzero exit status if a file is being edited. The check program can be used in a makefile to ensure that files are complete before a version is installed. Check the return code before performing the install.

Flags:

−b Ignores branches when determining which files are being edited.

−u [*Argument*] Lists only the files being edited by you or the user named by the *Argument* parameter.

tell Lists all files being edited, with a new line after each entry, on standard output.

Flags:

−b Ignores branches when determining which files are being edited.

−u [*Argument*] Lists only the files being edited by you or the user named by the *Argument* parameter.

diffs Shows the difference between the current version of the program you are editing and the previous deltas.

Flags:

−r, **−c**, **−i**, **−x**, **−t**

Can be passed to the **get** command.

−l, **−s**, **−e**, **−f**, **−h**, **−b**
Can be passed to the **diff** (not **sccsdiff**) command.

−C Can be passed to the **diff** (not **sccsdiff**) command as a **−c** flag.

print (*filename*(*s*)) Prints **verbose** information about the named files.

If the **PROJECTDIR** environment variable is set, its value determines the working directory. If this value begins with a / (slash), it is used directly. Otherwise, the value is interpreted as a user name whose home directory is examined for a subdirectory named **src** or **source**. If found, that subdirectory is used as the working directory.

Flags

−d*Path* Specifies a working directory for the SCCS files. The default is the current directory. The **−d** flag is prefixed to the entire path name of a file. When the **PROJECTDIR** environment variable is set and the **−d** flag is used, the command line overrides the environment value in determining the working directory.

−p Specifies a path name for the SCCS files. The default is the SCCS directory. The **−p** flag is inserted before the final component of the path name.

All flags specified after the command are passed to that command during execution. For a description of command flags, see the appropriate command description.

Example:

```
sccs −d/x −py get a/b
```

converts to:

```
get /x/a/y/s.b
```

This option is used to create aliases. For example:

```
alias syssccs sccs −d/usr/src
```

causes the **syssccs** command to become an alias command that can be used as follows:

```
syssccs get cmd/who.c
```

When used in this context, the above command will check the **/usr/src/cmd/SCCS** directory for the **s.who.c** file.

−r Runs the **sccs** command as the real user instead of as the effective user to which the **sccs** command is set (using the **set user id** command).

Certain commands, such as the **admin** command, cannot be run as **set user id**, which would allow anyone to change the authorizations. Such commands are always run as the real user.

Exit Status

This command returns the following exit values:

0 Successful completion.

>0 An error occurred.

Examples

1. To get a file for editing, edit it, and then produce a new delta, enter:

   ```
   sccs get -e file.c
   ex file.c
   sccs delta file.c
   ```

2. To get a file from another directory, enter:

   ```
   sccs -p/usr/src/sccs/ get cc.c
   ```

   OR

   ```
   sccs get /usr/src/sccs/s.cc.c
   ```

3. To get a list of files being edited that are not on branches, enter:

   ```
   sccs info -b
   ```

Files

/usr/bin/sccs Contains the **sccs** command, which is the administration program for the SCCS commands.

Related Information

The **admin** command, **cdc** command, **comb** command, **delta** command, **diff** command, **get** command, **prs** command, **rmdel** command, **sact** command, **sccsdiff** command, **sccshelp** command, **unget** command, **val**command, **vc** command, **what** command.

The **sccsfile** file format.

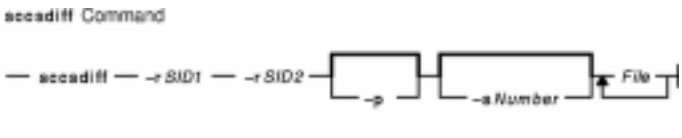
List of SCCS Commands, Source Code Control System (SCCS) Overview in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

sccsdiff Command

Purpose

Compares two versions of a SCCS file.

Syntax



sccsdiff −**r***SID1*−**r***SID2* [−**p**] [−**s***Number*] *File* ...

Description

The **sccsdiff** command reads two versions of an Source Code Control System (SCCS) file, compares them, and then writes to standard output the differences between the two versions. Any number of SCCS files can be specified, but the same arguments apply to all files.

Flags

−**p**	Pipes the output through the **pr** command.
−**r***SID1*	Specifies *SID1* as one delta of the SCCS file for the **sccsdiff** command to compare.
−**r***SID2*	Specifies *SID2* as the other delta of the SCCS file for the **sccsdiff** command to compare.
−**s***Number*	Specifies the file−segment size for the **bdiff** command to pass to the **diff** command. This is useful when the **diff** command fails due to a high system load.

Examples

To display the difference between versions 1.1 and 1.2 of SCCS file `s.test.c`, enter:

```
sccsdiff −r1.1 −r1.2 s.test.c
```

Files

/usr/bin/sccsdiff	Contains the SCCS **sccsdiff** command. The **sccsdiff** command supports multibyte character set (MBCS) data for the file names.

Related Information

The **bdiff** command, **diff** command, **get** command, **prs** command, **sccshelp** command.

The **sccsfile** file format.

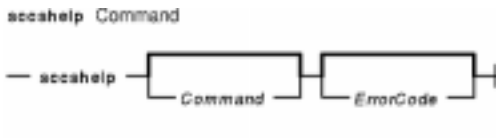
List of SCCS Commands, Source Code Control System (SCCS) Overview in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

sccshelp Command

Purpose

Provides information about a SCCS message or command.

Syntax



sccshelp [*ErrorCode*] [*Command*]

Description

The **sccshelp** command displays information about the use of a specified Source Code Control System (SCCS) command or about messages generated while using the commands. Each message has an associated code, which can be supplied as part of the argument to the **sccshelp** command. Zero or more arguments may be supplied. If you do not supply an argument, the **sccshelp** command prompts for one. You may include any of the SCCS commands as arguments to the **sccshelp** command.

The *ErrorCode* parameter specifies the code, consisting of numbers and letters, that appears at the end of a message. For example, in the following message, (cm7) is the code:

```
There are no SCCS identification keywords in the file. (cm7)
```

Examples

To get **sccshelp** on the **rmdel** command and two error codes, enter:

```
$ sccshelp rmdel gee ad3
```

The **sccshelp** command replies:

```
rmdel:
rmdel −r<SID> <file> ...
ERROR:
1255−141 gee is not a valid parameter. Specify a valid command or error code.
ad3:
The header flag you specified is not recognized.
The header flag you supplied with the −d or the −f flag is not correct.
Choose a valid header flag.
```

File

/usr/bin/sccshelp Contains the SCCS **sccshelp** command.

Related Information

The **admin** command, **cdc** command, **comb** command, **delta** command, **get** command, **prs** command, **rmdel** command, **sccsdiff** command, **what** command.

The **sccsfile** file format.

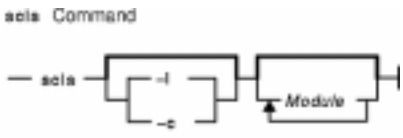
List of SCCS Commands, Source Code Control System (SCCS) Overview in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

scls Command

Purpose

Produces a list of module and driver names.

Syntax



scls [**−c** | **−l**] [*Module* ...]

Description

The **scls** command provides a method for the user to query the current Portable Streams Environment (PSE) configuration. The **scls** command produces a list of module and driver names. Flags can be used to produce enhanced lists. Any further parameters on the command line are module or driver names, and the output produced is for only those names.

> **Note:** The **scls** command requires the **sc** STREAMS module and the **nuls** driver. If either one is not available, the **scls** command will not be successful.

Flags

−c Produces a listing showing the number of times an interface routine was called.

−l Produces a long listing that shows the extension type, major number, and information pertaining to the **module_info** structure.

The **−c** and **−l** flags are mutually exclusive.

Parameters

module Specifies the name of the modules or drivers for which to output information.

Files

sc Dynamically loadable STREAMS configuration module

nuls Dynamically loadable STREAMS null device.

Related Information

The **strload** command.

List of Streams Commands in *AIX Communications Programming Concepts*.

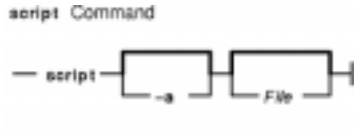
Configuring Drivers and Modules in the Portable Streams Environment (PSE), STREAMS Overview in *AIX Version 4.3 Communications Programming Concepts*.

script Command

Purpose

Makes a typescript of a terminal session.

Syntax



script [**−a**] [*File*]

Description

The **script** command makes a typescript of everything displayed on your terminal. The typescript is written to the file specified by the *File* parameter. The typescript can later be sent to the line printer. If no file name is given, the typescript is saved in the current directory with the file name **typescript**.

The script ends when the forked shell exits.

This command is useful for producing hardcopy records when hardcopy terminals are in short supply. For example, use the **script** command when you are working on a CRT display and need a hardcopy record of the dialog.

Since the **script** command sets the **SetUserID** mode bit, due to security reasons the value of LIBPATH variable is unset when the command is invoked. However, LIBPATH is automatically reset in the forked shell if it is defined in the environment file. For related information, see the **exec** subroutine.

Flags

−a Appends the typescript to the specified file or to the **typescript** file.

Files

/usr/bin/script Contains the **script** command.

Related Information

The **tee** command.

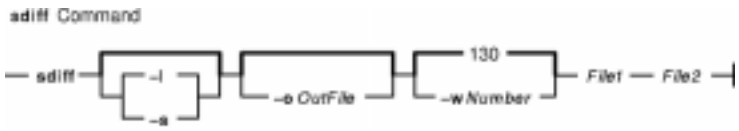
Input and Output Redirection Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* describes how the operating system processes input and output and how to use the redirect and pipe symbols.

sdiff Command

Purpose

Compares two files and displays the differences in a side−by−side format.

Syntax



sdiff [**−l** | **−s**] [**−o** *OutFile*] [**−w** *Number*] *File1File2*

Description

The **sdiff** command reads the files specified by the *File1* and *File2* parameters, uses the **diff** command to compare them, and writes the results to standard output in a side−by−side format. The **sdiff** command displays each line of the two files with a series of spaces between them if the lines are identical. It displays a < (less than sign) in the field of spaces if the line only exists in the file specified by the *File1* parameter, a > (greater than sign) if the line only exists in the file specified by the *File2* parameter, and a | (vertical bar) for lines that are different.

When you specify the **−o** flag, the **sdiff** command merges the files specified by the *File1* and *File2* parameters and produces a third file.

Note: The **sdiff** command invokes the **diff −b** command to compare two input files. The **−b** flag causes the **diff** command to ignore trailing spaces and tab characters and to consider other strings of spaces as equal.

Flags

−l Displays only the left side when lines are identical.

−o *OutFile* Creates a third file, specified by the *OutFile* variable, by a controlled line−by−line merging of the two files specified by the *File1* and the *File2* parameters. The following subcommands govern the creation of this file:

- **e** Starts the **ed** command with an empty file.
- **e b** or **e |** Starts the **ed** command with both sides.
- **e l** or **e <** Starts the **ed** command with the left side.
- **e r** or **e >** Starts the **ed** command with the right side.
- **l** Adds the left side to the output file.
- **r** Adds the right side to the output file.
- **s** Stops displaying identical lines.
- **v** Begins displaying identical lines.
- **q** Performs one of the following functions:
  - Exits the **ed** command.
  - Exits the **sdiff** command if no **ed** command is running.
  - Exits both commands. This action occurs when there are no more lines to be merged into the output file.

Each time you exit from the **ed** command, the **sdiff** command writes the resulting edited file to the end of the file specified by the *OutFile* variable. If you do not save the changes before exiting (for example, you press the Ctrl−C key sequence), the **sdiff** command writes the initial input to the output file.

−s Does not display identical lines.

−w *Number* Sets the width of the output line. The default value of the *Number* variable is 130 characters. The maximum width of the *Number* variable is 2048. The minimum width of the *Number* variable is 20. The **sdiff** command uses 2048 if a value greater than 2048 is specified.

Examples

1. To print a comparison of two files, enter:

   ```
   sdiff chap1.bak chap1
   ```

   The **sdiff** command displays a side−by−side listing that compares each line of the `chap1.bak` and `chap1` files.

2. To display only the lines that differ, enter:

   ```
   sdiff −s −w 80 chap1.bak chap1
   ```

   The **sdiff** command displays the differences at the work station. The `−w 80` flag and variable sets the page width to 80 columns. The `−s` flag indicates lines that are identical in both files will not be displayed.

3. To selectively combine parts of two files, enter:

   ```
   sdiff −s −w 80 −o chap1.combo chap1.bak chap1
   ```

   The **sdiff** command combines the `chap1.bak` and `chap1` files into a new file called `chap1.combo`. For each group of differing lines, the **sdiff** command prompts you which group to keep or whether you want to edit them using the **ed** command.

4. To combine and edit two files, `staff.jan` and `staff.apr`, and write the results to the `staff.year` file, perform the steps indicated.

   The `staff.jan` file contains the following lines:

   ```
   Members of the Accounting Department
   Andrea
   George
   Karen
   Sam
   Thomas
   ```

   The `staff.apr` file contains the following lines:

   ```
   Members of the Accounting Department
   Andrea
   Fred
   Mark
   Sam
   Wendy
   ```

   a. Enter the following command:

      ```
      sdiff −o staff.year staff.jan staff.apr
      ```

The **sdiff** command will begin to compare the contents of the `staff.jan` and `staff.apr` files and write the results to the `staff.year` file. The **sdiff** command displays the following:

```
Members of the Accounting Dept   Members of the Accounting Dept
Andrea                            Andrea
George                          | Fred
%
```

The % (percent sign) is the command prompt.

b. Enter the **e b** subcommand to start editing the output file with the **ed** command.

   The **sdiff** command displays a sequence of digits, indicating the byte count of lines being merged. In this case, the byte count is `23`.

c. Enter the **q** subcommand to exit the **ed** command and continue combining and editing the two files. The **sdiff** command displays the following:

```
Sam                               Sam
Thomas                          | Wendy
```

d. Enter the **e b** subcommand again. The **ed** command must be run each time a set of lines from the original two files are to be merged into the output file. The byte count in this instance is `13`.
e. Enter the **q** subcommand to save the changes. When all the lines of the two files have been merged into the output file, the **q** subcommand exits the **ed** and **sdiff** commands.

The `staff.year` file now contains the following:

```
Members of the Accounting Department
Andrea
George
Karen
Fred
Mark
Sam
Thomas
Wendy
```

Files

/usr/bin/sdiff Contains the **sdiff** command.

Related Information

The **diff** command, **ed** command.

Files Overview, Input and Output Redirection Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

securetcpip Command

Purpose

Enables the operating system network security feature.

Syntax

securetcpip

Description

The **securetcpip** command provides enhanced security for the network. This command performs the following:

1. Runs the **tcbck−a** command, which disables the nontrusted commands and daemons: **rcp**, **rlogin**, **rlogind**, **rsh**, **rshd**, **tftp**, and **tftpd**. The disabled commands and daemons are not deleted; instead, they are changed to mode 0000. You can enable a particular command or daemon by re−establishing a valid mode.
2. Adds a TCP/IP security stanza to the **/etc/security/config** file. The stanza is in the following format:

```
tcpip:
   netrc = ftp,rexec     /* functions disabling netrc */
```

Before running the **securetcpip** command, acquiesce the system by logging in as root user and executing the **killall** command to stop all network daemons.

> **Attention:** The **killall** command kills all processes except the calling process. If logged in or applications are running, exit or finish before executing the **killall** command.

After issuing the **securetcpip** command, shut down and restart your system. All of your TCP/IP commands and network interfaces should be properly configured after the system restarts.

Files

/etc/security/config Contains information for the security system.

Related Information

The **killall** command, **tcbck** command.

The **.netrc** file format.

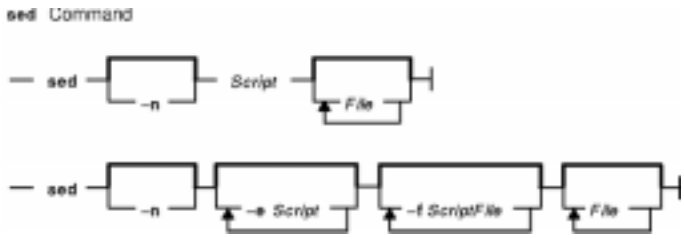
Understanding Trusted Processes for TCP/IP in *AIX Version 4.3 System Management Guide: Communications and Networks*.

sed Command

Purpose

Provides a stream editor.

Syntax



sed [**−n**] *Script* [*File* ...]

sed [**−n**] [**−e** *Script*] ... [**−f** *ScriptFile*] ... [*File* ...]

Description

The **sed** command modifies lines from the specified *File* parameter according to an edit script and writes them to standard output. The **sed** command includes many features for selecting lines to be modified and making changes only to the selected lines.

The **sed** command uses two work spaces for holding the line being modified: the pattern space, where the selected line is held; and the hold space, where a line can be stored temporarily.

An edit script consists of individual subcommands, each one on a separate line. The general form of **sed** subcommands is the following:

[*address−range*] *function*[*modifiers*]

The **sed** command processes each input *File* parameter by reading an input line into a pattern space, applying all **sed** subcommands in sequence whose addresses select that line, and writing the pattern space to standard output. It then clears the pattern space and repeats this process for each line specified in the input *File* parameter. Some of the **sed** subcommands use a hold space to save all or part of the pattern space for subsequent retrieval.

When a command includes an address (either a line number or a search pattern), only the addressed line or lines are affected by the command. Otherwise, the command is applied to all lines.

An address is either a decimal line number, a $ (dollar sign), which addresses the last line of input, or a context address. A context address is a regular expression similar to those used in the **ed** command except for the following differences:

- You can select the character delimiter for patterns. The general form of the expression is:

  ```
  \?pattern?
  ```

  where ? (question mark) is a selectable character delimiter. You can select any character from the current locale except for the space or new−line character. The \ (backslash) character is required only

for the first occurrence of the ? (question mark).

The default form for the pattern is the following:

```
/pattern/
```

A \ (backslash) character is not necessary.

- The **\n** sequence matches a new−line character in the pattern space, except the terminating new−line character.
- A **.** (period) matches any character except a terminating new−line character. That is, unlike the **ed** command, which cannot match a new−line character in the middle of a line, the **sed** command can match a new−line character in the pattern space.

Certain commands called *addressed* commands allow you to specify one line or a range of lines to which the command should be applied. The following rules apply to addressed commands:

- A command line without an address selects every line.
- A command line with one address, expressed in context form, selects each line that matches the address.
- A command line with two addresses separated by commas selects the entire range from the first line that matches the first address through the next line that matches the second. (If the second address is a number less than or equal to the line number first selected, only one line is selected.) Thereafter, the process is repeated, looking again for the first address.

Flags

−e *Script*	Uses the *Script* variable as the editing script. If you are using just one **−e** flag and no **−f** flag, the **−e** flag can be omitted.
−f *ScriptFile*	Uses the *ScriptFile* variable as the source of the edit script. The *ScriptFile* variable is a prepared set of editing commands applied to the *File* parameter.
−n	Suppresses all information normally written to standard output.

Note: You can specify multiple **−e** and **−f** flags. All subcommands are added to the script in the order specified, regardless of their origin.

sed Subcommands

The **sed** command contains the following **sed** script subcommands. The number in parentheses preceding a subcommand indicates the maximum number of permissible addresses for the subcommand.

Notes:

1. The *Text* variable accompanying the **a**, **c**, and **i** subcommands can continue onto more than one line, provided all lines but the last end with a \ (backslash) to quote the new−line character. Backslashes in text are treated like backslashes in the replacement string of an **s** command and can be used to protect initial blanks and tabs against the stripping that is done on every script line. The *RFile* and *WFile* variables must end the command line and must be preceded by exactly one blank. Each *WFile* variable is created before processing begins.
2. The **sed** command can process up to 999 subcommands in a pattern file.

(1) **a**

Text Places the *Text* variable in output before reading the next input line.

(2)**b[***label***]** Branches to the **:** command bearing the *label* variable. If the *label* variable is empty, it branches to the end of the script.

(2)**c**
Text Deletes the pattern space. With 0 or 1 address or at the end of a 2−address range, places the *Text* variable in output and then starts the next cycle.

(2)**d** Deletes the pattern space and then starts the next cycle.

(2)**D** Deletes the initial segment of the pattern space through the first new−line character and then starts the next cycle.

(2)**g** Replaces the contents of the pattern space with the contents of the hold space.

(2)**G** Appends the contents of the hold space to the pattern space.

(2)**h** Replaces the contents of the hold space with the contents of the pattern space.

(2)**H** Appends the contents of the pattern space to the hold space.

(1)**i**
Text Writes the *Text* variable to standard output before reading the next line into the pattern space.

(2)**l** Writes the pattern space to standard output showing nondisplayable characters as 4−digit hexadecimal values. Long lines are folded.

(2)**l** Writes the pattern space to standard output in a visually unambiguous form. The characters \\\, \\a, \\b, \\f, \\r, \\t, and \\v are written as the corresponding escape sequence. Non−printable characters are written as 1 three−digit octal number (with a preceding backslash character) for each byte in the character (most significant byte first). This format is also used for multibyte characters. This subcommand folds long lines. A backslash followed by a new−line character indicates the point of folding. Folding occurs at the 72nd column position. A $ (dollar sign) marks the end of each line.

(2)**n** Writes the pattern space to standard output if the default output is not suppressed. It replaces the pattern space with the next line of input.

(2)**N** Appends the next line of input to the pattern space with an embedded new−line character (the current line number changes). You can use this to search for patterns that are split onto two lines.

(2)**p** Writes the pattern space to standard output.

(2)**P** Writes the initial segment of the pattern space through the first new−line character to standard output.

(1)**q** Branches to the end of the script. It does not start a new cycle.

(2)**r** *RFile* Reads the contents of the *RFile* variable. It places contents in output before reading the next input line.

(2)**s/***pattern*/*replacement*/*flags*

Substitutes the *replacement* string for the first occurrence of the *pattern* parameter in the pattern space. Any character that is displayed after the **s** subcommand can substitute for the **/** (slash) separator except for the space or new−line character.

See the "Pattern Matching" section of the **ed** command.

The value of the *flags* variable must be zero or more of:

	g Substitutes all non−overlapping instances of the *pattern* parameter rather than just the first one.
	n Substitutes for the *n−th* occurrence only of the *pattern* parameter.
	p Writes the pattern space to standard output if a replacement was made.
	w *WFile* Writes the pattern space to the *WFile* variable if a replacement was made. Appends the pattern space to the *WFile* variable. If the *WFile* variable was not already created by a previous write by this **sed** script, the **sed** command creates it.
(2)**t***label*	Branches to the **:***label* variable in the script file if any substitutions were made since the most recent reading of an input line execution of a **t** subcommand. If you do not specify the *label* variable, control transfers to the end of the script.
(2)**w***WFile*	Appends the pattern space to the *WFile* variable.
(2)**x**	Exchanges the contents of the pattern space and the hold space.
(2)**y**/*pattern1*/*pattern2*/	Replaces all occurrences of characters in the *pattern1* variable with the corresponding *pattern2* characters. The number of characters in the *pattern1* and *pattern2* variables must be equal. The new−line character is represented by **\n**.
(2)**!***sed−cmd*	Applies the specified **sed** subcommand only to lines not selected by the address or addresses.
(0)**:***label*	Marks a branch point to be referenced by the **b** and **t** subcommands. This label can be any sequence of eight or fewer bytes.
(1)**=**	Writes the current line number to standard output as a line.
(2)**{***subcmd* **.** **.** **.** **}**	Groups subcommands enclosed in {} (braces).
(0)	Ignores an empty command.
(0)**#**	If a # (pound sign) appears as the first character on a line of a script file, that entire line is treated as a comment, with one exception. For the first line of a script file only, if the character after the # is an n, the default output will be suppressed. The rest of the line after the #n is ignored.

Exit Status

This command returns the following exit values:

0 Successful completion.

>**0** An error occurred.

Examples

1. To perform a global change, enter:

   ```
   sed  "s/happy/enchanted/g" chap1  >chap1.new
   ```

   This command sequence replaces each occurrence of the word `happy` found in the file `chap1` with the word `enchanted`. It puts the edited version in a separate file named `chap1.new`. The **g** character at the end of the **s** subcommand tells the **sed** command to make as many substitutions as possible on each line. Without the **g** character, the **sed** command replaces only the first occurrence of the word `happy` on a line.

   The **sed** command operates as a filter. It reads text from standard input or from the files named on the command line (`chap1` in this example), modifies this text, and writes it to standard output. Unlike most editors, it does not replace the original file. This makes the **sed** command a powerful command when used in pipelines.

2. To use the **sed** command as a filter in a pipeline, enter:

   ```
   pr  chap2 | sed "s/Page *[0-9]*$/(&)/" | enq
   ```

   This command sequence encloses the page numbers in parentheses before printing the file `chap2`. The **pr** command puts a heading and page number at the top of each page, then the **sed** command puts the page numbers in parentheses, and the **enq** command prints the edited listing.

   The **sed** command pattern `/Page *[0-9]*$/` matches page numbers that appear at the end of a line. The **s** subcommand changes this to `(&)`, where the & stands for the page number that was matched.

3. To display selected lines of a file, enter:

   ```
   sed  -n  "/food/p" chap3
   ```

   The `sed -n` displays each line in the file `chap3` that contains the word `food`. Normally, the **sed** command copies every line to standard output after it is edited. The **-n** flag stops the **sed** command from doing this. You then use subcommands like **p** to write specific parts of the text. Without the **-n** flag, this example displays all the lines in the file `chap3`, and it shows each line containing `food` twice.

4. To perform complex editing, enter:

   ```
   sed  -f  script.sed  chap4  >chap4.new
   ```

   This command sequence creates a **sed** script file when you want to do anything complex. You can then test and modify your script before using it. You can also reuse your script to edit other files. Create the script file with an interactive text editor.

5. A sample **sed** script file:

   ```
   :join
   /\\$/{N
   s/\\\n//
   b join
   }
   ```

   This **sed** script joins each line that ends with a \ (backslash) to the line that follows it. First, the

pattern `/\\$/` selects a line that ends with a \ for the group of commands enclosed in `{}` (braces). The **N** subcommand then appends the next line, embedding a new−line character. The `s/\\\n//` deletes the \ and embedded new−line character. Finally, `b join` branches back to the label `:join` to check for a \ at the end of the newly joined line. Without the branch, the **sed** command writes the joined line and reads the next one before checking for a second \.

> **Note:** The **N** subcommand causes the **sed** command to stop immediately if there are no more lines of input (that is, if the **N** subcommand reads an end−of−file character). It does not copy the pattern space to standard output before stopping. This means that if the last line of the input ends with a \, it is not copied to the output.

6. To copy an existing file (`oldfile`) to a new file (`newfile`) and replace all occurrences of the `testpattern` text string with the contents of the `$REPL` shell variable, enter:

```
cat oldfile | sed −e "s/testpattern/$REPL/g" > newfile
```

Related Information

The **awk** command, **ed** command, **grep** command.

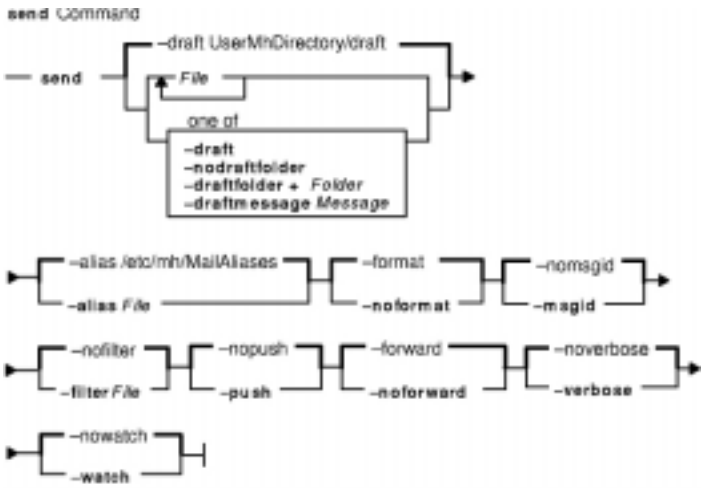
Manipulating Strings with sed, National Language Support Overview for Programmers in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

send Command

Purpose

Sends a message.

Syntax



send [*File* ... | { **−draft** | **−nodraftfolder** | **−draftfolder** +*Folder* | **−draftmessage** *Message* }] [**−alias** *File*] [**−format** | **−noformat**] [**−nomsgid** | **−msgid**] [**−nofilter** | **−filter** *File*] [**−nopush** | **−push**] [**−forward** | **−noforward**] [**−noverbose** | **−verbose**] [**−nowatch** | **−watch**]

Description

The **send** command routes messages through the mail delivery system. If the delivery fails, the **send** command displays an error message. By default, `From:` and `Date:` fields are added to each specified message. Unless a **$SIGNATURE** environment variable or `signature:` profile entry exists, the **send** command places the sender's address in the `From:` field.

The **send** command puts the current date in the `Date:` field. If the **dist** command calls the **send** command, the **send** command adds `Resent−` to the `From:`, `Date:`, and `Message−ID:` fields.

After successful delivery, the **send** command removes messages from active status by renaming them. The system renames messages by prefacing the current message number with a , (comma). Inactive files are unavailable to the Message Handler (MH) package. However, system commands can still manipulate inactive files. Until you use the **send** command again, you can retrieve an inactive file.

Flags

−alias *File*

Specifies a mail alias file to be searched. Three MH profile entries are required to use MH aliases:

```
ali: −alias Aliases
```

```
send: −alias Aliases
```

```
whom: -alias Aliases
```

where `Aliases` is the file to be searched. The default alias file is **/etc/mh/MailAliases**.

−draft Uses the current draft message if no file is specified. Without this flag and when no file is specified, the **send** command asks the user if the current draft message is the one to use.

−draftfolder +*Folder* Specifies the draft folder that contains the draft message to be sent. The **−draftfolder** +*Folder* flag followed by a *Message* parameter is the same as specifying the **−draftmessage** flag.

−draftmessage*Message* Specifies the message to be sent. You can use one of the following message references as the value of the *Message* parameter:

Number	Number of the message.
cur or **.** (period)	Current message. This is the default.
first	First message in a folder.
last	Last message in a folder.
next	Message following the current message.
prev	Message preceding the current message.

−filter *File* Uses the format instructions in the specified file to reformat copies of the message sent to the recipients listed in the `Bcc:` field.

−format Puts all recipient addresses in a standard format for the delivery transport system. This flag is the default.

−forward Adds a failure message to the draft message and returns it to the sender if the **send** command fails to deliver the draft. This flag is the default.

−help Lists the command syntax, available switches (toggles), and version information.

Note: For MH, the name of this flag must be fully spelled out.

−msgid Adds a message−identification component (such as `Message−ID:`) to the message.

−nodraftfolder Undoes the last occurrence of the **−draftfolder** +*Folder* flag. This flag is the default.

−nofilter Removes the `Bcc:` header field from the message for recipients listed in the `To:` and `cc:` fields. The flag then sends the message with minimal headers to recipients listed in the `Bcc:` field. This flag is the default.

−noformat Prevents alteration of the format of the recipient addresses.

−noforward Prevents return of the draft message to the sender if delivery fails.

−nomsgid Prevents addition of a message−identification component. This flag is the default.

−nopush Runs the **send** command in the foreground. This flag is the default.

−noverbose Prevents display of information during the delivery of the message to the **sendmail** command. This flag is the default.

−nowatch Prevents display information during delivery by the **sendmail** command. This flag is the default.

−push Runs the **send** command in the background. The **send** command does not display error messages on the terminal if delivery fails. Use the **−forward** flag to return messages to you that are not delivered.

−verbose Displays information during the delivery of the message to the **sendmail** command. This information allows you to monitor the steps involved in sending mail.

−watch Displays information during the delivery of the message by the **sendmail** command. This information allows you to monitor the steps involved in sending mail.

Profile Entries

The following entries are entered in the *UserMhDirectory*/**.mh_profile** file:

`Draft-Folder:`	Sets the default folder for drafts.
`mailproc:`	Specifies the program used to post failure notices.
`Path:`	Specifies the user's MH directory.
`postproc:`	Specifies the program used to post messages.
`Signature:`	Sets the mail signature.

Examples

To send a draft message that is in your **$HOME/Mail/draft** file, enter:

```
send
```

The system responds with a message similar to the following:

```
Use "/home/david/Mail/draft"?
```

If you enter `yes`, the draft message is sent, and you are returned to the shell prompt. In this example, the name of the **$HOME** directory is **/home/david**.

Files

$HOME/.mh_profile	Specifies the MH user profile.
/usr/bin/send	Contains the **send** command.

Related Information

The **ali** command, **comp** command, **dist** command, **forw** command, **post** command, **sendmail** command, **spost** command.

The **.mh_alias** file format, **.mh_profile** file format.

Mail Overview in *AIX Version 4.3 System User's Guide: Communications and Networks*.

Peek, Jerry. *MH and xmh: E−mail for Users and Programmers.* Sebastopol, CA: O'Reilly & Associates, 1992.

sendbug Command

Purpose

Mails a system bug report to a specified address.

Syntax

sendbug [*Address*]

Description

The **sendbug** command is a shell script to assist the user in composing and mailing bug reports in the correct format.

The **sendbug** command starts the editor specified by the **EDITOR** environment variable on a temporary copy of the bug report format outline. The default editor is vi.

Fill out the appropriate fields in the bug report format outline and exit the editor. The **sendbug** command mails the completed report to the address specified by the *Address* parameter. The default address is POSTMASTER.

Files

/usr/lib/bugformat Contains the bug report outline.

Related Information

The **bugfiler** command, **env** command, **sendmail** command.

Mail Overview for System Management in *AIX Version 4.3 System Management Guide: Communications and Networks*.

sendmail Command

Purpose

Routes mail for local or network delivery.

Syntax

sendmail [**−ba** | **−bd** | **−bD** | **−bh** | **−bH** | **−bi** | **−bm** | **−bp** | **−bs** | **−bv** | **−bt** [**−C***File*] [**−d***Value*]]
[**−B***Type*] [**−F***FullName*] [**−f***Name*] [**−h***Number*] [**−i**] [**−M***x Value*] [**−n**] [**−N***Dsn*]
[**−O***Option***=***Value*] [**−o***Option* [*Value*]] [**−p***Protocol*] [**−q** [*Time*]] [**−qI***Substr*]
[**−qR***Substr*] [**−qS***Substr*] [**−R***Return*] [**−r***addr*] [**−t**] [**−U**] [**−V***Envid*]] [**−v**]
[**−X***LogFile*] *Address*

> **Note:** The *Address* parameter is optional with the **−bd**, **−bi**, **−bp**, **−bt**, and **−q** [*Time*] flags.

Description

> **Note:** On sendmail V8.7 (AIX 4.2 and later), name resolution ordering is DNS, NIS, then local. If you wish to override this specify an **/etc/netsvc.conf** file or NSORDER environment variable.

The **sendmail** command receives formatted text messages and routes the messages to one or more users. Used on a network, the **sendmail** command translates the format of the header information of the message to match the requirements of the destination system. The program determines the network of the destination system by using the syntax and content of the addresses.

The **sendmail** command can deliver messages to:

- Users on the local system
- Users connected to the local system using the TCP/IP protocol
- Users connected to the local system using the Basic Networking Utilities (BNU) command protocol

The **sendmail** command is not intended as a user interface routine; other commands provide user−friendly interfaces. Use the **sendmail** command only to deliver preformatted messages.

The **sendmail** command reads standard input for message text. The **sendmail** command sends a copy of the message to all addresses listed whenever it reads an end of the message character, either an end−of−file (Ctrl−D) control sequence or a single period on a line.

Using the Configuration File

The **sendmail** command uses a configuration file (the **/etc/sendmail.cf** file by default) to set operational parameters and to determine how the command parses addresses. This file is a text file that you can edit with other text editors. After modifying **sendmail.cf**, refresh the **sendmail** daemon.

The current process ID of the **sendmail** command is stored in the **/etc/sendmail.pid** file. Issue the **kill −1** command as follows to have the **sendmail** command reread the newly edited **sendmail.cf**:

```
kill −1 `cat /etc/sendmail.pid`
```

If the **srcmstr** command is running, you may issue the **refresh** command, as follows, to build the configuration database, the aliases database, and the NLS database again.

```
refresh -s sendmail
```

The **sendmail** command rereads these databases and continues operation with the new data.

Defining Aliases

The **sendmail** command allows you to define aliases to use when the **sendmail** command handles the local mail. Aliases are alternate names that you can use in place of elaborate network addresses. You can also use aliases to build distribution lists.

Define aliases in the **/etc/aliases** file. This file is a text file you can edit. The **sendmail** command uses a database version of this file. You must build a new alias database by running the **sendmail−bi** command or the **newaliases** command before any changes made to the **/etc/aliases** file become effective.

> **Note:** When defining aliases in the **/etc/aliases** file, use only lowercase characters for nested aliases. Uppercase characters on the right−hand side of an alias are converted to lowercase before being stored in the Database Manager (DBM) database. In the following example, mail sent to `testalias` fails, because `TEST` is converted to `test` when the second line is stored.

```
TEST: user@machine
testalias: TEST
```

Every system must have a user or user alias designated as the **postmaster** alias. The default **postmaster** alias is a root file. You can assign this alias to a different user in the **/etc/aliases** file. The **postmaster** alias allows other users outside your system to send mail to a known ID and to get information about mailing to users on your system. Also, users on your system can send problem notifications to the **postmaster** ID.

Flags

−B*Type*	Indicates body type. **Note:** The **−b** flags are mutually exclusive.
−ba	Starts the **sendmail** command in ARPANET mode. All input lines to the command must end with a carriage return and a line feed (CR−LF). The **sendmail** command generates messages with a CR−LF at the end and looks at the `From:` and `Sender:` fields to find the name of the sender.
−bd	Starts the **sendmail** command as a daemon running in the background as a Simple Mail Transfer Protocol (SMTP) mail router.
−bD	Starts the **sendmail** command as a daemon running in the foreground as a Simple Mail Transfer Protocol (SMTP) mail router.
−bh	Prints the persistent host status database.
−bH	Purges the persistent host status database.
−bi	Builds the alias database from information defined in the **/etc/aliases** file. Running the **sendmail** command with this flag is the same as running the **/usr/sbin/newaliases** command.
−bm	Delivers mail in the usual way. (This is the default.)
−bp	Prints a listing of the mail queue. Running the **sendmail** command with this flag is the same as running the **/usr/sbin/mailq** command.
−bs	Uses the simple mail transfer protocol (SMTP) as described in RFC821 to collect mail from standard input. This flag also includes all of the operations of the **−ba** flag that are

compatible with SMTP.

−bt Starts the **sendmail** command in address test mode. This mode allows you to enter interactive addresses and watch as the **sendmail** command displays the steps it takes to parse the address. At the test−mode prompt, enter a rule set or multiple rule sets separated by commas and an address. Use this mode for debugging the address parsing rules in a new configuration file.

−bv Starts the **sendmail** command with a request to verify the user IDs provided in the *Address* parameter field of the command. The **sendmail** command responds with a message telling which IDs can be resolved to a mailer command. It does not try to collect or deliver a message. Use this mode to validate the format of user IDs, aliases, or mailing lists.

−C*File* Starts the **sendmail** command using an alternate configuration file specified by the *File* variable. Use this flag together with **−bt** to test a new configuration file before installing it as the running configuration file.

−d*Value* Sets the debugging value to the value specified by the *Value* variable. The only valid value is 21.*n*, where *n* is any nonzero integer. This produces information regarding address parsing and is typically used with the **−bt** flag. Higher values of *n* produce more verbose information.

−F*FullName* Sets the full name of the sender to the string provided in the *FullName* variable.

−f*Name* Sets the name of the sender of the mail. Only administrative user IDs designated in the configuration file with the **T** control line, or user IDs supplied by the *Name* variable can use this flag.

−h*Number* Sets the hop count to the value specified by the *Number* variable. The hop count is the number of times that the message has been processed by an SMTP router (not just the local copy of the **sendmail** command). The mail router increments the hop count every time the message is processed. When it reaches a limit, the message is returned with an error message in order to prevent infinite loops in the mail system.

−i Ignores dots alone on lines by themselves in incoming messages. This should be set if you are reading data from a file.

−M*x Value* Sets marco *x* to the specified *value*.

−N *Dsn* Sets delivery status notification conditions to DSN. The delivery status notification conditions can be: `never` for no notifications or for a comma separated list of the values, `failure` for notification if delivery failed, `delay` for notification if delivery is delayed, and `success` for notification when the message is successfully delivered.

−n Prevents the **sendmail** command from interpreting aliases.

−O *Option*=*Value* Sets *Option* to the specified *Value*. Use for long−form option names. This flag applies only to AIX Version 4.2 or later.

−o*Option* [*Value*] Sets the *Option* variable. If the option is a valued option, you must also specify a value for the *Value* variable.

> **Note:** For valid values, see "Options for the sendmail Command in the sendmail.cf File" in *AIX Version 4.3 System Management Guide: Communications and Networks*.

−p*Protocol* Sets the sending protocol. It is recommended that you set this. You can set *Protocol* in the form *Protocol***:***Host* to set both the sending protocol and the sending host. For example, `−pUUCP:uunet` sets the sending protocol to UUCP and the sending host to uunet. Some existing programs use **−oM** flag to set the **r** and **s** macros, which is equivalent to using the **−p** flag. This flag applies only to 4.2 or later.

−qI*Substr* Limits process jobs to those containing *Substr* as a substring of the queue ID.

−qR*Substr* Limits process jobs to those containing *Substr* as a substring of one of the recipients.

−qS*Substr* Limits process jobs to those containing *Substr* as a substring of the sender.

−q [*Time*] Processes saved messages in the queue at the intervals specified by the *Time* variable. If the *Time* variable is not specified, this flag processes the queue at once.

−R*Return*	Sets the amount of the message to be returned if the message bounces. The *Return* parameter can be `full` to return the entire message or `hdrs` to return only the headers.
−r*addr*	An obsolete form of **−f**.
−t	Sends the message to the recipients specified in the `To:`, `Cc:`, and `Bcc:` fields of the message header, as well as to any users specified on the command line.
−U	Sets initial (user) submission. This should always be set when called from a user agent such as Mail or exmh and should never be set when called by a network delivery agent such as rmail.
−V*Envid*	Sets the original envelope ID. This is propagated across SMTP to servers that support DSNs and is returned in DSN−compliant error messages.
−v	Starts the **sendmail** command in verbose mode. The **sendmail** command displays messages regarding the status of transmission and the expansion of aliases.
−X *LogFile*	Logs all traffic in and out of **sendmail** in *LogFile* for debugging mailer problems. Use this flag sparingly, since it produces a lot of data very quickly. This flag applies only to 4.2 or later.

You can also set or remove the **sendmail** configuration processing options. The person responsible for the mail system uses these options. To set these options, use the **−o** flag on the command line or the **O** control line in the configuration (**/etc/sendmail.cf**) file.

Security

Auditing Events:

Event	Information
SENDMAIL_Config	Configuration event
SENDMAIL_ToFile	File−creation event

Exit Status

The **sendmail** command returns exit status values. These exit values are defined in the **/usr/include/sysexits.h** file. The following table summarizes the meanings of these return values:

EX_CANTCREAT	The **sendmail** command cannot create a file that the user specified.
EX_CONFIG	An error was found in the format of the configuration file.
EX_DB	An error occurred while trying to gain access to a database.
EX_IOERR	An error occurred during I/O.
EX_NOHOST	The **sendmail** command could not recognize the specified host name.
EX_NOPERM	The user does not have permission to perform the requested operation.
EX_NOUSER	The **sendmail** command could not recognize a specified user ID.
EX_OK	The **sendmail** command successfully completed.
EX_OSERR	A temporary operating system error occurred. An example of such an error is a failure to create a new process.
EX_OSFILE	A system file error occurred. For example, a system file (such as **/etc/passwd**) does not exist, cannot be opened, or has another type of error preventing it from being used.
EX_PROTOCOL	The remote system returned something that was incorrect during a protocol exchange.
EX_SOFTWARE	An internal software error occurred (including bad arguments).
EX_TEMPFAIL	The **sendmail** command could not create a connection to a remote system. Try the

	request again later.
EX_UNAVAILABLE	A service or resource that the **sendmail** command needed was not available.
EX_USAGE	The command syntax was not correct.
EX_NOLHOST	The local host name is not defined.

Files

/usr/sbin/sendmail	Contains the **sendmail** command.
/usr/sbin/mailq	Contains the mail queue.
/usr/sbin/newaliases	Contains the alias database.
/usr/sbin/mailstats	Contains statistics found in the **/usr/lib/sendmail.st** file.
/etc/aliases	Contains the text version of the **sendmail** command aliases.
/etc/aliases.dir	Contains DBM formatted database for aliases.
/etc/aliases.pag	Contains DBM formatted database for aliases.
/etc/sendmail.cf	Contains the text version of the **sendmail** configuration file.
/etc/sendmail.st	Contains mail routing statistics information.
/usr/lib/smdemon.cleanu	Maintains aging copies of the log file found in the **/var/spool/mqueue** directory.
/var/spool/mqueue	Contains the temporary files and the log file associated with the messages in the mail queue.
/usr/bin/uux	Contains the mailer command to deliver Basic Networking Utilities (BNU) mail.
/usr/bin/bellmail	Contains the mailer command to deliver local mail.

Related Information

The **bellmail** command, **kill** command, **mail, Mail** command, **mailq** command, **mailstats** command, **newaliases** command, **refresh** command, **uux** command.

The **srcmstr** daemon.

BNU Overview, Chapter 2. Mail, and Chapter 3. Transmission Control Protocol/Internet Protocol (TCP/IP) Overview in *AIX Version 4.3 System Management Guide: Communications and Networks*.

Chapter 2. Mail Overview in *AIX Version 4.3 System User's Guide: Communications and Networks*.

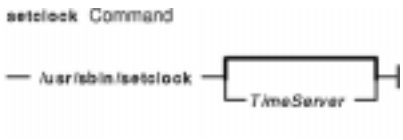
Peek, Jerry. *MH and xmh: E−mail for Users and Programmers.* Sebastopol, CA: O'Reilly & Associates, 1992.

setclock Command

Purpose

Sets the time and date for a host on a network.

Syntax



/usr/sbin/setclock [*TimeServer*]

Description

The **/usr/sbin/setclock** command gets the time from a network time server, and if run by a user with root user authority, sets the local time and date accordingly.

The **setclock** command takes the first response from the time server, converts the calendar clock reading found there, and displays the local date and time. If the **setclock** command is run by the root user, it calls the standard workstation entry points to set the system date and time.

If no time server responds or if the network is not operational, the **setclock** command displays a message to that effect and leaves the current date and time settings of the system unchanged.

> **Note:** Any host running the **inetd** daemon can act as a time server.

Parameter

TimeServer The host name or address of a network host that services TIME requests. The **setclock** command sends an Internet TIME service request to a time server host. If the *TimeServer* name is omitted, the **setclock** command sends the request to the default time server. The default time server in a DOMAIN environment is specified by the name server. Otherwise the default time server is specified in the **/etc/hosts** file.

Examples

1. To display the date and time using the time server host specified in the **/etc/hosts** file, enter:

   ```
   setclock
   Sat Mar 11 15:31:05 1988
   ```

   The **setclock** command displays the proper date and time.

2. To set the date and time, enter:

   ```
   su root
   setclock host1
   Thu Jan 12 15:24:15 1990
   ```

You must use the **su** command or log in as the root user before setting the time from the time server in

`host1`.

Related Information

The **timedc** command.

The **inetd** daemon, **timed** daemon.

The **hosts** file format.

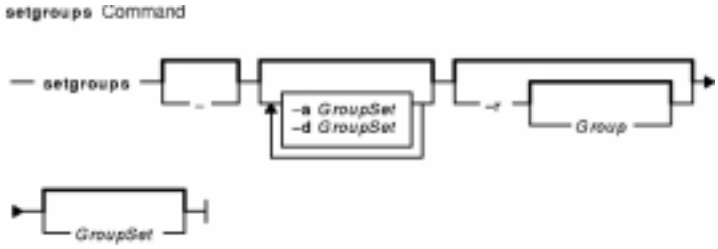
TCP/IP Daemons in *AIX Version 4.3 System Management Guide: Communications and Networks*.

setgroups Command

Purpose

Resets a session's process group set.

Syntax



setgroups [−] [−**a***GroupSet*] [−**d** *GroupSet*] [−**r**[*Group*]] [*GroupSet*]

Description

The **setgroups** command, by default, displays the user's current group set and process group set for the current shell. A user's group set is defined in the user database files. When given a flag and a *GroupSet* parameter, this command resets the process group set as listed by the *GroupSet* parameter. The *GroupSet* parameter is a comma−separated list of group names. The available groups are defined in the user database files.

You can also use the **setgroups** command to add or delete groups from the current group set. Using the **−r** flag, you can reset the real group ID. If you specify the *Groupset* parameter but no flags, the **setgroups** command resets all the groups and makes the first group in the list the real group. The **setgroups** command does not change the security characteristics of the controlling terminal.

When you run the **setgroups** command, the system always replaces your shell with a new one. The command replaces your shell regardless of whether the command is successful or not. For this reason, the command does not return error codes.

The **setgroups −r** command is identical to the **newgrp** command.

Flags

−a *GroupSet*	Adds the groups specified by the *GroupSet* parameter to the current session. The number of groups in the new set must not exceed **NGROUPS_MAX** groups, a value defined in the **limits.h** file. The real group ID is not changed.
−d *GroupSet*	Removes the groups specified by the *GroupSet* parameter from the current session. If the real group is removed, the next group listed in the current set becomes the real group.
−r *Group*	Resets the real group for the current process. If you do not specify a *Group* parameter and the current real group is not the primary group, the **−r** flag removes the current real group and resets the real group to the original primary group. If you specify a *Group* parameter, this behaves identically to the **newgrp** command.
−	Re−initializes the group set of the session to its original login state.

Security

Access Control: This command should be a general user program. This command should be installed as a program in the trusted computing base (TCB). The command should be owned by the root user with the **setuid** (SUID) bit set.

Files Accessed:

Mode	Files
r	**/etc/passwd**
r	**/etc/group**

Auditing Events:

Event	Information
USER_SetGroups	realgroup, groupset

Examples

1. As user `sah`, you can display your current group membership and process group set, by entering:

   ```
   setgroups
   ```

   Output similar to the following appears:

   ```
   sah:

      user groups = staff,payroll
      process groups = staff,payroll
   ```

2. To add the `finance` group to the process group of the current session, enter:

   ```
   setgroups −a  finance
   ```

3. To set your real group to `finance`, enter:

   ```
   setgroups finance,staff,payroll
   ```

   This sets `finance` as the real group. The `staff` and `payroll` groups make up the supplementary group list.

4. To delete the `payroll` group from the current process group set, enter:

   ```
   setgroups −d  payroll
   ```

5. To change the process group set back to your default set, enter:

   ```
   setgroups  −
   ```

This resets your current session to its original state just after you log in.

Files

/usr/bin/setgroups	Contains the **setgroups** command.
/etc/group	Contains basic group attributes.
/etc/passwd	Contains basic user attributes.

Related Information

The **login** command, **newgrp** command, **setsenv** command, **tsm** command.

setmaps Command

Purpose

Sets terminal maps or code set maps.

Syntax

Uses setmaps with no Input or Output Map File Designation

setmaps [**−v**] [**−c** | **−h**]

Selects a File from Default Directory as the Code Set Map File

setmaps [**−v**] **−s** **−i***MapName*

Selects a Designated File as the Code Set Map File

setmaps [**−v**] **−s** **−I***File1*

Selects a File from Default Directory as the Input or Output Terminal Map File

setmaps [**−v**] [**−D**] [**−k** *KeyName*] [**−d** *DirectoryPath*] { **−i** | **−o** } *MapName*

Selects Files from Default Directory as the Input or Output Terminal Map Files

setmaps [**−v**] [**−D**] [**−d** *DirectoryPath*] **−t***MapName*

Selects a Designated File as the Input or Output Terminal Map File

setmaps [**−v**] [**−D**] [**−k** KeyName] { **−I** | **−O** } *File1*

Loads the Default Terminal Map File for Later Use

setmaps [**−v**] [**−D**] [**−k** KeyName] [**−r**] **−l***File2*

Loads a Designated Terminal Map File for Later Use

setmaps [**−v**] [**−D**] [**−k** KeyName] [**−r**] **−L***File1*

Description

> **Note:** If this command is run without root user authority, the code set map is not loaded, only debugged.

The **setmaps** command handles terminal and code set maps. The **−s** flag must be used for code set maps. The operating system uses input and output terminal maps to convert internal data representations to the ASCII characters supported by asynchronous terminals. If you enter the **setmaps** command with no flags, it displays the names of the current input and output terminal maps.

A terminal map is a text file containing a list of rules that associate a pattern string with a replacement string. This file normally resides in the **/usr/lib/nls/termmap** directory. The operating system uses an input map file

to map input from the keyboard to an application and an output map file to map output from an application to the display.

Terminal mapping works as follows:

1. The system collects characters in a buffer until a pattern specified by a rule in the map file matches a substring in the buffer.
2. The system then constructs and returns the replacement string specified by the rule.

This processing continues with the remaining characters in the buffer.

The rules of a terminal map can test and change the state of the pattern processor. The state is identified by a single−byte character, conventionally a digit (0 through 9). The state is reset to 0, the initial state, whenever the system loads a new map or flushes the terminal input or output buffer (such as when it processes a KILL or INTR character or when a program issues an **ioctl** system call). A terminal map can use states to detect multibyte escape sequences, among other tasks. You can test for state *x* by specifying @x in a pattern. You can set the state to *x* by including @x in the replacement string.

The **setmaps** command, when using the **−s** flag, assigns a code set map to the standard input device. The operating system uses code set maps to determine the number of bytes of memory a character requires and the number of display columns it requires.

Flags

−c	Clears all mappings on this terminal.
−d*DirectoryPath*	Causes the *DirectoryPath* variable to be used as the path to the directory that contains the *MapName* variable. Specifying this flag and variable overrides the **/usr/lib/nls/termmap** directory.
−D	Produces a debug program printout of the specified map on the standard output device before loading the map. When using this to run the debug program on new maps, do not run with root user authority until the map is fully debugged to prevent the map from actually being loaded.
−h	Prints the usage information of the **setmaps** command (used with the **−v** flag for advanced users).
−i*MapName*	Selects the **/usr/lib/nls/termmap/MapName.in** file as the input map. When used with the **−s** flag, this flag selects the **/usr/lib/nls/csmap/MapName** file as the terminal code set map file.
−I*File1*	Selects the contents of the *File1* variable as the input map. The file specified by the *File1* variable can be either a full path name or a path name relative to the current working directory. When used with the **−s** flag, this flag selects the contents of the *File1* variable as the terminal code page map file.
−k*KeyName*	Associates the contents of the *KeyName* variable with the map being selected. This key name overrides the default key, which is normally set to the value of the *MapName* variable.
−l*File2*	Loads the **/usr/lib/nls/termmap/File2** file for later use. The *File2* variable includes the full path name and suffix (if any) of the map file. **Note:** You must have root user authority to specify this flag.
−L*File1*	Loads the specified map for later use. The *File1* variable includes the full path name and suffix (if any) of the map file. **Note:** You must have root user authority to specify this flag.
−o*MapName*	Selects the **/usr/lib/nls/termmap/MapName.out** file as the terminal output map.
−O*File1*	Selects the contents of the *File1* variable as the terminal output map. The *File1* variable includes the full path name and suffix (if any) of the map file.

−r	Forces reloading of the specified map, even if it is already loaded. Terminals using the old map continue to do so until they are logged off or until their maps are explicitly reset. If you do not specify this flag, a map is loaded only if it has not already been loaded into the kernel. **Note:** You must have root user authority to specify this flag.
−s	Treats any map as a code set map.
−t*MapName*	Selects the **/usr/lib/nls/termmap/MapName.in** file as the terminal input map and the **/usr/lib/nls/termmap/MapName.out** file as the terminal output map.
−v	Selects verbose output.

All maps loaded must have unique names. Use the **−k** flag to eliminate naming conflicts. Only the **−i**, **−o**, and **−t** flags implicitly add a suffix. Other flags specifying map names should include a suffix if appropriate. If a requested map name is already loaded in the kernel, that map is used even if the path information provided on the command line implies a different map.

To reset the code set map to its original state, the **/usr/lib/nls/csmap/sbcs** code set map should be used.

Examples

1. To display the current map settings for this terminal, enter:

   ```
   setmaps
   ```

2. To clear all mapping for the current terminal, enter:

   ```
   setmaps −c
   ```

3. To set up mapping (both input and output maps) for an `ibm3161−C` terminal, enter:

   ```
   setmaps −t ibm3161−C
   ```

4. To load the `vt220` input map into the kernel as the `fred` map, enter:

   ```
   setmaps −k fred −i vt220
   ```

5. To gather debug output for a new map called `bob` in a file called `bob.dump`, enter:

   ```
   setmaps −D −L /tmp/bob > bob.dump
   ```

6. To set up a code set map conforming to the IBM−932 code page for this terminal, enter:

   ```
   setmaps −s −i IBM−932
   ```

7. To set up a code set map conforming to the IBM−943 code page for this terminal, enter:

   ```
   setmaps −s −i IBM−943
   ```

8. To set up a code set map from the file `myEUC` for this terminal, enter:

   ```
   setmaps −s −I myEUC
   ```

Files

/usr/bin/setmaps	Contains the **setmaps** command.
/usr/lib/nls/termmap/*.in	Contains input map files.
/usr/lib/nls/termmap/*.out	Contains output map files.
/usr/lib/nls/csmap/sbcs	Contains code set map for a single−byte code page.
/usr/lib/nls/csmap/IBM−943	Contains code set map for the IBM−943 code page.

/usr/lib/nls/csmap/IBM−eucJP Contains code set map for the IBM−eucJP code page.

Related Information

The **stty** command.

The **setmaps** file format, **termios.h** file.

The **setcsmap** subroutine.

National Language Support Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*

setsenv Command

Purpose

Resets the protected state environment of a user.

Syntax

setsenv [−] *NewEnvironment*

Description

The **setsenv** command resets your protected state environment while you are logged in. The protected state environment is defined as a set of variables. These variables are kept in the kernel and can be modified only by a **SETUINFO** system call. The **setsenv** command uses the variables specified by the *NewEnvironment* parameter. This parameter consists of *EnvironmentVariable*=*Value* definitions separated by a blank space. For information on environment variables, see **environment** File.

You cannot reset the following environment variables with the **setsenv** command:

NAME	Your last authenticated user name. This corresponds to the real user ID of the current process.
TTY	The name of the terminal on which you logged in. This corresponds to the initial controlling terminal for the process. This variable cannot be set for processes initiated without a *full login*. A full login is a login initiated by the **getty** command.
LOGNAME	The name under which you logged in, if the current session was started from a terminal login program. If the session was not started from a terminal, this variable is not set.

If you enter the **setsenv** command without any defined variables, it displays the current protected state. The **setsenv** command does not change the security characteristics of the controlling terminal.

When you run the **setsenv** command, it replaces your current shell and gives you a new one. The command replaces your shell regardless of whether it completed successfully or not. For this reason, the command does not return error codes.

Flags

− Reinitializes the environment as if the user had just logged in to the system. Otherwise, the environment is not changed.

Security

Access Control: This command should be a standard user program. This command should be installed as a program in the trusted computing base (TCB). The command should be owned by the root user with the **setuid** (SUID) bit set.

Files Accessed:

Mode	File
r	**/etc/environment**
r	**/etc/security/environ**

Auditing Events:

Event	Information
USER_SetEnv	new environment string

Examples

1. To display the current environment variables, enter:

   ```
   setsenv
   ```

2. To add the `PSEUDO=tom` protected environment variable, enter:

   ```
   setsenv PSEUDO=tom
   ```

This example sets a user name for the **PSEUDO** protected environment variable.

Files

/usr/bin/setsenv	Specifies the path to the **setsenv** command.
/etc/environment	Contains environment information for each user.
/etc/security/environ	Contains privileged environment information for each user.

Related Information

The **login** command, **setgroups** command, **su** command, and**tsm** command.

The **getuinfo** subroutine, **setpenv** subroutine, **usrinfo** subroutine.

For more information about the identification and authentication of users, discretionary access control, the trusted computing base, and auditing, refer to Security Administration in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

sh Command

Purpose

Invokes the default shell.

Syntax

Refer to the syntax of the **ksh** command. The **/usr/bin/sh** file is linked to the Korn shell.

Description

The **sh** command invokes the default shell and uses its syntax and flags. The shell linked to the **/usr/bin/sh** path is the default shell. The standard configuration of the operating system links the **/usr/bin/sh** path to the Korn shell. See "Korn Shell" in *AIX Version 4.3 System User's Guide: Operating System and Devices* for specific information about Korn shell features.

Flags

Refer to the flags for the Korn shell (**ksh** command).

Files

/usr/bin/sh Contains the **sh** command.

Related Information

The **ksh** command.

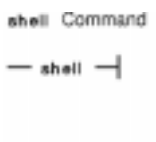
Korn Shell, Shells Overviewin *AIX Version 4.3 System User's Guide: Operating System and Devices*.

shell Command

Purpose

Executes a shell with the user's default credentials and environment.

Syntax



shell

Description

The **shell** command re−initializes a user's login session. When the command is given, the port characteristics of the process's controlling terminal are reset and all access to the port is revoked. The **shell** command then resets the process credentials and environment to the defaults established for the user and executes the user's initial program. All credentials and environment are established according to the login user ID of the invoking process.

If the **shell** command is invoked on the trusted path and the user's **tpath** attribute in the **/etc/security/user** file does not have a value of **always**, the trusted environment of the terminal is not maintained.

> **Note:** The **shell** command does not reset the login ID of the user.

Security

Access Control: The command should be **setuid** to the root user to reset the user's process credentials, and grant execute (x) access to all users. The command should have the **trusted computing base** attribute.

Files Accessed:

Mode	File
r	**/etc/passwd**
r	**/etc/group**
r	**/etc/security/audit/config**
r	**/etc/security/environ**
r	**/etc/security/limits**
r	**/etc/security/user**

Auditing Events:

Event	Information
USER_Shell	portname

Examples

To re−initialize your session to your default credentials and environment after using the trusted shell (**tsh**), enter:

```
shell
```

Files

/usr/bin/shell	Contains the **shell** command.
/etc/security/user	Contains the extended attributes of users.
/etc/passwd	Contains user IDs.
/etc/group	Contains group IDs.
/etc/security/audit/config	Contains the audit configuration information.
/etc/security/environ	Defines the environment attributes for users.
/etc/security/limits	Defines process resource limits for each user.

Related Information

The **getty** command, **init** command, **login** command, **logout** command, **setgroups** command, **su** command, **tsh** command, **tsm** command.

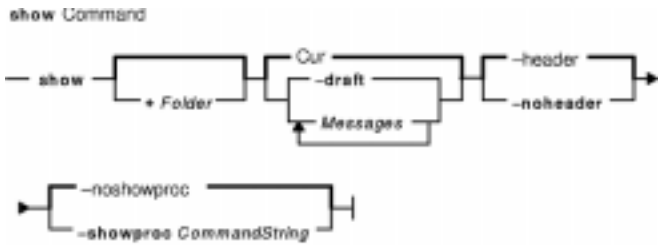
For more information about the identification and authentication of users, discretionary access control, the trusted computing base, and auditing, refer to Security Administration in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

show Command

Purpose

Shows messages.

Syntax



show [+*Folder*] [**−draft** | *Messages*] [**−header** | **−noheader**] [**−showproc** *CommandString* | **−noshowproc**]

Description

The **show** command displays the contents of messages. If standard output is not a display, the **show** command lists each message with a one−line header and two separation lines. By default, the **show** command displays the current message in the current folder.

The **show** command invokes a listing program to create the list. The default listing program is **/usr/bin/more**. You can define your own default with the `showproc:` entry in your **$HOME/.mh_profile** file. If you set the `showproc:` entry to `mhl`, the **show** command calls an internal **mhl** routine instead of the **mhl** command. You can also specify the program to perform a listing in the *CommandString* parameter of the **−showproc** flag.

The **show** command passes any flags it does not recognize to the listing program. Thus, you can specify flags for the listing program, as well as for the **show** command.

If the `Unseen−Sequence:` entry is present in your **$HOME/.mh_profile** file and the entry is not empty, the **show** command removes each of the messages shown from each sequence named by the profile entry. If several messages are specified, the last message shown becomes the current message.

Flags

−draft	Shows the *UserMhDirectory*/**draft** file if it exists.
+*Folder*	Specifies a folder. The current folder is the default.
−header	Displays a one−line description of the message being shown. The description includes the folder name and message number. If you show more than one message, this flag does not produce message headers. The **−header** flag is the default.
−help	Lists the command syntax, available switches (toggles), and version information. **Note:** For MH, the name of this flag must be fully spelled out.
Messages	Specifies the messages to show. You can specify several messages, a range of messages, or a single message. Use the following references to specify

messages:

Number	Number of the message.
Sequence	A group of messages specified by the user. Recognized values include:

all	All messages in a folder.
cur or **.** (period)	Current message. This is the default.
first	First message in a folder.
last	Last message in a folder.
next	Message following the current message.
prev	Message preceding the current message.

−noheader	Prevents display of a one−line description of each message.
−noshowproc	Uses the **/usr/bin/cat** command to perform the listing. This is the default.
−showproc *CommandString*	Uses the specified command string to perform the listing.

Profile Entries

The following entries are entered in the *UserMhDirectory*/**.mh_profile** file:

`Current−Folder:`	Sets the default current folder.
`Path:`	Specifies the user's MH directory.
`showproc:`	Specifies the program used to show messages.
`Unseen−Sequence:`	Specifies the sequences used to keep track of the unseen messages.

Examples

1. To display the contents of the current message in the current folder one screen at a time, enter:

   ```
   show
   ```

   If the message continues for more than one screen, press the Enter key until you have read the entire message.

2. To see the contents of all the messages in the current folder, enter:

   ```
   show all
   ```

   If the messages continue for more than one screen, press the Enter key until you have read all the messages.

3. To see the contents of message `5` in the `meetings` folder, enter:

   ```
   show +meetings 5
   ```

4. To see the contents of all the messages belonging to the `weekly` sequence in the `meeting` folder, enter:

   ```
   show +meeting weekly
   ```

Files

$HOME/.mh_profile	Specifies the MH user profile.
UserMhDirectory/**draft**	Contains the current message draft.
/usr/bin/show	Contains the **show** command.

Related Information

The **mhl** command, **next** command, **pick** command, **prev** command, **scan** command, **sendmail** command.

The **.mh_alias** file format, **.mh_profile** file format.

Mail Overview in *AIX Version 4.3 System User's Guide: Communications and Networks*.

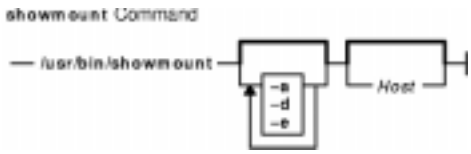
Peek, Jerry. *MH and xmh: E−mail for Users and Programmers.* Sebastopol, CA: O'Reilly & Associates, 1992.

showmount Command

Purpose

Displays a list of all clients that have remotely mounted file systems.

Syntax



/usr/bin/showmount [**−a**] [**−d**] [**−e**] [*Host*]

Description

The **showmount** command displays a list of all clients that have remotely mounted a file system from a specified machine in the *Host* parameter. This information is maintained by the **mountd** daemon on the *Host* parameter. This information is saved in the **/etc/rmtab** file in case the server crashes. The default value for the *Host* parameter is the value returned by the **hostname** command.

> **Note:** If a client crashes, its entry will not be removed from the list until the client reboots and starts the **umount−a** command.

Flags

−a Prints all remote mounts in the format *HostName***:***Directory*, in which *HostName* is the name of the client and *Directory* is a directory pathname that has been remotely mounted.

−d Lists only directories that have been remotely mounted by clients.

−e Prints the list of exported directories.

Examples

1. To display a list of all remote directories mounted by a host, enter:

   ```
   /usr/bin/showmount −a zeus
   ```

   In this example, the `showmount` command produces a list of all of the remote directories mounted by the clients on the host machine named `zeus`.

2. To display a list of only the directories mounted by a client on the host, enter:

   ```
   /usr/bin/showmount −d athena
   ```

   In this example, the `showmount` command produces a list of all remote directories mounted by the client machines on the host named `athena.`

3. To print a list of all directories exported from a machine, enter:

   ```
   /usr/bin/showmount −e zeus
   ```

In this example, the `showmount` command produces a list of all remote directories exported by the host

machine named `zeus`.

Files

/etc/rmtab Contains information about the current state of all exported directories.
/etc/xtab Lists currently exported directories.

Related Information

The **hostname** command, **umount** command.

The **mountd** daemon.

List of NFS Commands.

Network File System (NFS) Overview for System Management, NFS Problem Determination in *AIX Version 4.3 System Management Guide: Communications and Networks*.

shutacct Command

Purpose

Turns off processing accounting.

Syntax

/usr/sbin/acct/shutacct [**"***Reason***"**]

Description

The **shutacct** command turns off process accounting and calls the **acctwtmp** command to add a record stating the reason to the **/var/adm/wtmp** file. The **shutacct** command is invoked by the **shutdown** command.

> **Note:** It is necessary to place quotation marks around the *Reason* value in the **/var/adm/wtmp** file.

Variables

Reason Specifies the reason for accounting system shutdown. This value is optional.

Security

Access Control: This command should grant execute (x) access only to members of the adm group.

Files

/usr/sbin/acct The path to the accounting commands.
/var/adm/wtmp The login and logout history file.

Related Information

The **turnacct** command.

For more information about the Accounting System, the preparation of daily and monthly reports, and the accounting files, see the Accounting Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

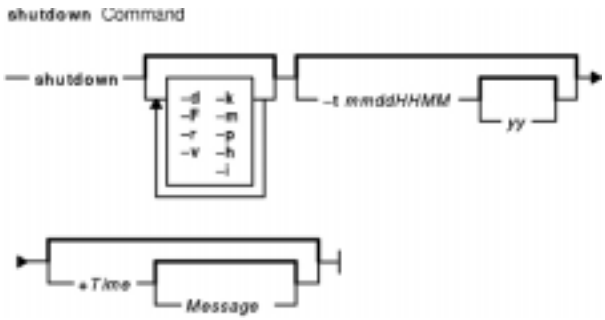
Setting Up an Accounting System in *AIX Version 4.3 System Management Guide: Operating System and Devices* explains the steps you must take to establish an accounting system.

shutdown Command

Purpose

Ends system operation.

Syntax



shutdown [**−d**] [**−F**] [**−h**] [**−i**] [**−k**] [**−m**] [**−p**] [**−r**] [**−t***mmddHHMM* [*yy*]] [**−v**] [**+***Time* [*Message*]]

Description

The **shutdown** command halts the operating system. Only a user with root user authority can run this command. During the default shutdown, users are notified (by a **wall** command) of the impending system shutdown with a message. However, shutdown is not complete until the user receives a shutdown completion message. Do not attempt to restart the system or turn off the system before the shutdown completion message is displayed; otherwise, file system damage can result.

> **Note:** The `halt completed` message is not displayed on the tty from which shutdown is invoked if it is connected to the system through a multiport adapter.

As shutdown time approaches, warning messages are displayed on the terminals of all users on the system.

After the specified number of seconds (60 by default), the system stops the accounting and error logging processes and writes an entry to the error log. The **shutdown** command then runs the **killall** command to end any remaining processes and runs the **sync** command to flush all memory resident disk blocks. Finally, it unmounts the file systems and calls the **halt** command.

> **Note:** Users who have files open on the node that is running the **shutdown** command, but who are not logged in to that node, are not notified about the shutdown.

If you request a complete halt to the operating system, the **shutdown** command stops all processes, unmounts all file systems, and calls the **halt** command.

The system administrator can place local customized shutdown procedures in a shell script named **/etc/rc.shutdown**. This script runs at the beginning of the shutdown if it exists. If the script runs but fails with a non−zero return code, the shutdown stops.

> **Attention:** If you are bringing the system down to maintenance mode, you must run the **shutdown** command from the **/** (root) directory to ensure that it can cleanly unmount the file systems.

Note: By default, if issued on models having a power supply capable of software control, the **shutdown** command powers down the system.

If the **−r** flag is specified, the **reboot** command is run. The **−m** flag brings the system down to maintenance (single user) mode, and the **−k** flag avoids shutting down the system. The **−i** flag runs an interactive shutdown procedure.

To shut down the system quickly, use **shutdown −F**. This flag bypasses messages to users and brings the system down as quickly as possible. Use this option only if there are no other users logged in.

To schedule the system to restart at a specified future date, use the **shutdown −t** flag.

Flags

−d Brings the system down from a distributed mode to a multiuser mode.

−F Does a fast shutdown, bypassing the messages to other users and bringing the system down as quickly as possible.

−h Halts the operating system completely; same as the **−v** flag.

−i Specifies interactive mode. Displays interactive messages to guide the user through the shutdown.

−k Avoids shutting down the system.

−m Brings the system down to maintenance (single user) mode.

−p Halts the system without a power down. This is used by uninterruptible power supply (UPS). This flag only applies to AIX Version 4.2 or later.

Note: The **−p** flag will have no effect if used in combination with flags not requiring a permanent halt. Power will still be turned off if other operands request a delayed poweron and reboot

−r Restarts the system after being shutdown with the **reboot** command.

−t *mmddHHMM* [*yy*] Restarts the system on the date specified by *mmddHHMM* [*yy*] where

mm Specifies the month.

dd Specifies the day.

HH Specifies the hour.

MM Specifies the minute.

yy Specifies the year.

The **shutdown −t** flag cannot be used with the **−v** or **−h** option.

Note: This option is only supported on systems that have a power supply which automatically turns power off at shutdown and an alarm to allow reboot at a later time. Systems without this capability may hang or may reboot immediately after shutdown.

−v Halts the operating system completely.

Parameters

+Time Specifies the time at which the **shutdown** command stops the system. An immediate shutdown is indicated by the word `now` displayed on the screen. A future time can be specified in one of two formats: +number or hour:minute. The first form brings the system down in the specified number of minutes and the second brings the system down at the time of day indicated (as a 24−hour clock). If the *Message* parameter is specified, the *Time* parameter must also be specified.

Message Specifies the message

Examples

1. To turn off the machine, enter:

   ```
   shutdown
   ```

   This shuts down the system, waiting 1 minute before stopping the user processes and the **init** process.

2. To give users more time to finish what they are doing and bring the system to maintenance mode, enter:

   ```
   shutdown -m +2
   ```

This brings the system down from multiuser mode to maintenance mode after waiting 2 minutes.

Files

/usr/sbin/shutdown Contains the **shutdown** command.

Related Information

The **errpt** command, **init** or **telinit** command, **kill** command, **killall** command, **halt** command, **reboot** command, and **sync** command.

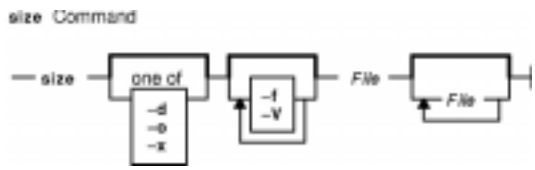
The **sigaction** subroutine.

size Command

Purpose

Displays the section sizes of the Extended Common Object File Format (XCOFF) object files.

Syntax



size [**−d** | **−o** | **−x**] [**−f**] [**−V**] [**−X** {**32**|**64**|**32_64**}]*File* [*File* ...]

Description

The **size** command writes to standard output the number of bytes required by all sections, along with their sum for each XCOFF file. If the **−f** flag is specified, the section name follows the section size.

Flags

The output is in decimal notation unless you change the output with the following flags:

−d Writes in decimal notation.

−f Writes the section name in parenthesis following the section size.

−o Writes in octal notation.

−x Writes in hexadecimal notation.

−X*mode* Specifies the type of object file **size** should examine. The *mode* must be one of the following:

32 Processes only 32−bit object files

64 Processes only 64−bit object files

32_64 Processes both 32−bit and 64−bit object files

The default is to process 32−bit object files (ignore 64−bit objects). The *mode* can also be set with the **OBJECT_MODE** environment variable. For example, **OBJECT_MODE=64** causes **size** to process any 64−bit objects and ignore 32−bit objects. The **−X** flag overrides the **OBJECT_MODE** variable.

−V Prints the version number of the **size** command.

Examples

1. To display the size of the **a.out** file in decimal, enter:

   ```
   size
   ```

   This displays the size in bytes of the executable **a.out** file. The size of each section of the object file is given, followed by the total:

   ```
   3720 + 1752 + 4152 = 9624
   ```

2. To display the size of an object file in octal, enter:

   ```
   size -o driver.o
   ```

   This displays the size of the **driver.o** object file in octal.

3. To display the size of several object files in hexadecimal, enter:

   ```
   size -x *.o
   ```

This displays in hexadecimal the size of each file ending with **.o** in the current directory.

Related Information

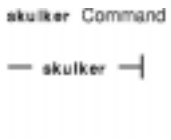
The **ar** command, **as** command, **dump** command, **ld** command, **nm** command, **strip** command.

skulker Command

Purpose

Cleans up file systems by removing unwanted files.

Syntax



skulker

Description

> **Attention:** Because the **skulker** command is run by a root user, and its whole purpose is to remove files, it has the potential for unexpected results. Before installing a new **skulker** command, test any additions to its file removal criteria by running the additions manually using the **xargs−p** command. After you have verified that the new **skulker** command removes only the files you want removed, you can install it.

The **skulker** command is a command file for periodically purging obsolete or unneeded files from file systems. Candidate files include files in the **/tmp** directory, files older than a specified age, **a.out** files, **core** files, or **ed.hup** files.

The **skulker** command is normally invoked daily, often as part of an accounting procedure run by the **cron** command during off−peak periods. Modify the **skulker** command to suit local needs following the patterns shown in the distributed version. Local users should be made aware of the criteria for automatic file removal.

The **find** command and the **xargs** command form a powerful combination for use in the **skulker** command. Most file selection criteria can be expressed conveniently with **find** expressions. The resulting file list can be segmented and inserted into **rm** commands using the **xargs** command to reduce the overhead that would result if each file were deleted with a separate command.

Related Information

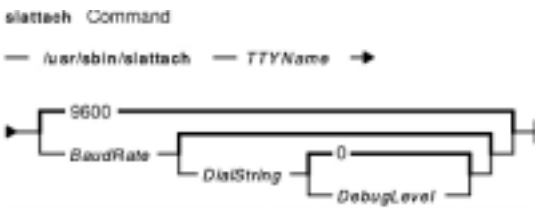
The **cron** daemon, **find** command, **rm** command, **xargs** command.

slattach Command

Purpose

Attaches serial lines as network interfaces.

Syntax



/usr/sbin/slattach *TTYName* [*BaudRate DialString* [*DebugLevel*]]

Description

The **/usr/sbin/slattach** command assigns a TTY line to a network interface.

The **slattach** command is run by the **/etc/rc.net** file during system startup to automatically configure any Serial Line Internet Protocol (SLIP) network interfaces defined by the System Management Interface Tool (SMIT). SLIP interfaces can also be configured manually as shown in the examples section.

For a directly connected SLIP interface, broken connections are retried automatically without manual intervention. For a SLIP interface connected by modem, broken connections must be manually redialed. If a user supplies a dial string in the **slattach** command line, the user must re−enter the command and dial string to restore a broken connection.

To detach the interface, run the **ifconfig** *Interface* **down** command after terminating the **slattach** command. The *Interface* parameter is the name shown by the **netstat** command.

If configuring a slip interface from the command line, the **/usr/sbin/ifconfig** command must be invoked for the slip interface with the appropriate parameters and the slip tty line discipline must also be available in order for this command to succeed. To check if the slip tty line discipline is already loaded, run the command `strinfo −m | grep slip`. If no output is shown, the module has not yet been loaded. Load the module by issuing the command `strload −m /usr/lib/drivers/slip`.

Notes:

1. Once the SLIP interface has been configured with **ifconfig**, any user who has permission on the TTY may issue the **slattach** command.
2. You must configure the tty devices used by the **slattach** command before establishing a connection. You may also need to make an entry for the tty device in the BNU **/usr/lib/uucp/Devices** file.
3. Sample shell script, **/usr/sbin/slipcall**, provides a simplified interface for invoking **slattach** and connecting to remote systems. **slipcall** is useful for connecting to dial−in SLIP networks which require a user to login before activating the SLIP tty line discipline. The basic configuration of **slipcall** will connect to other AIX systems with **sliplogin** configurations and derive the local and remote internet addresses and network mask assigned by the called system. It then configures the local interface

with the remote system's specified values.

Parameters

BaudRate	Sets the speed of the connection. The default speed is 9600.
DebugLevel	Sets the level of debug information desired. A number from 0 through 9 may be specified. A value of 0 specifies no debug information; a value of 9 specifies the most debug information. The default value is 0.
DialString	Specifies a string of expect/respond sequences using the Basic Networking Utility (BNU)/UNIX to UNIX Copy Program (UUCP) **chat** syntax.
TTYName	Specifies a TTY line. This string is in the form `ttyxx` or `/dev/ttyxx`.

Examples

1. To attach the SLIP network interface to the `tty1` port with a direct connection, issue the following command:

   ```
   slattach /dev/tty1
   ```

   This command attaches `tty1` to a network interface to be used by the SLIP.

2. To attach the SLIP network interface to `tty1` using a modem connection, issue the following command:

   ```
   slattach /dev/tty1 9600 '""AT OK \pATF1 OK \pATDT34335 CONNECT""'
   ```

Files

/etc/uucp/Devices Lists definitions of devices used for remote connections.

Related Information

The **ifconfig** command, **netstat** command, **sliplogin** command.

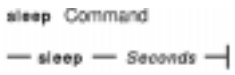
TCP/IP Network Interfaces in *AIX Version 4.3 System Management Guide: Communications and Networks*.

sleep Command

Purpose

Suspends execution for an interval.

Syntax



sleep *Seconds*

Description

The **sleep** command suspends execution of a process for at least the interval specified by the *Seconds* parameter. The amount of time specified in the *Seconds* parameter can range from 1 to **MAXINT** (2,147,483,647) seconds.

Exit Status

This command returns the following exit values:

0 The execution was successfully suspended for at least *Seconds* seconds, or a **SIGALRM** signal was received.

>0 An error occurred.

Examples

1. To run a command after a certain amount of time has passed, enter:

   ```
   (
   echo  "SYSTEM SHUTDOWN IN 10 MINUTES!" | wall
   sleep 300; echo "SYSTEM SHUTDOWN IN 5 MINUTES!" | wall
   sleep 240; echo "SYSTEM SHUTDOWN IN 1 MINUTE!" | wall
   sleep 60; shutdown
   )&
   ```

   This command sequence warns all users 10 minutes, 5 minutes, and 1 minute before the system is shut down.

2. To run a command at regular intervals, enter:

   ```
   while true
   do
    date
    sleep 60
   done
   ```

This shell procedure displays the date and time once a minute. To stop it, press the Interrupt key sequence.

Related Information

The **shutdown** command, **wall** command.

The **alarm** subroutine, **pause** subroutine, **sigaction** subroutine, **sleep** subroutine.

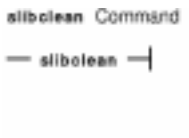
Shells Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

slibclean Command

Purpose

Removes any currently unused modules in kernel and library memory.

Syntax



slibclean

Description

The **slibclean** command unloads all object files with load and use counts of 0. It can also be used to remove object files that are no longer used from both the shared library region and in the shared library and kernel text regions by removing object files that are no longer required.

Files

/usr/sbin/slibclean Contains the **slibclean** command.

Related Information

The **unload** subroutine.

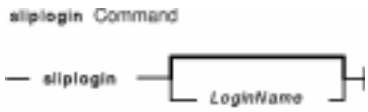
Using Kernel Processes in *AIX Version 4.3 Kernel Extensions and Device Support Programming Concepts*.

sliplogin Command

Purpose

Converts a standard−input terminal line into a Serial Line Internet Protocol (SLIP) link to a remote host.

Syntax



sliplogin [*LoginName*]

Description

The **sliplogin** command configures a standard−input terminal line into a Serial Line Internet Protocol (SLIP) link to a remote host; that is, the command attaches a serial line network interface.

> **Note:** User requires root authority to attach a network interface.

The **sliplogin** command searches the **/etc/slip.hosts** file for a *loginname* entry that matches the value of the *LoginName* parameter. If a matching entry is found, **sliplogin** configures the line appropriately for SLIP (that is, for 8−bit transparent input/output) and converts it to SLIP line discipline. Then, **sliplogin** invokes the applicable login shell script which initializes the SLIP interface with the local and remote Internet Protocol (IP) addresses, netmask, and optional arguments associated with the *loginname* entry in the **/etc/slip.hosts** file.

The usual initialization script file is **/etc/slip.login**. However, in order to accommodate special initialization needs of a particular host, a script file named **/etc/slip.login.***userlogin* (where *userlogin* corresponds to the *loginname* entry in the **/etc/slip.hosts** file) can be created. The **sliplogin** command uses the **/etc/slip.login.***userlogin* script file when it exists, instead of the **/etc/slip.login** script file.

To deinitialize the SLIP interface, the **sliplogin** command uses either the **/etc/slip.logout** script file or the **/etc/slip.logout.***userlogin* script file, if one of them exists, with preference given to the latter. The **/etc/slip.logout** script file is given the same arguments as the **/etc/slip.login** script file; the **/etc/slip.logout.***userlogin* script file is given the same arguments as the **/etc/slip.login.***userlogin* script file. In its default form, the **/etc/slip.logout** script file deletes all routes through the network interface for the specified SLIP unit. Additional processes to be done when the SLIP interface is disconnected can be added to either logout script file.

Notes:

1. The interface automatically deactivates when the remote connection terminates or when the **sliplogin** command dies.
2. Use the **slattach** command to access a remote system that has a SLIP link configured. Use the sample shell script file **/usr/sbin/slipcall** to invoke the **slattach** command with the proper parameters needed to call a remote system and configure the local interface with the appropriate values assigned by the remote system.
3. When using **sliplogin** as a user's login shell on a tty device, then this tty port used

needs to be enabled for login. (This differs from the configuration when using **slattach** instead of **sliplogin** as a SLIP server process.

/etc/slip.hosts File

The **/etc/slip.hosts** file is the configuration file containing the names of preconfigured sliplogin users and the IP addresses to be assigned to the local and remote interface when the user logs in. **sliplogin** searches this file for matching *LoginName* entries. This file has the following format:

- Comments (lines starting with a #) and blank lines are ignored.
- Other lines must start with a *loginname* argument, and the fields should contain whatever is appropriate for the **slip.login** file that is executed for that name.
- Arguments are separated by white space and follow normal sh(1) quoting conventions. However, the *loginname* argument cannot be quoted. Usually lines have the following form:

  ```
  loginname local_address remote_address netmask opt_args
  ```

  where *local_address* and *remote_address* are the IP host names or addresses of the local and remote ends of the SLIP line, and *netmask* is the appropriate IP netmask. These arguments are passed directly to the **ifconfig** command. *Opt_args* are optional arguments used to configure the line.

- The AIX implementation of **sliplogin** allows the **/etc/slip.hosts** file to contain multiple entries for a single SLIP user with differing addresses. This enables multiple SLIP interfaces to be activated by the **sliplogin** command for the same user name. When user entries are retrieved from the **/etc/slip.hosts** file, only entry addresses meeting the following criteria are selected.

  The entry is ignored if a **slip.hosts** entry specifies a local address which is already in use on another non−SLIP interface on the local system.

  The entry is ignored if the remote address specified in an **/etc/slip.hosts** entry is already in use on any other interface.

/etc/slip.login File

The **/etc/slip.login** or **/etc/slip.login.***userlogin* file is the setup script invoked by the **sliplogin** command to initialize the user's network interface. The **/etc/slip.login.***userlogin* file is invoked if it exists, where the value of the *LoginName* parameter of the **sliplogin** command corresponds to a *loginname* entry in the **/etc/slip.hosts** file. If this file cannot be accessed, the **/etc/slip.login** file is invoked instead. The login script file contains the following parameters:

slipunit	Specifies the unit number of SLIP interface assigned to this line. For example, 0 for sl0 (sl0 is s, lowercase L, zero.)
speed	Specifies the speed of the line.
args	Specifies the arguments from the **/etc/slip.hosts** file entries, in order, starting with *loginname*.

/etc/slip.logout File

The **/etc/slip.logout** or **/etc/slip.logout.***userlogin* file is the setup script invoked by **sliplogin** to deinitialize the user's network interface. The **/etc/slip.logout.***userlogin* file is invoked if it exists, where the value of the *LoginName* parameter of **sliplogin** corresponds to a *loginname* entry in the **/etc/slip.hosts** file. If this file cannot be accessed, the **/etc/slip.logout** file is invoked instead.

Flags

</dev/ttyx Redirects the command to the **ttyx** device if the user is already logged into a **tty** device and wants

to configure their terminal as a SLIP line.

Parameters

LoginName Specifies the desired login name. The default is the current login name.

Example

The normal use of the **sliplogin** command is to create an **/etc/passwd** entry for each legal, remote SLIP site with **sliplogin** as the shell for the entry. For example,

```
foo:!:2010:1:slip line to foo:/tmp:/usr/sbin/sliplogin
```

An entry must then be added to the **/etc/slip.hosts** file. The entry should resemble the following example:

```
foo 1.1.1.1 1.1.1.2 0xffffff00 normal
```

where *loginname* = `foo`, *local_address* = `1.1.1.1`, *remote_address* = `1.1.1.2`, *netmask* = `0xffffff00,` and `opt_args = normal.` (The optional argument `normal` indicates which SLIP mode to activate. For AIX release 4.1, only normal mode is supported.)

Diagnostics

The **sliplogin** command logs various information to the system log daemon (**syslogd**). The messages are listed here, grouped by severity levels.

- Error Severity

Message	**Description**
ioctl (TCGETS): *reason*	The **ioctl** subroutine failed to get the line parameters for the reason indicated.
ioctl (TCSETS): *reason*	The **ioctl** subroutine failed to set the line parameters for the reason indicated.
ioctl (TIOCGETD): *reason*	The **ioctl** subroutine failed to get the current **tty** discipline for the reason indicated.
/etc/slip.hosts: *reason*	The **/etc/slip.hosts** file could not be opened for the reason indicated.
Check of flags for interface *xxx* failed. Errno is reason.	An attempt to check the status of the indicated interface to avert possible addressing conflicts failed for the reason indicated in the **errno** global variable.
Access denied for user − no /etc/slip.login[.*userlogin*] file.	No **/etc/slip.login** or **/etc/slip.login.***userlogin* script file could be found.
Access denied for user − no /etc/slip.hosts entries available.	No *loginname* entry in the **/etc/slip.hosts** file matched the *LoginName* value specified in the command.
Access denied − getlogin returned 0.	The user issuing the **sliplogin** command does not have a password entry in the **/etc/passwd** file.

Logout script failed: exit status *xxx* from /etc/slip.logout[.*userlogin*]	An attempt to run the **/etc/slip.logout** or **/etc/slip.logout.***userlogin* script file failed with the indicated exit status.
No SLIP interface for ttyx. Errno is reason.	No SLIP interface could be located for the **ttyx** device for the reason indicated in the **errno** global variable. Try either running the **ifconfig slx up** command or using SMIT to add a network interface for the **tty** device.
Open /dev/null: *reason*	An attempt to open the **/dev/null** device failed for the reason indicated.
/etc/slip.logout file not found	The **/etc/slip.logout** file could not be located.
sliplogin: cannot add SLIP discipline to ttyx	No SLIP interface exists for the **ttyx** device. Try either running the **ifconfig slx up** command or using SMIT to add a network interface for the **tty** device.
SLIP discipline removal from tty failed. Errno is reason.	An attempt to remove the SLIP discipline from the **tty** device failed for the reason indicated in the **errno** global variable.
tcgetattr: *reason*	An attempt to read the current attributes of the **tty** device failed for the reason indicated.
userlogin login failed: exit status *xxx* from /etc/slip.login[.*userlogin*]	A system call to execute the **/etc/slip.login** or **/etc/slip.login.***userlogin* script file failed with the indicated exit status.

- Information Severity

Message	**Description**
Attaching SLIP unit *xxx* for *userlogin* on ttyx.	The **sliplogin** command found a *loginname* entry in the **/etc/slip.hosts** file that matched the *LoginName* value specified in the command, invoked the applicable **/etc/slip.login** or **/etc/slip.login.***userlogin* file, and is now attaching the indicated network interface.
Closed *userlogin* SLIP unit *xxx* (signal)	The indicated SLIP unit for the indicated *userlogin* was closed because the **sliplogin** command terminated due to a signal.

- Notice Severity

Message	**Description**
Attaching SLIP unit *xxx* for *userlogin*.	The indicated SLIP unit has been successfully attached for the indicated *userlogin*.

Files

/etc/slip.hosts	The configuration file that contains the names of preconfigured sliplogin users and the IP addresses to be assigned to the local and remote interface when the user logs in.
/etc/slip.login or **/etc/slip.login.***userlogin*	The setup script invoked by the **sliplogin** command to initialize the user's network interface.
/etc/slip.logout or **/etc/slip.logout.***userlogin*	The setup script invoked by the **sliplogin** command to deinitialize the user's network interface.

Related Information

The **slipcall** command script file.

The **slattach** command.

slocal Command

Purpose

Processes incoming mail.

Syntax

slocal [**−verbose** | **−noverbose**] [**−debug**]

Description

The **slocal** command performs a set of actions each time a message is sent to the user. The **slocal** command is not started by the user. The **slocal** command is called by the **sendmail** command.

The **sendmail** command starts the **slocal** command upon encountering the following line in the **$HOME/.forward** files:

```
/usr/lib/mh/slocal
```

For each incoming message, the **slocal** command performs the actions specified in the **.maildelivery** file. If the **slocal** command cannot find the **$HOME/.maildelivery** file, the **slocal** command uses the **/etc/mh/maildelivery** default file. If the delivery request fails, the **slocal** command delivers the message to the **/usr/mail/$USER** file.

Flags

−debug	Provides information for debugging.
−help	Lists the command syntax, available switches (toggles), and version information. **Note:** For Message Handler (MH), the name of this flag must be fully spelled out.
−noverbose	Does not display information as the system executes commands in the **.maildelivery** file. This flag is the default.
−verbose	Displays information as the system executes commands in the **.maildelivery** file.

Files

/usr/lib/mh/mtstailor	Contains MH command definitions.
/etc/mh/.maildelivery	Contains the default MH instructions for local mail delivery.
$HOME/.maildelivery	Provides the user with MH instructions for local mail delivery.
$HOME/.forward	Contains either the line that starts the **slocal** command or a path to forward mail.
/etc/mh/mh_profile	Contains parameters that customize the MH package.

Related Information

The **rcvdist** command, **rcvpack** command, **rcvstore** command, **rcvtty** command, **sendmail** command.

Mail Overview in *AIX Version 4.3 System User's Guide: Communications and Networks*.

Peek, Jerry. *MH and xmh: E−mail for Users and Programmers.* Sebastopol, CA: O'Reilly & Associates, 1992.

smcaprop Command

Purpose

Provides read−only information on the Certificate Authority.

Syntax

smcaprop

Description

The **smcaprop** command can be run on a machine that has been defined as internal Certificate Authority (CA). The command prompts for the CA private key ring password, and then provides read−only information on the CA (CA Name, Most recent certificate issued, CA certificate expiration date, etc.). Detailed information on all operations executed by the CA (key ring generation, certificate signing, etc.) can be found in the CA log file **/usr/websm/security/SMCa.log**.

Use the **/usr/websm/bin/wsm** command to access the graphical interface. The fast path is **wsm system**.

Examples

```
smcaprop
```

Files

/usr/websm/security/SMCa.log	Lists detailed information on all operations executed by the CA.
/usr/websm/security/SM.caprivkr	Certificate private key ring file.

Related Information

The **smdefca**, **smexpcacert**, **smimpcacert**, **smlistcerts**, **smsigncert**, and the **smundefca** command.

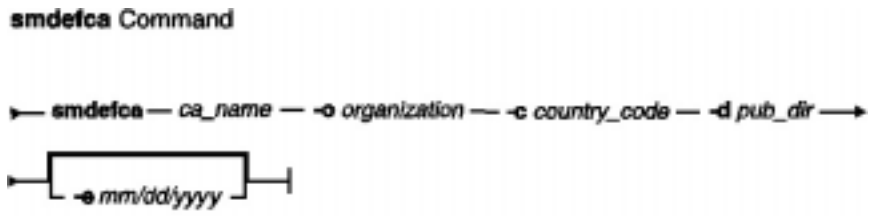
See Setting Up and Running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

smdefca Command

Purpose

Defines an internal certificate authority.

Syntax



smdefca*ca_name* **−o***organization* **−c***country_code* **−d***pub_dir* [**−e***mm/dd/yyyy*]

Description

The **smdefca** command is used to define an internal CA (Certificate Authority) for Web−based System Manager servers and clients on the current machine. When you define a Web−based System Manager−CA, the following files are generated:

/usr/websm/security/SM.caprivkr
: This is the CA private key ring that includes the CA private key and the CA certificate. This is the most sensitive file from the aspect of Web−based System Manager security. It is created **root** protected and password encrypted.

SMpubkr.class *(created on the specified pub_dir)*
: The public key ring file. This file has to be distributed to each Web−based System Manager client (for application mode) and server (for applet mode) and should be placed in **/usr/websm/codebase**.

If a CA is already defined on the current machine, the **smundefca** command must be used first to unconfigure it.

Use the **/usr/websm/bin/wsm** command to access the graphical interface. The fast path is **wsm system**.

Flags

ca_name	A name that uniquely defines your Web−based System Manager−CA. The machine full TCP/IP name with some additional serial number might be a good choice. If you ever redefine a CA, it is recommended that you use a different name in order to identify which CA, by name, is used by each server and client. **Note:** Do not set the CA name to be exactly the machine's full TCP/IP name (this will break the SMGate utility, in case you want to use it in managing this machine from a remote browser).
−o*organization*	Organization name (required for the CA certificate).
−c*country_code*	Two−letter ISO country code (required for the CA certificate).
−d*pub_dir*	The output directory for the public key ring file **SMpubkr.class**.
−e*mm/dd/yyyy*	Expiration date for the CA certificate. The default expiration date is four years from the date of issuing the command.

Examples

```
smdefca IBMCA1 −o IBM −c US −d /usr/websm/security/tmp −e 12/31/1999
```

Files

/usr/websm/security/SMpubkr.class	CA public key ring file.
/usr/websm/security/SMCa.log	Lists detailed information on all operations executed by the CA.
/usr/websm/security/SMCa.sn	Certificate number file.
/usr/websm/security/SM.caprivkr	Certificate private key ring file.

Related Information

The **smcaprop**, **smexpcacert**, **smimpcacert**, **smlistcerts**, **smsigncert**, and the **smundefca** command.

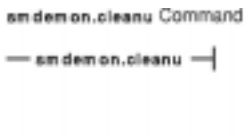
See the Setting Up and Running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

smdemon.cleanu Command

Purpose

Cleans up the **sendmail** queue for periodic housekeeping.

Syntax



/usr/lib/smdemon.cleanu

Description

The **smdemon.cleanu** command is a shell procedure that cleans up the **sendmail** command queue and maintains the **/var/spool/mqueue/log** file.

To enable the **smdemon.cleanu** command, you must remove the comment statement by deleting the # (pound sign) character from the beginning of the **smdemon.cleanu** line in the **/var/spool/cron/crontabs/root** file. If the **/var/spool/mqueue** directory does not exist, do not change the **/var/spool/cron/crontabs/root** file.

Be careful that the average size of a log file for each **smdemon.cleanu** session multiplied by the number of log files does not use more space than you need. You can arrange the number of log files to suit your needs.

> **Note:** The **smdemon.cleanu** command is not usually entered on the command line. The command is executed by the **cron** daemon.

Examples

To run the **smdemon.cleanu** procedure automatically, edit the **/var/spool/cron/crontabs/root** file and delete the # (comment character) from the beginning of the **smdemon.cleanu** line as follows:

```
# ulimit 5000; /usr/lib/smdemon.cleanu > /dev/null
```

Files

/var/spool/cron/crontabs/root	Schedules when the **smdemon.cleanu** command will run.
/var/spool/mqueue	Contains the **log** file and temporary files associated with the message in the mail queue.

Related Information

The **cron** daemon.

The **sendmail** command.

Managing Mail LoggingHow to Manage the Log in *AIX Version 4.3 System Management Guide:*

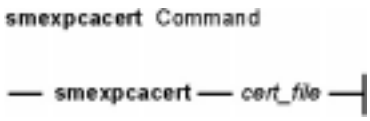
Communications and Networks.

smexpcacert Command

Purpose

Exports the CA certificate.

Syntax



smexpcacert*cert_file*

Description

The **smexpcacert** command can be run on a machine that has been defined as internal certificate authority (CA). The command prompts for the CA private key ring password, and then writes the CA certificate of the internal CA to the file *cert_file*. The full path name of the output file for the CA certificate is specified with *cert_file*.

Use the **/usr/websm/bin/wsm** command to access the graphical interface. The fast path is **wsm system**.

Examples

```
smexpcacert /tmp/CA1.cert
```

Files

/usr/websm/security/SMCa.log Lists detailed information on all operations executed by the CA.

Related Information

The **smcaprop**, **smdefca**, **smimpcacert**, **smlistcerts**, **smsigncert**, and the **smundefca** command.

See Setting Up and Running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

smgenkeycr Command

Purpose

Generates server private keys and certificate requests.

Syntax

smgenkeycr {*server_name* | **−f***list_file*} **−o***organization***−c***country_code***−d***out_dir* [**−k 512**]

Description

The **smgenkeycr** command generates a private key and a certificate request for the Web−based System Manager servers (managed AIX machines). The private keys and certificate requests are generated either for each server in the input server list file or for the server whose name is specified.

Use the **/usr/websm/bin/wsm** command to access the graphical interface. The fast path is **wsm system**.

Flags

server_name	The full TCP/IP name of the server. If the name specified is **S**, a private key ring file **S.privkr** will be generated in the output directory.
−f*list_file*	The full path name of a file with the list of server machine names (one line with full TCP/IP name per server). For each server **S** in the list, a private key ring file **S.privkr** will be generated in the output directory.
−o*organization*	Organization name (required for the server certificate).
−c*country_code*	Two−letter ISO country code (required for the server certificate).
−d*out_dir*	The output directory for the server private key ring files.
−k 512	This option does not exist in the exportable version. The server private key length will be 512. The default in the US version is 1024, in the exportable − 512.

Examples

```
smgenprivkr S101.IBM.COM −o IBM −c US −d /usr/websm/security/tmp
smgenprivkr −f /usr/websm/security/tmp/server.list −o IBM −c US −d /usr/websm/security/tmp
```

Related Information

The **smgenprivkr**, **smimpservercert**, **sminstkey**, **smlistcerts**, and the **smserverprop** command.

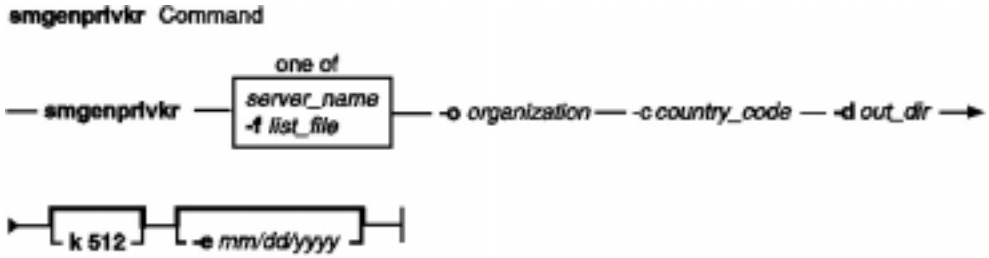
See Setting Up and Running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

smgenprivkr Command

Purpose

Generates server private key ring files./

Syntax



smgenprivkr {*server_name*| **−f***list_file*} **−o***organization***−c***country_code***−d***out_dir* [**−k 512**] [**−e***mm/dd/yyyy*]

Description

The **smgenprivkr** command can be run on a machine that has been defined as internal certificate authority (CA). The **smgenprivkr** command generates 'ready to go' private key ring files for the Web−based System Manager servers (managed AIX machines). The private key ring files are generated either for each server in the input server list file, or for the one server whose name is specified.

Use the **/usr/websm/bin/wsm** command to access the graphical interface. The fast path is **wsm system**.

Flags

server_name	The full TCP/IP name of the server. If the name specified is **S**, a private key ring file **S.privkr** will be generated in the output directory.
−f*list_file*	The full path name of a file with the list of server machine names (one line with full TCP/IP name per server). For each server **S** in the list, a private key ring file **S.privkr** will be generated in the output directory.
−o*organization*	Organization name (required for the server certificate).
−c*country_code*	Two−letter ISO country code (required for the server certificate).
−d*out_dir*	The output directory for the servers private key ring files.
−k 512	This option does not exist in the exportable version. The server's private key length will be 512. The default in the US version is 1024, in the exportable 512.
−e*mm/dd/yyyy*	Expiration date for the server certificates. The default expiration date is two years from the date of issuing the command.

Examples

```
smgenprivkr S101.IBM.COM −o IBM −c US −d /usr/websm/security/tmp −e 12/31/1999
```

```
smgenprivkr −f /usr/websm/security/tmp/server.list −o IBM −c US −d /usr/websm/security/tmp
```

Files

/usr/websm/security/SMCa.log Lists detailed information on all operations executed by the CA.

Related Information

The **smgenkeycr**, **smimpservercert**, **sminstkey**, **smlistcerts**, and the **smsigncert** command.

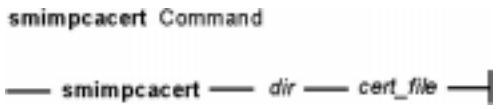
See the Setting Up and Running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

smimpcacert Command

Purpose

Imports the certificate authority's certificate.

Syntax



smimpcacert*dircert_file*

Description

The **smimpcacert** command imports the CA certificate *cert_file* into the public key ring file **SMpubkr.class** that resides on the specified directory *dir*. If there is no **SMpubkr.class** file in *dir*, a new **SMpubkr.class** containing only the certificate of *cert_file* is created there.

Use the **/usr/websm/bin/wsm** command to access the graphical interface. The fast path is **wsm system**.

Flags

dir	The directory of **SMpubkr.class**.
cert_file	The full path name of the CA certificate file.

Examples

```
smimpcacert /usr/websm/security/tmp CA1.cert
```

Related Information

The **smcaprop**, **smdefca**, **smexpcacert**, **smlistcerts**, **smsigncert**, and the **smundefca** command.

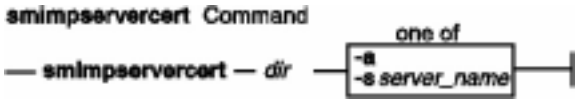
See the Setting Up and Running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

smimpservercert Command

Purpose

Imports the Server Certificate.

Syntax



smimpservercert*dir* {**−a** | **−s***server_name*}

Description

The **smimpservercert** command imports a server certificate (***.cert** file) to the server private key file (***.privk**), generating a private key ring file (***.privkr**). You can import the certificate of one server, or all certificates with matching private key files in the specified directory *dir*.

Use the **/usr/websm/bin/wsm** command to access the graphical interface. The fast path is **wsm system**.

Flags

dir	The directory where the certificate requests (***.certreq** files) and the private keys (***.privk** files) reside, and to which the private key rings (***.privkr** files) will be written.
−a	All certificates (***.certreq** files) with matching private key files (***.privk**) in the specified *dir* directory will be processed. Each certificate **S.cert** will be imported into the private key file **S.privk**, generating the private key ring file **S.privkr** in the specified *dir* directory.
−s*server_name*	The full TCP/IP name of the server whose certificate *server_name***.cert** will be imported into its private key file *server_name***.privk**, generating the private key ring file *server_name***.privkr** in the specified *dir* directory.

Examples

```
smimpservercert /usr/websm/security/tmp S101.IBM.COM

smimpservercert /usr/websm/security/tmp −a
```

Related Information

The **smgenkeycr**, **smgenprivkr**, **sminstkey**, **smlistcerts**, and the **smserverprop** command.

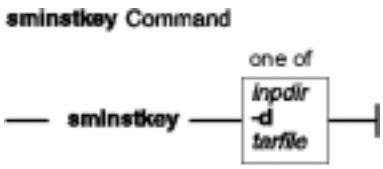
See the Setting Up and Running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

sminstkey Command

Purpose

Installs private key ring.

Syntax



sminstkey [*inpdir* | **−d** | *tarfile*]

Description

The **sminstkey** command expects the private key ring file of the current machine (**S.privkr** where **S** is the full TCP/IP machine name) in a directory, or on a diskette archive created by TAR, or in a TAR file, and installs it as **/usr/websm/security/SM.privkr**.

> **Note:** In case of TAR file or diskette TAR, the private key ring should appear there without a path.

If the source private key ring file is password−encrypted, the command prompts for the password.

Use the **/usr/websm/bin/wsm** command to access the graphical interface. The fast path is **wsm system**.

Flags

inpdir The source **S.privkr** is in the directory *inpdir*.
−d The source **S.privkr** is in a diskette archive created by TAR.
tarfile The source **S.privkr** is in the TAR file *tarfile*.

Examples

```
sminstkey /usr/websm/security/tmp
```

```
sminstkey −d
```

```
sminstkey /afs/security/privkrs.tar
```

Files

/usr/websm/security/SM.privkr Server private key ring file.

Related Information

The **smgenkeycr**, **smgenprivkr**, **smimpservercert**, **smlistcerts**, and the **smserverprop** command.

See the Setting Up and Running Web−based System Manager in *AIX Version 4.3 System Management*

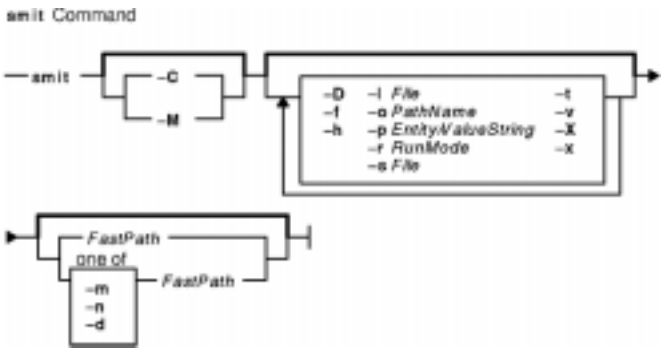
Guide: Operating System and Devices.

smit Command

Purpose

Performs system management.

Syntax



smit [**−C** | **−M**] [**−D**] [**−f**] [**−h**] [**−l** *File*] [**−o** *PathName*] [**−p** *Entity/ValueString*] [**−r** *RunMode*] [**−s** *File*] [**−t**] [**−v**] [[**−m** | **−n** | **−d**] *FastPath*] [**−X**] [**−x**]

Description

The **smit** command invokes the System Management Interface Tool (SMIT). SMIT is an interactive interface application designed to simplify system management tasks. The **smit** command displays a hierarchy of menus that can lead to interactive dialogues. SMIT builds and runs commands as directed by the user. Because SMIT runs commands, you need the authority to execute the commands that SMIT runs.

SMIT creates two files, the **smit.script** file and the **smit.log** file. Invoking the **smit** command with the **−s***PathName* flag saves the **smit.script** file in the file specified by the *PathName* parameter. If the **−s** flag is not specified, the script information is saved in the **$HOME/smit.script** file. Invoking the **smit** command with the **−l***PathName* flag saves the **smit.log** file in the file specified by the *PathName* parameter. If the **−l** flag is not specified, the log information is recorded in the **$HOME/smit.log** file. You must have write permission for the directory in which you have requested the **smit** file to be written or the **smit.script** file and **smit.log** file are not created. SMIT does not overwrite the **smit.log** file or the **smit.script** file. The files are appended when possible.

The **smit.script** file automatically records the commands with the command flags and parameters used. The **smit.script** file can be used as an executable shell script to duplicate system configuration. SMIT creates the **smit.log** file, which contains additional detailed information that can be used by programmers in extending the SMIT system. The **smit.log** file is affected by the **−D**, **−l**, **−t**, and **−v** flags.

The **smit** command takes you to the top level of the menu hierarchy if you do not use the *FastPath* parameter. To enter the menu at lower levels, use the *FastPath* parameter. All commands run by SMIT can be used as *FastPaths*. The *FastPath* parameter will assist you as you become familiar with the commands. For example, you can enter: `smit chuser` to go directly to the dialog from which you can change user characteristics. To learn more about *FastPaths* see, How to Define FastPaths in SMIT in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

SMIT requires access to the following files:

sm_menu_opt	SMIT database
sm_name_hdr	SMIT database
sm_cmd_hdr	SMIT database
sm_cmd_opt	SMIT database
smit.log	SMIT log file
smit.script	SMIT script file
/usr/lpp/msg/.../smit.cat	Message Catalog

Note: If any of these files are corrupt, or exist on an NFS server and that server goes down, SMIT may fail to respond.

Flags

−C	Starts SMIT using an ASCII (also called Curses) interface.
−D	Sets the debug mode; sets **−t** and **−v** flags.
−d*FastPath*	Identifies that the *FastPath* is the name of a dialogue.
−f	Allows standard in and standard out from SMIT to be redirected.
−h	Displays the command usage message.
−l*File*	Redirects the **smit.log** file to the specified *File*.
−M	Starts SMIT using a windows (also called Motif) interface.
−m*FastPath*	Identifies that the *FastPath* is the name of a menu.
−n*FastPath*	Identifies that the *FastPath* is the name of a selector.
−o *PathName*	Specifies a directory *PathName* of an alternate repository for SMIT objects. The default directory is **/etc/objrepos**.
−p *Entity/ValueString*	This flag only applies to smit Windows version. Allows nameselects and dialogs to be filled in from the command line. Also allows you to operate on multiple entities simultaneously. You can set the environment variables ENTITY_SEP and VALUE_SEP to override the default comma and semicolon separators. This flag applies only to AIX Version 4.2 or later. You can enter *Entity/ValueString* in any of the following formats: "*Entity1***:***Val1***,***Val2*... **;***Entity2***:***Val1***,***Val2*... **;** ..." or "*Val1***,***Val2*... **;***Val1***,***Val2*... **;** ..."
−r *RunMode*	This flag only applies to smit Windows version. Specifies the mode to run msmit in. This flag applies only to AIX Version 4.2 or later. You can enter the following values for *RunMode*: **1** Exit msmit when **done** is clicked in the output window. **2** Exit msmit when **ok** is clicked in a dialog. Print the dialog options upon exit. Do not run the command. **3** Run msmit silently, print the dialog options. Do not run the command. **4** Exit msmit when **ok** is clicked in the dialog. Print the commands upon exit. Do not run the command.
−s *File*	Redirects the **smit.script** file to the specified *File*.
−t	Records detailed trace information in the **smit.log** file.

−v	Records the command strings for intermediate and target task commands run by SMIT, and also records their output in the **smit.log** file.
−x	Does not run any **command_to_execute**, but still logs them for later execution.
−X	Does not run any **command_to_discover**, **command_to_list**, **command_to classify** or **command_to_execute**.

Examples

1. To display the main menu in the overall system management hierarchy, enter:

   ```
   smit
   ```

2. To change the characteristics of a user, enter:

   ```
   smit chuser
   ```

   The **chuser** command is an example of a *FastPath* parameter. The **smit** command and the *FastPath* parameter **chuser** takes you directly to the dialog, `Change User Attributes`, which guides you through changing the characteristics of a user.

3. To make the **smit.script** file executable for duplicate configuration, enter:

   ```
   chmod +x smit.script
   ```

   Then, to duplicate your configuration, enter:

   ```
   smit.script
   ```

   The **smit.script** file can be edited to create slight variations in the configuration commands, or to use only subsets of the commands. The **smit.script** file should be renamed or copied to prevent SMIT from modifying it.

   > **Note:** SMIT runs commands under the Korn shell (**/usr/bin/ksh**). Some command strings in the **smit.script** file may require this environment to run correctly.

Files

/usr/bin/smit	Contains the **smit** command.
/etc/objrepos	Specifies the default directory for the SMIT database.
smit.log	Specifies detailed information of your session, with time stamps.
smit.script	Specifies only the target task commands run by SMIT, with time stamps.

Related Information

The **chmod** command.

System Management Interface Tool (SMIT): Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

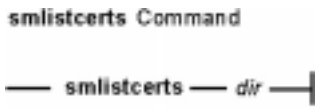
System Management Interface Tool (SMIT) Overview for Programming in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

smlistcerts Command

Purpose

Lists CA certificates.

Syntax



smlistcerts*dir*

Description

The **smlistcerts** command lists the CA certificates contained in the public key ring file **SMpubkr.class** that resides on the specified directory *dir*. The directory of **SMpubkr.class** is specified by *dir*.

Use the **/usr/websm/bin/wsm** command to access the graphical interface. The fast path is **wsm system**.

Examples

```
smlistcerts /usr/websm/codebase
```

Related Information

The **smcaprop**, **smdefca**, **smexpcacert**, **smimpcacert**, **smsigncert**, and the **smundefca** command.

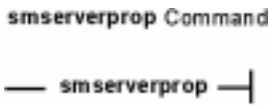
See Setting Up and Running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

smserverprop Command

Purpose

Lists server properties.

Syntax



smserverprop

Description

The **smserverprop** command provides read−only information on the local Web−based System Manager server (Name, key length, certificate expiration date, Certificate Authority name etc.).

Use the **/usr/websm/bin/wsm** command file to access the graphical interface. The fast path is **wsm system**.

Examples

```
smserverprop
```

Files

/usr/websm/security/SM.privkr Server private key ring file.

Related Information

The **smgenkeycr**, **smgenprivkr**, **smimpservercert**, **sminstkey**, and the **smlistcerts** command.

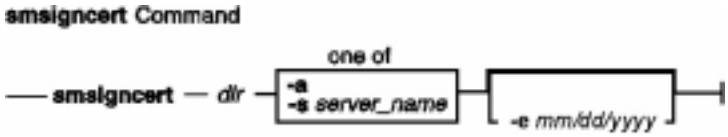
See the Setting Up and Running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

smsigncert Command

Purpose

Processes certificate requests and generates certificates.

Syntax



smsigncert*dir* {**−a** | **−s***server_name*} [**−e***mm/dd/yyyy*]

Description

The **smsigncert** command can be run on a machine that has been defined as internal certificate authority (CA). The command uses the CA private key to process certificate requests (***.certreq** files) and generate certificates (***.cert** files). You can process the request of one server, or all server requests in the specified directory *dir*.

Use the **/usr/websm/bin/wsm** command to access the graphical interface. The fast path is **wsm system**.

Flags

dir	The directory where the certificate requests (***.certreq** files) reside, and to which the certificates (***.cert** files) will be written.
−a	All certificate requests (***.certreq** files) in the specified *dir* directory will be processed. For each certificate request (**S.certreq**), a certificate **S.cert** will be generated in the specified *dir* directory.
−s*server_name*	The full TCP/IP name of the server whose certificate request (*server_name***.certreq** in the specified *dir* directory) will be processed.
−e*mm/dd/yyyy*	Expiration date for the server certificates. The default expiration date is two years from the date of issuing the command.

Examples

```
smsigncert /usr/websm/security/tmp S101.IBM.COM −e 12/31/1999

smsigncert /usr/websm/security/tmp −a
```

Files

/usr/websm/security/SMCa.log Lists detailed information on all operations executed by the CA.

Related Information

The **smcaprop**, **smdefca**, **smexpcacert**, **smlistcerts**, and the **smundefca** command.

See Setting Up and Running Web−based System Manager in *AIX Version 4.3 System Management Guide:*

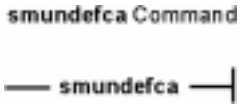
Operating System and Devices.

smundefca Command

Purpose

Unconfigures internal Certificate Authority.

Syntax



smundefca

Description

The **smundefca** command is used to unconfigure the internal certificate authority (CA) that was previously defined on the current machine.

The **smundefca** command removes the following files:

/usr/websm/security/SM.caprivkr	The CA private key ring which includes the CA private key and the CA certificate.
/usr/websm/security/SMCa.sn	The certificate number file.

The log file **/usr/websm/security/SMCa.log** is not deleted.

Use the **/usr/websm/bin/wsm** command to access the graphical interface. The fast path is **wsm system**.

Examples

```
smundefca
```

Files

/usr/websm/security/SMCa.log	Lists detailed information on all operations executed by the CA.
/usr/websm/security/SM.caprivkr	Certificate private key ring file.
/usr/websm/security/SMCa.sn	Certificate number file.

Related Information

The **smcaprop**, **smdefca**, **smexpcacert**, **smimpcacert**, **smlistcerts**, and the **smsigncert** command.

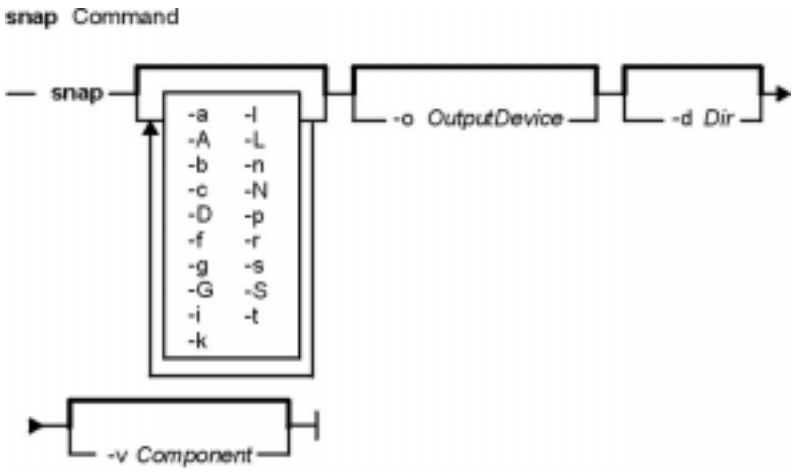
See Setting Up and Running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

snap Command

Purpose

Gathers system configuration information.

Syntax



snap [**−a**] [**−A**] [**−b**] [**−c**] [**−D**] [**−f**] [**−g**] [**−G**] [**−i**] [**−k**] [**−l**] [**−L**][**−n**] [**−N**] [**−p**] [**−r**] [**−s**] [**−S**] [**−t**] [**−o** *OutputDevice*] [**−d***Dir*] [**−v** *Component*]

Description

The **snap** command gathers system configuration information and compresses the information into a **tar** file. The file can then be downloaded to disk or tape, or transmitted to a remote system. The information gathered with the **snap** command may be required to identify and resolve system problems.

> **Note:** Root user authority is required to execute the **snap** command.

Use the **snap −o /dev/rfd0** command to copy the compressed image to diskette. Use the **snap −o /dev/rmt0** command to copy the image to tape.

Approximately 8MB of temporary disk space is required to collect all system information, including contents of the error log. If you do not gather all system information with the **snap −a** command, less disk space may be required (depending on the options selected).

> **Note:** If you intend to use a tape to send a snap image to IBM for software support, the tape must be one of the following formats:
>
> - 8mm, 2.3 Gb capacity
> - 8mm, 5.0 Gb capacity
> - 4mm, 4.0 Gb capacity
>
> Using other formats prevents or delays IBM software support from being able to examine the contents.

The **snap −g** command gathers general system information, including the following:

- Error report
- Copy of the customized Object Data Manager (ODM) database
- Trace file
- User environment
- Amount of physical memory and paging space
- Device and attribute information
- Security user information

The output of the **snap −g** command is written to the **/tmp/ibmsupt/general/general.snap** file.

The **snap** command checks for available space in the **/tmp/ibmsupt** directory, the default directory for **snap** command output. You can write the output to another directory by using the **−d** flag. If there is not enough space to hold the **snap** command output, you must expand the file system.

Each execution of the **snap** command appends information to previously created files. Use the **−r** flag to remove previously gathered and saved information.

Flags

−a	Gathers all system configuration information. This option requires approximately 8MB of temporary disk space.
−A	Gathers asynchronous (TTY) information.
−b	Gathers SSA information.
−c	Creates a compressed **tar** image (**snap.tar.Z** file) of all files in the **/tmp/ibmsupt** directory tree or other named output directory. **Note:** Information not gathered with this option should be copied to the **snap** directory tree before using the **−c** flag. If a test case is needed to demonstrate the system problem, copy the test case to the **/tmp/ibmsupt/testcase** directory before compressing the **tar** file.
−D	Gathers dump and **/unix** information. The primary dump device is used. **Attention:** If **bosboot −k** was used to specify the running kernel to be other than /**unix**, the incorrect kernel will be gathered. Make sure that /**unix** is , or is linked to, the kernel in use when the dump was taken. Gathers dump and **/unix** information. The primary dump device is used.
−d*Dir*	Identifies the optional **snap** command output directory (**/tmp/ibmsupt** is the default).
−f	Gathers file system information.
−g	Gathers the output of the **lslpp −hBc** command, which is required to recreate exact operating system environments. Writes output to the **/tmp/ibmsupt/general/lslpp.hBc** file. Also collects general system information and writes the output to the **/tmp/ibmsupt/general/general.snap** file.
−G	Includes predefined Object Data Manager (ODM) files in general information collected with the **−g** flag.
−i	Gathers installation debug vital product data (VPD) information.
−k	Gathers kernel information
−l	Gathers programming language information.
−L	Gathers LVM information.
−n	Gathers Network File System (NFS) information.
−N	Suppresses the check for free space.
−o*OutputDevice*	Copies the compressed image onto diskette or tape.
−p	Gathers printer information.

−r	Removes **snap** command output from the **/tmp/ibmsupt** directory.
−s	Gathers Systems Network Architecture (SNA) information.
−S	Includes security files in general information collected with the **−g** flag.
−t	Gathers Transmission Control Protocol/Internet Protocol (TCP/IP) information.
−v*Component*	Displays the output of the commands executed by the **snap** command. Use this flag to view the specified name or group of files. **Note:** Press the Ctrl−C key sequence to interrupt the **snap** command. A prompt will return with the following options: press the Enter key to return to current operation; press the S key to stop the current operation; press the Q key to quit the **snap** command completely.

Examples

1. Enter the following command to gather all system configuration information:

   ```
   snap −a
   ```

   The output of this command is written to the **/tmp/ibmsupt** directory.

2. Enter the following command to create a **tar** image of all files contained in the **/tmp/ibmsupt** directory:

   ```
   snap −c
   ```

3. Enter the following command to gather general system configuration information, including the output of the **lslpp −hBc** command:

   ```
   snap −g −o /dev/rfd0
   ```

Output is written to the **/tmp/ibmsupt/general/lslpp.hBc** and **/tmp/ibmsupt/general/general.snap** files. This command also writes the system information to a removable diskette.

Files

/usr/sbin/snap	Contains the **snap** command.
/tmp/ibmsupt	Contains **snap** command output.
/tmp/ibmsupt/general/lslpp.hBc	Contains the output of the **lslpp −hBc** command, which is required to recreate exact operating system environments.
/tmp/ibmsupt/general/general.snap	Contains general system information that is collected with the **snap −g** command.
/tmp/ibmsupt/testcase	Contains the test case that demonstrates your system problem.

Related Information

The **crash** command, **errpt** command, **lslpp** command, **sysdumpdev** command, **sysdumpstart** command.

snmpd Command

Purpose

Starts the Simple Network Management Protocol (SNMP) daemon as a background process.

Syntax

snmpd [**−c***ConfigFile*] [**−d***Level*] [**−f***LogFile*]

Description

The **snmpd** command starts the SNMP daemon. This command may only be issued by a user with root privileges or by a member of the system group.

The SNMP daemon is a server that supports the standard Simple Network Management Protocol (SNMP) documented by RFC 1157 and the Management Information Base (MIB) as defined in RFC 1155 and RFC 1213. The SNMP daemon provides the following three functions:

- Receiving and authenticating SNMP requests from network monitors.
- Processing requests and returning results to the originating monitor.
- Sending trap notification to all hosts listed in the configuration file.

The SNMP daemon server keeps log messages in a file specified by the *LogFile* variable if the **−f** flag is used or in a log file specified in the configuration file. When the size of the log file exceeds the predefined maximum log file size, the **snmpd** command will rotate the log file by moving the old log file to another file as follows:

- LogFile.3 is deleted.
- LogFile.2 is moved to LogFile.3.
- LogFile.1 is moved to LogFile.2.
- LogFile.0 is moved to LogFile.1.
- LogFile is moved to LogFile.0.
- Logging continues in LogFile.

If logging is not directed from the **snmpd** command line with the **−f** flag, logging can be directed from the configuration file. See "Understanding the SNMP Daemon Logging Facility" in *AIX Communications Programming Concepts* for more information.

Supported set variables are:

- **sysContact**
- **sysName**
- **sysLocation**
- **ifAdminStatus**
- **atPhysAddress**
- **atNetAddress**

- **ipForwarding**
- **ipDefaultTTL**
- **ipRouteDest**
- **ipRouteNextHop**
- **ipRouteType**
- **ipNetToMediaPhysAddress**
- **ipNetToMediaNetAddress**
- **ipNetToMediaType**
- **snmpEnableAuthenTraps**
- **smuxPstatus**
- **smuxTstatus**

See "Understanding SNMP Daemon Support for SET Request Processing" in *AIX Communications Programming Concepts* for more information on the supported set variables.

The following commands should be issued before the SNMP daemon is started:

- **ifconfig lo0 loopback**
- **startsrc −s inetd**

These commands are normally executed during system startup when the **/etc/rc.net** and **/etc/rc.tcpip** shell scripts are called. (The **snmpd** command can be placed in the **/etc/rc.tcpip** shell script.)

The **snmpd** daemon should be controlled using the System Resource Controller (SRC). Entering **snmpd** at the command line is not recommended.

Manipulating the snmpd Daemon with the System Resource Controller

The **snmpd** daemon is a subsytem controlled by the System Resource Controller (SRC). The **snmpd** daemon is a member of the **tcpip** system group. The **snmpd** daemon is enabled by default and can be manipulated by SRC commands.

Use the following SRC commands to manipulate the **snmpd** daemon:

startsrc	Starts a subsystem, group of subsystems, or a subserver. Issuing the **startsrc** command causes the **snmpd** command to generate a *coldStart* trap.
stopsrc	Stops a subsystem, group of subsystems, or a subserver.
refresh	Causes a subsystem or group of subsystems to reread the appropriate configuration file. Issuing a **refresh** command causes the **snmpd** daemon to generate a *warmStart* trap.
traceson	Enables tracing of a subsystem, group of subsystems, or a subserver. If the user issuing the **traceson** command is not the root user, the debugging level will not exceed level 2.
tracesoff	Disables tracing of a subsystem, group of subsystems, or a subserver.
lssrc	Gets the status of a subsystem, group of subsystems, or a subserver. If the user issuing the long status form of the **lssrc** command is not the root user, no community name information is displayed.

Flags

−c*ConfigFile*	Specifies full path and file name of the configuration file for the **snmpd** daemon. This file is read when the **snmpd** daemon starts up and when a **refresh** or **kill −1** signal is issued. If the **−c** flag is not specified, the default configuration file is **/etc/snmpd.conf**. See the **snmpd.conf** file for information on this file format.
−d*Level*	Specifies the level of tracing the **snmpd** command produces. The *Level* value can be one of: **0** All notices, exceptions, and fatal messages

1 Level 0 plus debug messages

2 Level 1 plus a hexadecimal dump of incoming and outgoing packets

3 Level 2 plus an English version of the request and response packets

If the **−d** flag is not specified, the debugging level is set to 0.

−f*LogFile* Specifies the full path and file name into which **snmpd** tracing information is logged. If the **−f** flag is not specified, no information will be logged. See the **snmpd.conf** file for more information on setting logging parameters.

Examples

1. To start the **snmpd** daemon, enter a command similar to the following:

   ```
   startsrc −s snmpd −a "−f /tmp/snmpd.log"
   ```

   This command starts the **snmpd** daemon and logs information to the **/tmp/snmpd.log** file at debug level 0.

2. To stop the **snmpd** daemon normally, enter:

   ```
   stopsrc −s snmpd
   ```

   This command stops the daemon. The **−s** flag specifies the subsystem that follows to be stopped.

3. To get short status from the **snmpd** daemon, enter:

   ```
   lssrc −s snmpd
   ```

   This command returns the name of the daemon, the process ID of the daemon, and the state of the daemon (active or inactive).

4. To get a long status from the **snmpd** daemon, enter:

   ```
   lssrc −ls snmpd
   ```

   If you are the root user, this long form of the status report lists the configured community names and associated access privileges and views for **snmp** requests. The long form also lists the community names associated with the hosts for trap notification, logging configuration parameters, **snmpd** specific configuration parameters and **smux** configuration parameters.

5. To enable tracing for the **snmpd** daemon, enter the following:

   ```
   traceson −s snmpd
   ```

   This command enables **snmpd** debugging if the **snmpd** daemon is configured for logging.

6. To view the contents of the DHCP Server database files **/etc/dhcpsd.ar** and **/etc/dhcpsd.cr**, enter:
   ```
   lssrc −l −s dhcpsd
   ```

Files

/etc/services Contains port assignments for required services. The following entries must be present in the **/etc/services** file if the entries are not already present:

```
snmp        161/udp
snmp−trap   162/udp
smux        199/tcp
```

Notes:

1. The snmp port *must* be 161 as required by RFC 1157.
2. The snmp−trap port *must* be 162 as required by RFC 1157.
3. The smux port *must* be 199.
4. The **/etc/services** file is shipped with these entries already in place.
5. If the **/etc/services** file is being served from a server, these entries must be present in the server's **/etc/services** file.

/etc/snmpd.conf Specifies the configuration parameters for the **snmpd** agent.

Related Information

The **gated** daemon.

Understanding the SNMP Daemon, Problem Determination for the SNMP Daemon in *AIX Version 4.3 System Management Guide: Communications and Networks*.

Understanding the Simple Network Management Protocol (SNMP), and Understanding the Management Information Base (MIB) in *AIX Communications Programming Concepts*.

Example of SMUX Error Logging Subroutines in *AIX Communications Programming Concepts*.

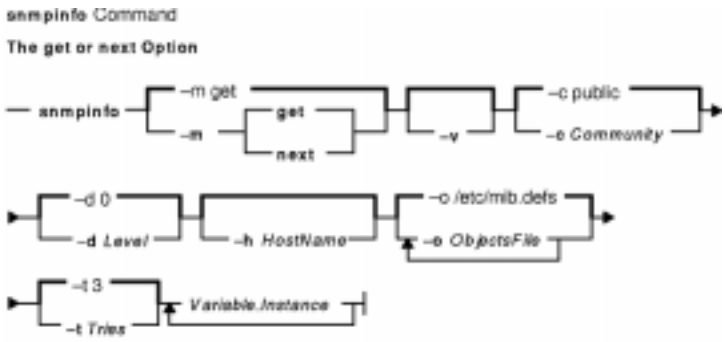
snmpinfo Command

Purpose

Requests or modifies values of Management Information Base (MIB) variables managed by a Simple Network Management Protocol (SNMP) agent.

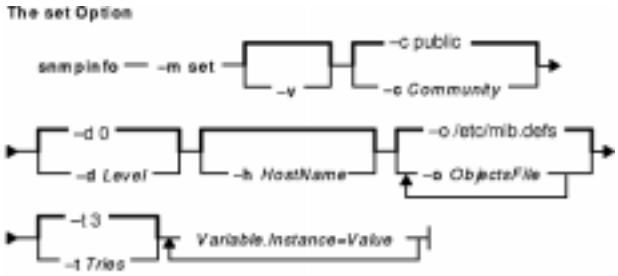
Syntax

The get or next Option



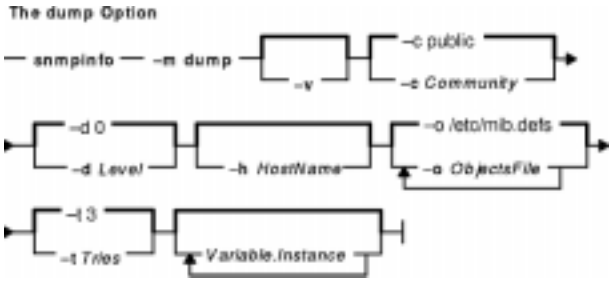
snmpinfo [**−m get** | **next**] [**−v**] [**−c** *Community*] [**−d** *Level*] [**−h** *HostName*] [**−o** *ObjectsFile*] ... [**−t** *Tries*] *Variable.Instance ...*

The set Option



snmpinfo −m set [**−v**] [**−c** *Community*] [**−d** *Level*] [**−h** *HostName*] [**−o** *ObjectsFile*] ... [**−t** *Tries*] *Variable .Instance=Value ...*

The dump Option



snmpinfo −m dump [**−v**] [**−c** *Community*] [**−d** *Level*] [**−h** *HostName*] [**−o** *ObjectsFile*] ... [**−t** *Tries*] [*Variable***.***Instance*] ...

Description

The **snmpinfo** command requests or modifies values for one or more MIB variables for an SNMP agent. This command may only be issued by a user with root privileges or by a member of the system group.

If the you specify the **get** option, the **snmpinfo** command requests information about one or more MIB variables from an SNMP agent.

If you specify the **next** option, the **snmpinfo** command requests information from an SNMP agent about the instances following the specified instances. The **next** option makes it possible to obtain MIB values without knowledge of the instance qualifiers.

If you specify the **set** option, the **snmpinfo** command modifies values for one or more MIB variables for an SNMP agent. Only a few MIB variables are designated read−write. The agent that manages the MIB database may take various actions as a side effect of modifying MIB variables. For example, setting the **ifAdminStatus** MIB variable to 2 will normally shut down a network interface. The action taken is determined by the implementation of the SNMP agent that manages the database.

If you specify the **dump** option, the **snmpinfo** command can be used to traverse the entire MIB tree of a given agent. If a group is passed in as the *Variable* parameter, the **snmpinfo** command will traverse that specified path of the MIB tree.

The **snmpinfo** command has a debug facility that will dump debug information for transmitted and received packets. The facility is enabled with the **−d** flag.

Parameters

Value Specifies the value to which the MIB *Variable* parameter is to be set. A value must be specified for each variable. If a value is not specified, the request packet will be invalid.

Variable Specifies the name in text format or numeric format of a specific MIB variable as defined in the **/etc/mib.defs** file. If the option to the **−m** flag is **next** or **dump**, the *Variable* parameter may be specified as a MIB group.

Instance Specifies the instance qualifier for the MIB *Variable* parameter. The *Instance* parameter is required if the option to the **−m** flag is **get** or **set**. The *Instance* parameter is optional if the option to the **−m** flag is **next** or **dump**.

Notes:

1. There should be no blank spaces in the *Variable.Instance* parameter sequence.
2. If the *Instance* parameter is not specified, do not place a . (dot) after the *Variable* parameter.

For further information, consult RFC 1213, which defines the Management Information Base (MIB) for network management, and RFC 1157, which defines the SNMP protocol for creating requests for MIB information and formatting responses.

Flags

−c*Community* Specifies the community name to be used to query the SNMP agent. If the **−c** flag is not specified, the default community name is **public**.

−d*Level* Specifies the level of I/O debug information. The *Level* value can be one of:

0 No debug information.

1 Port bindings and the number of bytes transmitted and received.

2 Level 1 plus a hexadecimal dump of incoming and outgoing packets.

3 Level 2 plus an English version of the request and response packets.

If the **−d** flag is not specified, the default debug level is 0.

−h*HostName* Specifies the host name of the SNMP agent to be queried. If the **−h** flag is not specified, the default host name is the host name of the machine on which the user is currently logged in.

−m*Option* Specifies the mode by which to access the MIB variables.

The *Option* value can be one of:

get Requests information about the specified MIB variables.

next Requests the instances following the specified instances.

set Modifies the specified write access MIB variables.

dump Dumps the specified section of the MIB tree.

Notes:

1. The option name can be specified by the minimum number of characters required to make it unique.
2. If the **−m** flag is not specified, the default mode is **get**.

−o*ObjectsFile* Specifies the name of the objects definition file that defines the MIB objects the **snmpinfo** command can request. If the **−o** flag is not specified, the default objects definition file name is **/etc/mib.defs**. See the **mosy** command for information on creating this file. More than one *ObjectsFile* can be referenced with the restriction that files containing parent definitions be specified before files containing child definitions.

−t *Tries* Specifies the number of times the **snmpinfo** command transmits the SNMP request to the SNMP agent before terminating with the message `no SNMP response`. If the **−t** flag is not specified, the default number of tries is 3.

−v Specifies that the output from the **snmpinfo** command be displayed in verbose mode. If the **−v** flag is not specified, the information will not be displayed in verbose mode.

Examples

1. To get the values for the MIB variable `ifDescr.1`, for the interface associated with `ifIndex.1` and `SysDescr`, enter:

   ```
   snmpinfo −m get −v sysDescr.0 ifDescr.1
   ```

   In this example, the **−m get** flag specifies that the **snmpinfo** command should retrieve the value of MIB variables `ifDescr.1`, (the interface description for the interface associated with the `ifIndex.1),` and `sysDescr.0 (`the system description of the local host).

2. To get the value for the MIB variable following the **ipAdEntIfIndex** MIB variable for the host specified by IP address 192.100.154.1, enter:

   ```
   snmpinfo −m next −v 1.3.6.1.2.1.4.20.1.2.192.100.154.1
   ```

   In this example, the **−m next** flag specifies that the **snmpinfo** command should retrieve the information for the MIB variable **ifAdEntIfIndex**`.192.100.154.1.`

3. To get the value of the first MIB variable in the system group, enter:

   ```
   snmpinfo −m next −v −h giants system
   ```

   In this example, the **−m next** flag specifies that the **snmpinfo** command should retrieve the

information for the MIB variable following the system group, which is `sysDescr.0`; the **−v** flag indicates verbose mode; the **−h** flag indicates that the agent to be queried is `giants;` the group to retrieve information from is `system`.

4. To set the value of a MIB variable, enter a command similar to the following:

   ```
   snmpinfo −m set −v −h giants −c monitor −t 2 ifAdminStatus.1=2
   ```

   In this example, the MIB **ifAdminStatus** variable is set to 2, or down, for the interface associated with `ifIndex.1` on the host known as `giants`. The **−c** flag specifies the community for the host. The **−t2** flag specifies that the **snmpinfo** command will transmit the SNMP request to the SNMP agent 2 times before terminating if no response is received from the SNMP agent.

5. To dump a group of the MIB tree in verbose mode, enter a command similar to the following:

   ```
   snmpinfo −m dump −v interfaces
   ```

   In this example the `interfaces` group is dumped in verbose mode.

6. To dump the entire MIB tree, enter:

   ```
   snmpinfo −m dump
   ```

Files

/etc/mib.defs Defines the Management Information Base (MIB) variables the SNMP agent should recognize and handle.

Related Information

The **mosy** command.

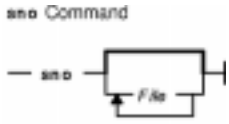
Understanding the Simple Network Management Protocol (SNMP), Using the Management Information Base (MIB) Database, and Understanding the Management Information Base (MIB) in *AIX Communications Programming Concepts*.

sno Command

Purpose

Provides a SNOBOL interpreter.

Syntax



sno [*File* ...]

Description

The **sno** command provides a SNOBOL compiler and interpreter, with some differences from standard SNOBOL. It reads the named files and the standard input and compiles all input through a statement containing the **end** label. The rest is available to the **syspit** pseudo−variable.

The **sno** command differs from standard SNOBOL in the following ways:

- There are no unanchored searches. To get the same effect, use lines similar to the following:

  `a **b` Produces an unanchored search for *b*.

  `a *x* b = x c` Produces an unanchored assignment.
- There is no back referencing.

  ```
  x = "abc"
  ```

  `a *x* x` Produces an unanchored search for **abc**.

- Function declaration is done at compile time by the use of the (non−unique) **define** label. Execution of a function call begins at the statement following the **define** label. Functions cannot be defined at run time, and the use of the name **define** is preempted. There is no provision for automatic variables other than parameters. For example:

  ```
  define f()
  define f(a, b, c)
  ```

- All labels except **define** (even **end**), must have a nonempty statement.
- Labels, functions, and variables must all have distinct names. In particular, the nonempty statement on **end** cannot merely name a label.
- If **start** is a label in the program, program execution begins there. If not, execution begins with the first executable statement. The **define** label is not an executable statement.
- There are no built−in functions.
- Parentheses for arithmetic are not needed. Normal precedence applies. Because of this, the arithmetic operators \ (backslash) and ***** (asterisk) must be set off by spaces.
- The right side of assignments must be nonempty.
- Either ' (single quotation mark) or " (double quotation mark) can be used for literal quotation marks.
- The pseudo−variable **sysppt** is not available.

Examples

To run the file `test.s` through the **sno** command and direct the output into the file output, enter:

```
sno < test.s > output
```

Files

/usr/bin/sno Contains the **sno** command.

Related Information

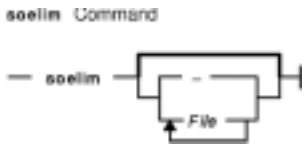
The **awk** command.

soelim Command

Purpose

Processes **.so** requests in **nroff** command files.

Syntax



soelim [*File* ... | −]

Description

The **soelim** command reads specified files or standard input and performs inclusion specified by the **nroff** command and **troff** command requests of the form `.so filename` when the request appears at the beginning of input lines. Any combination of ASCII spaces and ASCII tab characters can follow the **.so** request and precede the file name. No characters should follow the file name.

The **soelim** command is useful because commands, such as the **tbl** command, do not normally perform file inclusions during processing.

When the − (minus sign) flag is specified, a file name corresponding to standard input is included.

Flag

− Indicates a file name corresponding to standard input.

> **Note:** Inclusion can be suppressed by using a **'** (single quotation mark) instead of a **.** (period), as follows:

Parameter

File	Specifies files that the command performs inclusion on. The default is standard input.

```
'so /usr/share/lib/tmac/tmac.s
```

Example

Following is a sample usage of the **soelim** command:

```
soelim exum?.n | tbl | nroff −ms −Tlp | col −Tlp | pg
```

In this example, you use the **soelim** command to preprocess the file inclusion (**.so**) requests. The output is then passed to the **tbl** command. This makes it easier to place tables in separate files that can be included in forming a large document.

Related Information

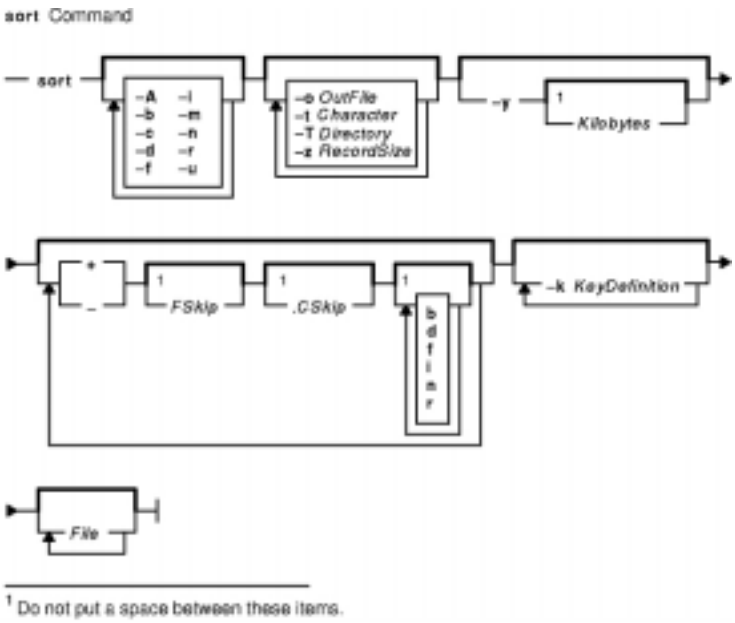
The **colcrt** command, **nroff** command, **tbl** command, **troff** command.

sort Command

Purpose

Sorts files, merges files that are already sorted, and checks files to determine if they have been sorted.

Syntax



sort [**−A**] [**−b**] [**−c**] [**−d**] [**−f**] [**−i**] [**−m**] [**−n**] [**−r**] [**−u**] [**−o** *OutFile*] [**−t** *Character*] [**−T** *Directory*] [**−y** [*Kilobytes*]] [**−z***RecordSize*] [[**+**[*FSkip*] [**.***CSkip*] [**b**] [**d**] [**f**] [**i**] [**n**] [**r**]] [**−**[*FSkip*] [**.***CSkip*] [**b**] [**d**] [**f**] [**i**] [**n**] [**r**]]] ... [**−k** *KeyDefinition*] ... [*File ...*]

Description

The **sort** command sorts lines in the files specified by the *File* parameter and writes the result to standard output. If the *File* parameter specifies more than one file, the **sort** command concatenates the files and sorts them as one file. A **−**(minus sign) in place of a file name specifies standard input. If you do not specify any file names, the command sorts standard input. An output file can be specified with the **−o** flag.

If no flags are specified, the **sort** command sorts entire lines of the input file based upon the collation order of the current locale.

Sort Keys

A sort key is a portion of an input line that is specified by a field number and a column number. Fields are parts of input lines that are separated by field separators. The default field separator is a sequence of one or more consecutive blank characters. A different field separator can be specified using the **−t** flag. The tab and the space characters are the blank characters in the C and English Language locales.

When using sort keys, the **sort** command first sorts all lines on the contents of the first sort key. Next, all the lines whose first sort keys are equal are sorted upon the contents of the second sort key, and so on. Sort keys are numbered according to the order they appear on the command line. If two lines sort equally on all sort keys, the entire lines are then compared based upon the collation order in the current locale.

When numbering columns within fields, the blank characters in a default field separator are counted as part of the following field. Leading blanks are not counted as part of the first field, and field separator characters specified by the **−t** flag are not counted as parts of fields. Leading blank characters can be ignored using the **−b** flag.

Sort keys can be defined using the following two methods:

- **−k** *KeyDefinition*
- *FSkip***.***CSkip* (obsolescent version).

Sort Key Definition Using the −k Flag

The **−k** *KeyDefinition* flag uses the following form:

−k [*FStart*[**.***CStart*]][*Modifier*][**,**[*FEnd*[**.***CEnd*]][*Modifier*]]

The sort key includes all characters beginning with the field specified by the *FStart* variable and the column specified by the *CStart* variable and ending with the field specified by the *FEnd* variable and the column specified by the *CEnd* variable. If *Fend* is not specified, the last character of the line is assumed. If *CEnd* is not specified the last character in the *FEnd* field is assumed. Any field or column number in the *KeyDefinition* variable may be omitted. The default values are:

FStart Beginning of the line
CStart First column in the field
FEnd End of the line
CEnd Last column of the field

If there is any spaces between the fields, **sort** considers them as separate fields.

The value of the *Modifier* variable can be one or more of the letters **b**, **d**, **f**, **i**, **n**, or **r**. The modifiers apply only to the field definition they are attached to and have the same effect as the flag of the same letter. The modifier letter **b** applies only to the end of the field definition to which it is attached. For example:

```
-k 3.2b,3r
```

specifies a sort key beginning in the second nonblank column of the third field and extending to the end of the third field, with the sort on this key to be done in reverse collation order. If the *FStart* variable and the *CStart* variable fall beyond the end of the line or after the *FEnd* variable and the *CEnd* variable, then the sort key is ignored.

A sort key can also be specified in the following manner:

[+[*FSkip1*] [.*CSkip1*] [*Modifier*]] [−[*FSkip2*] [.*CSkip2*] [*Modifier*]]

The *+FSkip1* variable specifies the number of fields skipped to reach the first field of the sort key and the *+CSkip* variable specifies the number of columns skipped within that field to reach the first character in the sort key. The *−FSkip* variable specifies the number of fields skipped to reach the first character *after* the sort key, and the *−CSkip* variable specifies the number of columns to skip within that field. Any of the field and column skip counts may be omitted. The defaults are:

FSkip1 Beginning of the line
CSkip1 Zero
FSkip2 End of the line
CSkip2 Zero

The modifiers specified by the *Modifier* variable are the same as in the **−k** flag key sort definition.

The field and column numbers specified by *+FSkip1***.***CSkip1* variables are generally one less than the field and column number of the sort key itself because these variables specify how many fields and columns to skip before reaching the sort key. For example:

```
+2.1b −3r
```

specifies a sort key beginning in the second nonblank column of the third field and extending to the end of the third field, with the sort on this key to be done in reverse collation order. The statement `+2.1b` specifies that two fields are skipped and then the leading blanks and one more column are skipped. If the *+FSkip1***.***CSkip1* variables fall beyond the end of the line or after the *−FSkip2***.***CSkip2* variables, then the sort key is ignored.

Note: The maximum number of fields on a line is 10.

Flags

> **Note:** A **−b**, **−d**, **−f**, **−i**, **−n**, or **−r** flag that appears before any sort key definition applies to all sort keys. None of the **−b**, **−d**, **−f**, **−i**, **−n**, or **−r** flags may appear alone after a **−k** *KeyDefinition*; if they are attached to a *KeyDefinition* variable as a modifier, they apply only to the attached sort key. If one of these flags follows a *+Fskip.Cskip* or *−Fskip.Cskip* sort key definition, the flag only applies to that sort key.

−A	Sorts on a byte−by−byte basis using ASCII collation order instead of collation in the current locale.
−b	Ignores leading spaces and tabs to find the first or last column of a field.
−c	Checks that input is sorted according to the ordering rules specified in the flags. A nonzero value is returned if the input file is not correctly sorted.
−d	Sorts using dictionary order. Only letters, digits, and spaces are considered in comparisons.
−f	Changes all lowercase letters to uppercase before comparison.
−i	Ignores all nonprinting characters during comparisons.
−k *KeyDefinition*	Specifies a sort key. The format of the *KeyDefinition* option is: [*FStart*[**.***CStart*]][*Modifier*][**,**[*FEnd*[**.***CEnd*]][*Modifier*]] The sort key includes all characters beginning with the field specified by the *FStart* variable and the column specified by the *CStart* variable and ending with the field specified by the *FEnd* variable and the column specified by the *CEnd* variable. The value of the *Modifier* variable can be **b**, **d**, **f**, **i**, **n**, or **r**. The modifiers are equivalent to the flags of the same letter.
−m	Merges multiple input files only; the input are assumed to be already sorted.
−n	Sorts numeric fields by arithmetic value. A numeric field may contain leading blanks, an optional minus sign, decimal digits, thousands−separator characters, and an optional radix character. Numeric sorting of a field containing any nonnumeric character gives unpredictable results.
−o *OutFile*	Directs output to the file specified by the *OutFile* parameter instead of standard output. The value of the *OutFile* parameter can be the same as the *File* parameter.
−r	Reverses the order of the specified sort.
−t*Character*	Specifies *Character* as the single field separator character.
−u	Suppresses all but one line in each set of lines that sort equally according to the sort keys and options.

−T *Directory* Places all temporary files that are created into the directory specified by the *Directory* parameter.

−y[*Kilobytes*] Starts the **sort** command using the number of kilobytes of main storage specified by the *Kilobytes* parameter and adds storage as needed. (If the value specified in the *Kilobytes* parameter is less than the minimum storage site or greater than the maximum, the minimum or maximum is used instead). If the **−y** flag is omitted, the **sort** command starts with the default storage size. The **−y0** flag starts with minimum storage, and the **−y** flag (with no *Kilobytes* value) starts with maximum storage. The amount of storage used by the **sort** command affects performance significantly. Sorting a small file in a large amount of storage is wasteful.

−z*RecordSize* Prevents abnormal termination if any of the lines being sorted are longer than the default buffer size. When the **−c** or **−m** flags are specified, the sorting phase is omitted and a system default buffer size is used. If sorted lines are longer than this size, sort terminates abnormally. The **−z** option specifies recording of the longest line in the sort phase so adequate buffers can be allocated in the merge phase. *RecordSize* must designate a value in bytes equal to or greater than the longest line to be merged.

Exit Status

This command returns the following exit values:

0 All input files were output successfully, or **−c** was specified and the input file was correctly sorted.

1 Under the **−c** option, the file was not ordered as specified, or if the **−c** and **−u** options were both specified, two input lines were found with equal keys.

>1 An error occurred.

Examples

1. To sort the `fruits` file with the **LC_ALL**, **LC_COLLATE**, or **LANG** environment variable set to `En_US`, enter:

   ```
   LANG=En_US sort fruits
   ```

   This command sequence displays the contents of the `fruits` file sorted in ascending lexicographic order. The characters in each column are compared one by one, including spaces, digits, and special characters. For instance, if the `fruits` file contains the text:

   ```
   banana
   orange
   Persimmon
   apple
   %%banana
   apple
   ORANGE
   ```

   the **sort** command displays:

   ```
   %%banana
   ORANGE
   Persimmon
   apple
   apple
   banana
   orange
   ```

   In the ASCII collating sequence, the % (percent sign) precedes uppercase letters, which precede

lowercase letters. If your current locale specifies a character set other than ASCII, your results may be different.

2. To sort in dictionary order, enter:

```
sort -d fruits
```

This command sequence sorts and displays the contents of the `fruits` file, comparing only letters, digits, and spaces. If the `fruits` file is the same as in example 1, then the **sort** command displays:

```
ORANGE
Persimmon
apple
apple
%%banana
banana
orange
```

The **-d** flag ignores the % (percent sign) character because it is not a letter, digit, or space, placing `%%banana` with `banana`.

3. To group lines that contain uppercase and special characters with similar lowercase lines, enter:

```
sort -d -f fruits
```

The **-d** flag ignores special characters and the **-f** flag ignores differences in case. With the **LC_ALL**, **LC_COLLATE**, or **LANG** environment variable set to `C`, the output for the `fruits` file becomes:

```
apple
apple
%%banana
banana
ORANGE
orange
Persimmon
```

4. To sort, removing duplicate lines, enter:

```
sort -d -f -u fruits
```

The **-u** flag tells the **sort** command to remove duplicate lines, making each line of the file unique. This command sequence displays:

```
apple
%%banana
orange
Persimmon
```

Not only is the duplicate `apple` removed, but `banana` and `ORANGE` as well. These are removed because the **-d** flag ignores the %% special characters and the **-f** flag ignores differences in case.

5. To sort as in example 4, removing duplicate instances unless capitalized or punctuated differently, enter:

```
sort -u +0 -d -f +0 fruits
```

Entering the `+0 -d -f` does the same type of sort that is done with `-d -f` in example 3. Then the `+0` performs another comparison to distinguish lines that are not identical. This prevents the **-u** flag from removing them.

Given the `fruits` file shown in example 1, the added `+0` distinguishes `%%banana` from `banana` and `ORANGE` from `orange`. However, the two instances of `apple` are identical, so one of them is deleted.

```
apple
%%banana
banana
ORANGE
orange
Persimmon
```

6. To specify the character that separates fields, enter:

   ```
   sort -t: +1 vegetables
   ```

   This command sequence sorts the `vegetables` file, comparing the text that follows the first colon on each line. The `+1` tells the **sort** command to ignore the first field and to compare from the start of the second field to the end of the line. The `-t:` flag tells the **sort** command that colons separate fields. If `vegetables` contains:

   ```
   yams:104
   turnips:8
   potatoes:15
   carrots:104
   green beans:32
   radishes:5
   lettuce:15
   ```

   Then, with the **LC_ALL**, **LC_COLLATE**, or **LANG** environment variable set to C, the **sort** command displays:

   ```
   carrots:104
   yams:104
   lettuce:15
   potatoes:15
   green beans:32
   radishes:5
   turnips:8
   ```

   Note that the numbers are not in numeric order. This happened when a lexicographic sort compares each character from left to right. In other words, `3` comes before `5`, so `32` comes before `5`.

7. To sort numbers, enter:

   ```
   sort -t: +1 -n vegetables
   ```

   This command sequence sorts the `vegetables` file numerically on the second field. If the `vegetables` file is the same as in example 6, then the **sort** command displays:

   ```
   radishes:5
   turnips:8
   lettuce:15
   potatoes:15
   green beans:32
   carrots:104
   yams:104
   ```

8. To sort more than one field, enter:

   ```
   sort -t: +1 -2 -n +0 -1 -r vegetables
   ```

OR

```
sort -t: -k2,2n -k1,1r vegetables
```

This command sequence performs a numeric sort on the second field (`+1 -2 -n`). Within that ordering, it sorts the first field in reverse alphabetic order (`+0 -1 -r`). With the **LC_ALL**, **LC_COLLATE**, or **LANG** environment variable set to C, the output looks like this:

```
radishes:5
turnips:8
potatoes:15
lettuce:15
green beans:32
yams:104
carrots:104
```

The command sorts the lines in numeric order. When two lines have the same number, they appear in reverse alphabetic order.

9. To replace the original file with the sorted text, enter:

```
sort -o vegetables vegetables
```

This command sequence stores the sorted output into the `vegetables` file (`-o vegetables`).

Files

/usr/bin/sort Contains the **sort** command.

/var/tmp Temporary space during the **sort** command processing.

/usr/tmp Temporary space during the **sort** command processing, if file cannot be created in **/var/tmp**.

/tmp Temporary space during the **sort** command processing, if file cannot be created in */var/tmp* or **/usr/tmp**.

Related Information

The **comm** command, **join** command, and **uniq** command.

Files Overview, Input and Output Redirection Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

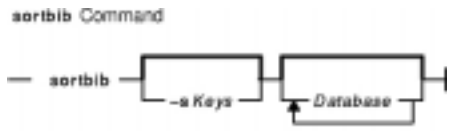
National Language Support Overview for Programming in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

sortbib Command

Purpose

Sorts a bibliographic database.

Syntax



sortbib [**−s***Keys*] [*Database ...*]

Description

The **sortbib** command sorts files of records containing **refer** command key letters by user−specified keys. The records can be separated by blank lines, or enclosed by the .[(period, left bracket) and the .] (period, right bracket) delimiters, but the two styles cannot be mixed together. The **sortbib** command reads through each database specified by the *Database* parameter and pulls out key fields, which are sorted separately. The sorted key fields contain the file pointer, byte offset, and length of corresponding records. These records are delivered using disk seeks and reads, so the **sortbib** command cannot be used in a pipeline to read standard input.

By default, the **sortbib** command alphabetizes by the first `%A` and `%D` fields, which contain the senior author and date.

The **sortbib** command sorts by the last word in the `%A` field, which is assumed to be the author's last name. A word in the final position, such as `jr.` or `ed.`, is ignored if the name preceding ends with a comma. Authors with two−word last names, or names with uncommon constructions, can be sorted correctly by using the **nroff** command convention **\0** in place of a space character. Specifying the `%Q` field is similar to the `%A` field, except sorting begins with the first, not the last, word.

> **Note:** Records with missing author fields should be sorted by title.

The **sortbib** command sorts by the last word of the `%D` line, which is usually the year. It ignores leading articles when sorting by titles in the `%T` or `%J` fields. The articles ignored are specific to the locale and specified in the locale−specific **refer**message catalog. Within this catalog, the articles are contained in a single message. Each article is separated by any number of ASCII space or tab characters. If a sort−significant field is absent from a record, the **sortbib** command places the record before other records containing that field.

No more than 16 databases can be sorted together at one time. Records longer than 4096 characters are truncated.

The *Database* parameter contains **refer** command key letters by user−specified keys that the **sortbib** command sorts through.

Flags

−s*Keys* Specifies field keys to sort on.

Examples

1. To sorts by author, title, and date:

   ```
   sortbib −sATD Database
   ```

2. To sort by author and date:

   ```
   sortbib −sA+D Database
   ```

Files

/tmp/Sbib*XXXXX* Contains the temporary file.
/usr/bin/sort Contains the **sort** command.

Related Information

The **addbib** command, **indxbib** command, **lookbib** command, **refer** command, **roffbib** command, **sort** command.

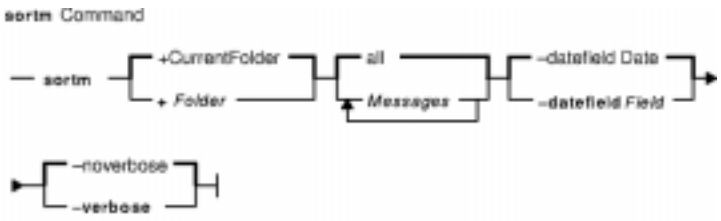
The **refer**message catalog.

sortm Command

Purpose

Sorts messages.

Syntax



sortm [+*Folder*] [*Messages*] [**−datefield** *Field*] [**−noverbose** | **−verbose**]

Description

The **sortm** command sorts messages according to their `Date:` field and renumbers them consecutively beginning with number one. Messages that are in the folder, but not specified to be sorted, are placed after the sorted messages. The **sortm** command displays a message if it cannot parse a date field.

To specify a field other than the `Date:` field, specify the **−datefield** flag. If you specify a folder, it becomes the current folder. The current message remains the current message for the specified folder, even if it moves during the sort.

Flags

−datefield *Field*	Specifies the header field to be used in the sort. The `Date:` field is the default.
+Folder	Specifies the folder with messages to be sorted. The default is the current folder.
−help	Lists the command syntax, available switches (toggles), and version information. **Note:** For Message Handler (MH), the name of this flag must be fully spelled out.
Messages	Specifies the messages to be sorted. Use the following references to specify messages:
	Number Number of the message.
	Sequence A group of messages specified by the user. Recognized values are:
	all All messages in a folder. This is the default.
	cur or **.** (period) Current message.
	first First message in a folder.
	last Last message in a folder.
	next Message following the current message.
	prev Message preceding the current message.
−noverbose	Prevents display of information during the sort. This flag is the default.
−verbose	Displays information during the sort. This information allows you to monitor the steps involved.

Profile Entries

The following entries are found in the *UserMhDirectory*/**.mh_profile** file:

`Current-Folder:`	Sets the default current folder.
`Path:`	Specifies the *UserMhDirectory*.

Examples

1. To sort all the messages in the current folder according to the date, enter:

   ```
   sortm
   ```

2. To sort messages `5` through `10` in the `easter` folder according to the date, enter:

   ```
   sortm +easter 5-10
   ```

Files

$HOME/.mh_profile	Contains the MH user profile.
/usr/bin/sortm	Contains the **sortm** command.

Related Information

The **folder** command.

The **.mh_alias** file format, **.mh_profile** file format.

Mail Overview in *AIX Version 4.3 System User's Guide: Communications and Networks*.

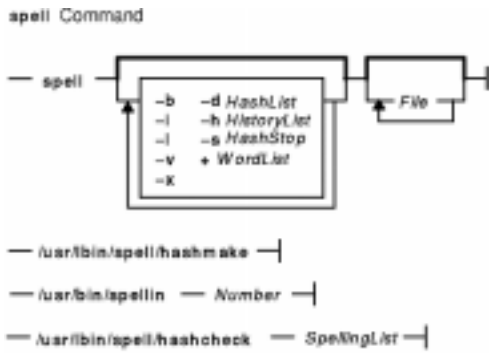
Peek, Jerry. *MH and xmh: E-mail for Users and Programmers.* Sebastopol, CA: O'Reilly & Associates, 1992.

spell Command

Purpose

Finds English Language spelling errors.

Syntax



spell [**−b**] [**−i**] [**−l**] [**−v**] [**−x**] [**−d** *HashList*] [**−h** *HistoryList*] [**−s** *HashStop*] [**+** *WordList*] [*File ...*]

Description

The **spell** command reads words in the file indicated by the *File* variable and compares them to those in a spelling list. Words that cannot be matched in the spelling list or derived from words in the spelling list (by applying certain inflections, prefixes, and suffixes) are written to standard output. If no file name is specified, the **spell** command reads standard input.

The **spell** command ignores the same **troff**, **tbl**, and **eqn** codes as does the **deroff** command.

The coverage of the spelling list is uneven. You should create your own dictionary of special words used in your files. Your dictionary is a file containing a sorted list of words, one per line. To create your dictionary, use the **spellin** command.

Files containing an alternate spelling list, history list, and stop list can be specified by file name parameters following the **−d**, **−f**, and **−h** flags. Copies of all output can be accumulated in the history file.

Three programs help maintain and check the hash lists used by the **spell** command:

/usr/lbin/spell/hashmake	Reads a list of words from standard input and writes the corresponding 9−digit hash code to standard output.
/usr/bin/spellin *Number*	Reads the specified *Number* of hash codes from standard input and writes a compressed spelling list to standard output.
/usr/lbin/spell/hashcheck *SpellingList*	Reads a compressed *SpellingList*, recreates the 9−digit hash codes for all the words in it, and writes these codes to standard output.

The *File* parameter specifies the files that the **spell** command reads and compares them with the spelling list. If no file is specified, the command reads standard input.

Flags

−b	Checks British spelling. However, this flag does not provide a reasonable prototype for British spelling. The algorithms to derive a match against the spelling dictionary by applying certain inflections, prefixes, and suffixes are based on American English spelling.
−d*HashList*	Specifies the *HashList* file as the alternative spelling list. The default is **/usr/share/dict/hlist[ab]**.
−h*HistoryList*	Specifies the *HistoryList* file as the alternative history list, which is used to accumulate all output. The default is **/usr/lbin/spell/spellhist**. **Note:** The *HistoryList* file must be an existing file with read and write permissions.
−i	Suppresses processing of include files.
−l	Follows the chain of all include files (**.so** and **.nx** formatting commands). Without this flag, the **spell** command follows chains of all include files except for those beginning with **/usr/lib**.
−s*HashStop*	Specifies the *HashStop* file as the alternative stop list, which is used to filter out misspellings that would otherwise pass. The default is **/usr/share/dict/hstop**.
−v	Displays all words not in the spelling list and indicates possible derivations from the words.
−x	Displays every possible word stem with an = (equal sign).
+*WordList*	Checks *WordList* for additional word spellings. *WordList* is the name of a file you provide that contains a sorted list of words, one per line. With this flag, you can specify a set of correctly spelled words (in addition to the **spell** command's own spelling list) for each job.

Exit Status

The following exit values are returned:

0 Indicates successful completion.

>0 Indicates an error occurred.

Examples

1. To check your spelling, enter:

   ```
   spell chap1 >mistakes
   ```

   This creates a file named `mistakes` containing all the words found in `chap1` that are not in the system spelling dictionary. Some of these may be correctly spelled words that the **spell** command does not recognize. Save the output of the **spell** command in a file because the word list may be long.

2. To check British spelling, enter:

   ```
   spell −b chap1 >mistakes
   ```

   This checks `chap1` against the British dictionary and writes the questionable words in the `mistakes` file.

3. To see how the **spell** command derives words, enter:

   ```
   spell −v chap1 >deriv
   ```

   This lists words not found literally in the dictionary but are derived from forms of dictionary words. The prefixes and suffixes used to form the derivations are indicated for each word. Words that are not in the dictionary at all are also listed.

4. To check your spelling against an additional word list, enter:

```
spell +newwords chap1
```

This checks the spelling of words in `chap1` against the system dictionary and against `newwords`. The `newwords` file lists words in alphabetical order, one per line. You can create this file with a text editor, such as the ed editor, and alphabetize it with the **sort** command.

Files

/usr/share/dict/hlist[ab]	Contains hashed spelling lists, both American and British.
/usr/share/dict/hstop	Contains a hashed stop list.
/usr/lbin/spell/spellhist	Contains a history file.
/usr/lbin/spell/compress	Contains an executable shell program to compress the history file.
/usr/lbin/spell/hashmake	Creates hash codes from a spelling list.
/usr/bin/spellin*Number*	Creates spelling list from hash codes.
/usr/lbin/spell/hashcheck*SpellingList*	Creates hash codes from a compressed spelling list.
/usr/lbin/spell/spellinprg	Main program called by the **spellin** file.
/usr/lbin/spell/spellprog	Checks spelling.

Related Information

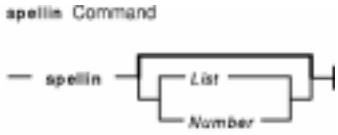
The **deroff** command, **eqn** command, **neqn** command, **sed** command, **sort** command, **spellin** command, **spellout** command, **tbl** command, **tee** command, and **troff** command.

spellin Command

Purpose

Creates a spelling list.

Syntax



spellin [*List* | *Number*]

Description

The **spellin** command creates a spelling list for use by the **spell** command. The parameter for the **spellin** command can be a file name or a number. The **spellin** command combines the words from the standard input and the already existing spelling list file and places a new spelling list on the standard output. If no list file is specified, a new list is created. If *Number* is specified, the **spellin** command reads the specified number of hash codes from standard input and writes a compressed spelling list.

Examples

To add the word `hookey` to the spelling list named `myhlist`, enter:

```
echo hookey | spellin /usr/share/dict/hlista > myhlist
```

Related Information

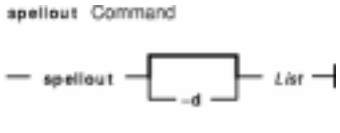
The **spell** command.

spellout Command

Purpose

Verifies that a word is not in the spelling list.

Syntax



spellout [**−d**] *List*

Description

The **spellout** command looks up each word from standard input and prints on standard output those that are missing from the hashed list file specified by the *List* parameter. The hashed list file is similar to the dictionary file used by the **spell** command.

Flags

−d Prints those words that are present in the hashed list file.

Examples

To verify that the word `hookey` is not on the default spelling list, enter:

```
echo hookey | spellout /usr/share/dict/hlista
```

In this example, the **spellout** command prints the word `hookey` on standard output if it is not in the hashed list file. With the **−d** flag, **spellout** prints the word `hookey` if it is found in the hash file.

Related Information

The **spell** command, **spellin** command.

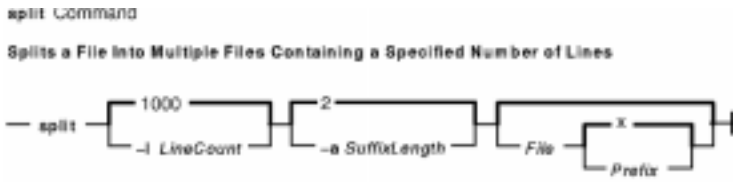
split Command

Purpose

Splits a file into pieces.

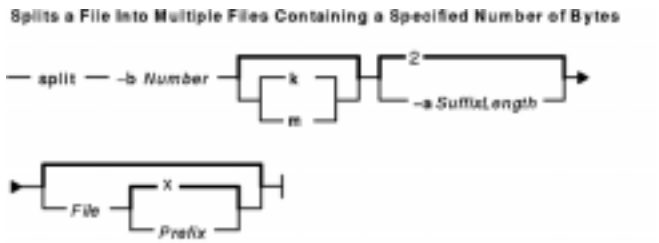
Syntax

To Split a File Into Multiple Files Containing a Specified Number of Lines



split [**−l***LineCount*] [**−a***SuffixLength*] [*File* [*Prefix*]]

To Split a File Into Multiple Files Containing a Specified Number of Bytes



split −b*Number* [**k** | **m**] [**−a***SuffixLength*] [*File* [*Prefix*]]

Description

The **split** command reads the specified file and writes it in 1000−line pieces to a set of output files. The name of the first output file is constructed by combining the specified prefix (*x* by default) with the *aa* suffix, the second by combining the prefix with the *ab* suffix, and so on lexicographically through *zz* (a maximum of 676 files). The number of letters in the suffix, and consequently the number of output name files, can be increased by using the **−a** flag.

You cannot specify a *Prefix* longer than **PATH_MAX** − 2 bytes (or **PATH_MAX** − *SuffixLength* bytes if the **−a** flag is specified). The **PATH_MAX** variable specifies the maximum path−name length for the system as defined in the **/usr/include/sys/limits.h** file.

If you do not specify an input file or if you specify a file name of − (minus sign), the **split** command reads standard input.

Flags

Note: The **−b** and **−l** flags are mutually exclusive.

−a*SuffixLength* Specifies the number of letters to use in forming the suffix portion of the output name files. The number of letters determines the number of possible output filename combinations. The default is two letters.

−b*Number*	Splits the file into the number of bytes specified by the *Number* variable. Adding the *k* (kilobyte) or *m* (megabyte) multipliers to the end of the *Number* value causes the file to be split into *Number**1024 or *Number**1,048,576 byte pieces, respectively.
−l*LineCount*	Specifies the number of lines in each output file. The default is 1000 lines.

Exit Status

This command returns the following exit values:

0 The command ran successfully.
>0 An error occurred.

Examples

1. To split a file into 1000−line segments, enter:

   ```
   split book
   ```

   This example splits `book` into 1000−line segments named `xaa`, `xab`, `xac`, and so forth.

2. To split a file into 50−line segments and specify the file−name prefix, enter:

   ```
   split −l 50 book sect
   ```

   This example splits `book` into 50−line segments named `sectaa`, `sectab`, `sectac`, and so forth.

3. To split a file into 2KB segments, enter:

   ```
   split −b 2k book
   ```

   This example splits the book into 2*1024−byte segments named `xaa`, `xab`, `xac`, and so forth.

4. To split a file into more than 676 segments, enter:

   ```
   split −l 5 −a 3 book sect
   ```

This example splits a book into 5−line segments named `sectaaa`, `sectaab`, `sectaac`, and so forth, up to `sectzzz` (a maximum of 17,576 files).

Files

/usr/bin/split Contains the **split** command.

Related Information

The **csplit** command.

Files Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

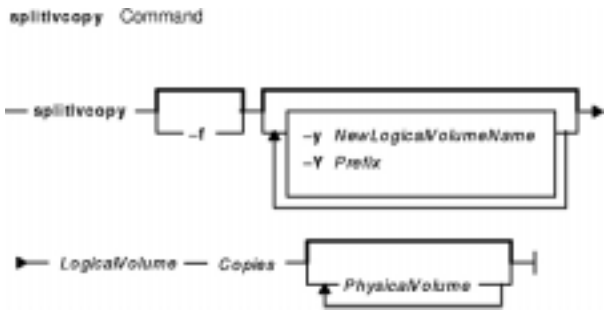
Input and Output Redirection Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

splitlvcopy Command

Purpose

Splits copies from one logical volume and creates a new logical volume from them.

Syntax



splitlvcopy [**−f**] [**−y** *NewLogicalVolumeName*] [**−Y** *Prefix*]
LogicalVolumeCopies [*PhysicalVolume ...*]

Description

> **Note:** To use this command, you must either have root user authority or be a member of the system group.
>
> **Attention:** Although the **splitlvcopy** command can split logical volumes that are open, including logical volumes containing mounted filesystems, this is not recommended. You may lose consistency between *LogicalVolume* and *NewLogicalVolume* if the logical volume is accessed by multiple processes simultaneously. When splitting an open logical volume, you implicitly accept the risk of potential data loss and data corruption associated with this action. To avoid the potential corruption window, close logical volumes before splitting and unmount filesystems before splitting

The **splitlvcopy** command removes copies from each logical partition in *LogicalVolume* and uses them to create *NewLogicalVolume*. The *Copies* parameter determines the maximum number of physical partitions that remain in *LogicalVolume* after the split. Therefore, if *LogicalVolume* has 3 copies before the split, and the *Copies* parameter is 2, *LogicalVolume* will have 2 copies after the split and *NewLogicalVolume* will have 1 copy. You can not split a logical volume so that the total number of copies in *LogicalVolume* and *NewLogicalVolume* after the split is greater than the number of copies in *LogicalVolume* before the split.

The *NewLogicalVolume* will have all the same logical volume characteristics as *LogicalVolume*. If *LogicalVolume* does not have a logical volume control block the command will succeed with a warning message and creates *NewLogicalVolume* without a logical volume control block.

There are additional considerations to take when splitting a logical volume containing a filesystem. After the split there will be two logical volumes but there will only be one entry in the **/etc/filesystems** file which refers to *LogicalVolume*. To access *NewLogicalVolume* as a filesystem you must create an additional entry in **/etc/filesystems** with a different mount point which refers to *NewLogicalVolume*. If the mount point does not already exist, you have to create it before the new filesystem can be mounted. In addition, if *NewLogicalVolume* was created while *LogicalVolume* was open, you have to run the command

```
fsck /dev/NewLogicalVolume
```

before the new filesystem can be mounted.

You can not use the System Management Interface Tool (SMIT) to run this command. Message catalogs are not supported for this command and therefore the error messages are provided in English only with no message catalog numbers. Documentation for splitlvcopy consists of this man page.

Flags

−f	Specifies to split open logical volumes without requesting confirmation. By default, **splitlvcopy** requests confirmation before splitting an open logical volume. This includes open raw logical volumes and logical volumes containing mounted filesystems.
−y *NewLogicalVolumeName*	Specifies the name of the new logical volume to move copies to from *LogicalVolume*.
−Y *Prefix*	Specifies the *Prefix* to use instead of the prefix in a system−generated name for the new logical volume. The prefix must be less than or equal to 13 characters. A name cannot begin with a prefix already defined in the **PdDv** class in the Device Configuration Database for other devices, nor be a name already used by another device.

Parameters

Copies	Specifies the maximum number of physical partitions that remain in LogicalVolume after the split.
LogicalVolume	Specifies the logical volume name or logical volume ID to split.
PhysicalVolume	Specifies the physical volume name or the physical volume ID to remove copies from.

Exit Status

This command returns the following exit values:

0 Successful completion.
>0 An error occurred.

Security

Access Control: You must have root authority to run this command or be a member of the system group.

Auditing Events: N/A

Examples

To split one copy of each logical partition belonging to logical volume named **oldlv** which currenttly has 3 copies of each logical partition, and create the logical volume **newlv**, enter:

```
splitlvcopy −y newlv oldlv 2
```

Each logical partition in the logical volume **oldlv** now has two physical partitions. Each logical partition in the logical volume **newlv** now has one physical partition.

Files

/etc/splitlvcopy	Contains the **splitlvcopy** command.
/tmp	Contains the temporary files created while the **splitlvcopy** command is running.

Related Information

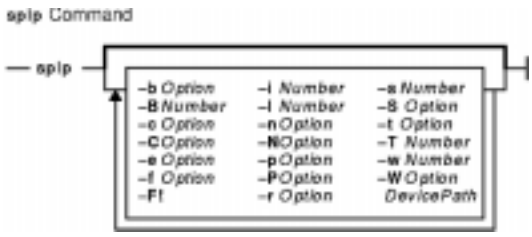
Commands: **rmlvcopy** and **mklv**.

splp Command

Purpose

Changes or displays printer driver settings.

Syntax



splp [**−b** *Option*] [**−B** *Number*] [**−c** *Option*] [**−C** *Option*] [**−e** *Option*] [**−f** *Option*] [**−F!**] [**−i** *Number*] [**−l** *Number*] [**−n** *Option*] [**−N** *Option*] [**−p** *Option*] [**−P** *Option*] [**−r** *Option*] [**−s** *Number*] [**−S** *Option*] [**−t** *Option*] [**−T** *Number*] [**−w** *Number*] [**−W** *Option*] [*DevicePath*]

Description

The **splp** command changes or displays settings for a printer device driver. The default device path is **/dev/lp0**; all flags are optional. If the device path does not begin with a **/** (backslash) character, the **/dev** directory is assumed. Also, if no flags are specified, the **splp** command reports the current settings for the specified device path. To change the current settings, specify the appropriate flags. No other processing is done, and there is no other output.

The changes that the **splp** command makes remain in effect until the next time you restart the system or rerun the **splp** command. The **splp** command can be run from the **/etc/inittab** command file to configure your printer each time you start up the system.

> **Note:** The **splp** command settings for the **−b**, **−c**, **−C**, **−f**, **−i**, **−l**, **−n**, **−p**, **−r**, **−t**, **−w**, and **−W** flags apply only when data is sent directly to the printer device (for example, redirecting the output of the **cat** command directly to the specifies device path). When files are queued for printing with the **enq**, **qprt**, **lp**, or **lpr** commands, the settings for these flags are ignored and are not changed.

Flags

−b*Option* Specifies whether backspaces are sent to the printer:
- **+** Specifies backspaces be sent to the printer.
- **!** Specifies backspaces be discarded.

−B *Number* Sets the speed to the specified number of bits per second. Values for the *Number* variable are 50, 75, 110, 134, 150, 300, 600, 1200, 1800, 2400, 4800, 9600, 19,200, and 38,400.

−c*Option* Specifies whether carriage returns are sent to the printer:
- **+** Sends carriage returns to the printer.
- **!** Translates carriage returns to line feeds.

−C*Option* Specifies whether all lowercase characters are converted to uppercase characters:

+ Converts lowercase characters to uppercase characters.

! Does not convert lowercase characters to uppercase characters.

−e*Option* Specifies the processing to be performed when an error is detected:

+ Returns an error.

! Waits until error clears.

−f*Option* Specifies whether the printer is sent form feeds or simulates a form feed with line feeds or carriage returns:

+ Sends form feeds to the printer.

! Simulates a form feed with line feeds or carriage returns.

−F! Resets font status indicators for an 3812 Page Printer or an 3816 Page Printer. This flag causes fonts to be reloaded from the printer's font diskette into the printer's memory by the next spooled print job. This flag should be specified if the printer has been turned off and then turned back on, or if the fonts in the printer's memory have become corrupted.

−i *Number* Indents the specified number of columns, where the value of the *Number* variable is an integer.

−l *Number* Prints the specified number of lines per page, where the value of the *Number* variable is an integer.

−n*Option* Specifies whether the printer is sent line feeds or translates line feeds to carriage returns:

+ Sends line feeds to the printer.

! Translates line feeds to carriage returns.

−N*Option* Specifies whether parity generation and detection is enabled:

+ Enables parity generation and detection.

! Disables parity generation and detection.

−p*Option* Specifies whether the system sends all characters to the printer unmodified or translates characters according to the settings for the **−b**, **−c**, **−C**, **−f**, **−i**, **−l**, **−n**, **−r**, **−t**, **−w**, and **−W** flags:

+ Sends all characters to the printer unmodified, overriding other settings.

! Translates characters according to the settings.

−P*Option* Specifies the parity:

+ Specifies odd parity.

! Specifies even parity.

−r *Option* Specifies whether carriage returns are added after line feeds:

+ Sends a carriage return after a line feed.

! Does not send a carriage return after a line feed.

−s *Number* Selects character size where the *Number* variable is the number of bits. Values for the *Number* variable can be 5, 6, 7, or 8. See the **termio.h** special file for additional information on character size.

−S*Option* Specifies the number of stop bits per character:

+ 2 stop bits per character.

! 1 stop bit per character.

−t*Option* Specifies whether tabs are to be expanded:

+ Does not expand tabs.

! Expands tabs on 8 position boundaries.

−T*Number* Sets the time−out period to the number of seconds specified by the *Number* variable. The value of the *Number* variable must be an integer.

−w *Number* Prints the number of columns specified by the *Number* variable. The value of the *Number* variable must be an integer.

−W*Option* Specifies whether to wrap characters beyond the specified width to the next line and print **. . .** (3 dots) after the new−line character:

+ Wraps characters beyond the specified width to the next line and prints `...` (3 dots) after the new−line character.

! Truncates characters beyond the specified width.

Examples

1. To display the current printer settings for the **/dev/lp0** printer, enter:

   ```
   splp
   ```

2. To change the printer settings, enter:

   ```
   splp  −w 80 −W + −C +
   ```

This changes the settings of the **/dev/lp0** printer for 80−column paper (the **−w80** flag). It also wraps each line that is more than 80 columns wide onto a second line (the **−W+** flag), and prints all alphabetic characters in uppercase (the **−C+** flag).

Files

/dev/lp*	Contains the printer attribute file.
/etc/inittab	Contains the printer configuration command file.

Related Information

The **cat** command, **enq** command, **lp** command, **lpr** command, **qprt** command.

The **termio.h** file.

Printer Overview for System Management in the *AIX Version 4.3 Guide to Printers and Printing*.

Adding a Printer Using the Printer Colon File in the *AIX Version 4.3 Guide to Printers and Printing*.

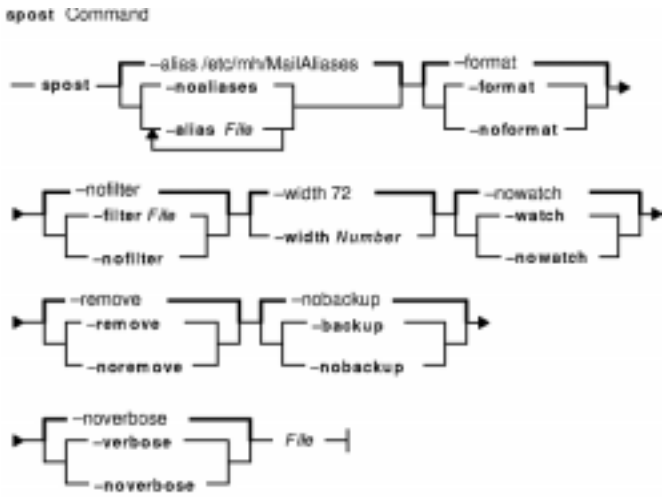
Virtual Printer Definitions and Attributes in the *AIX Version 4.3 Guide to Printers and Printing*.

spost Command

Purpose

Routes a message.

Syntax



spost [**−noalias** | **−alias***File* ...] [**−format** | **−noformat**] [**−filter***File* | **−nofilter**] [**−width***Number*] [**−watch** | **−nowatch**] [**−remove** | **−noremove**] [**−backup** | **−nobackup**] [**−verbose** | **−noverbose**]*File*

Description

The **spost** command routes messages to the correct destinations. The **spost** command is not started by the user. The **spost** command is called by other programs only.

The **spost** command searches all components of a message that specify a recipient's address and parses each address to check for proper format. The **spost** command then puts addresses in the standard format and starts the **sendmail** command. The **spost** command performs a function similar to the **post** command, but it does less address formatting than the **post** command.

The **spost** command is the default (over the **post** command). Change the default by setting the **postproc** variable in your **.mh_profile**. For example:

```
postproc: /usr/lib/mh/post
```

The *File* parameter is the name of the file to be posted.

Flags

−alias *File*	Searches the specified mail alias file for addresses. You can repeat this flag to specify multiple mail alias files. The **spost** command automatically searches the **/etc/mh/MailAliases** file.
−backup	Renames the message file by placing a , (comma) before the file name after the **spost** command successfully posts the message.
−filter *File*	Uses the header components in the specified file to copy messages sent to the `Bcc:` field

recipients.

−format Puts all recipient addresses in a standard format for the delivery transport system. This flag is the default.

−help Lists the command syntax, available switches (toggles), and version information.
Note: For Message Handler (MH), the name of this flag must be fully spelled out.

−noalias Does not use any alias files for delivering the message.

−nobackup Does not rename the message after posting the file. This flag is the default.

−nofilter Strips the `Bcc:` field header from the message and sends it to recipients specified in the `Bcc:` component. This flag is the default.

−noformat Does not alter the format of the recipient addresses.

−noremove Does not remove the temporary message file after posting the message.

−noverbose Does not display information during the delivery of the message to the **sendmail** command. This flag is the default.

−nowatch Does not display information during delivery by the **sendmail** command. This flag is the default.

−remove Removes the temporary message file after the message has been successfully posted. This flag is the default.

−verbose Displays information during the delivery of the message to the **sendmail** command. This information allows you to monitor the steps involved.

−watch Displays information during the delivery of the message by the **sendmail** command. This information allows you to monitor the steps involved.

−width *Number* Sets the width of components that contain addresses. The default is 72 columns.

Files

$HOME/.mh_profile Contains the Message Handler (MH) user profile.

/tmp/pst*Number* Contains the temporary message file.

/etc/mh/MailAliases Contains the default mail aliases.

/usr/lib/mh/.mh_profile Contains the Message Handler (MH) user profile.

Related Information

The **ali** command, **conflict** command, **mhmail** command, **post** command, **send** command, **sendmail** command, and **whom** command.

The **.mh_alias** file format.

Mail Overview in *AIX Version 4.3 System User's Guide: Communications and Networks*.

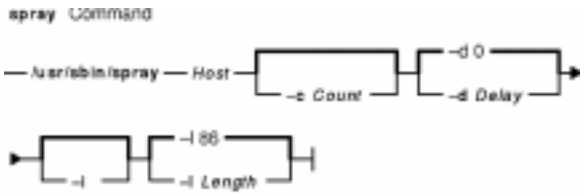
Peek, Jerry. *MH and xmh: E−mail for Users and Programmers.* Sebastopol, CA: O'Reilly & Associates, 1992.

spray Command

Purpose

Sends a specified number of packets to a host and reports performance statistics.

Syntax



/usr/sbin/spray *Host* [**−c***Count*] [**−d***Delay*] [**−i**] [**−l** *Length*]

Description

The **spray** command uses the Remote Procedure Call (RPC) protocol to send a one−way stream of packets to the host you specify. This command reports how many packets were received and at what transfer rate. The *Host* parameter can be either a name or an Internet address. The host only responds if the **sprayd** daemon is running.

See the **rpc.sprayd** daemon documentation for factors that affect **spray** command performance.

Flags

−c*Count* Specifies the number of packets to send. The default value is the number of packets required to make the total stream size 100,000 bytes.

−d*Delay* Specifies the time, in microseconds, the system pauses between sending each packet. The default is 0.

−i Uses the Internet Control Message Protocol (ICMP) echo packets rather than the RPC protocol. Since ICMP echoes automatically, it creates a two−way stream. You must be root user to use this option.

−l*Length* Specifies the number of bytes in the packet that holds the RPC call message. The default value of the *Length* parameter is 86 bytes, the size of the RPC and UDP headers.

The data in the packet is encoded using eXternal Data Representation (XDR). Since XDR deals only with 32−bit quantities, the **spray** command rounds smaller values up to the nearest possible value.

When the *Length* parameter is greater than 1500 for Ethernet or 1568 for token−ring, the RPC call can no longer fit into one Ethernet packet. Therefore, the *Length* field no longer has a simple correspondence to Ethernet packet size.

Examples

1. When sending a **spray** command to a workstation, specify the number of packets to send and the length of time the system will wait between sending each packet as follows:

```
/usr/sbin/spray zorro −c 1200 −d 2
```

In this example, the `spray` command sends `1200` packets at intervals of `2` microseconds to the workstation named `zorro`.

2. To change the number of bytes in the packets you send, enter:

```
/usr/sbin/spray zorro −l 1350
```

In this example, the `spray` command sends `1350`−byte packets to the workstation named `zorro`.

3. To send echo packets using the ICMP protocol instead of the RPC protocol, enter:

```
/usr/sbin/spray zorro −i
```

In this example, the `spray` command sends echo packets to the workstation named `zorro`.

Related Information

The **sprayd** daemon.

List of NFS Commands.

Network File System (NFS) Overview for System Management in *AIX Version 4.3 System Management Guide: Communications and Networks*.

NFS Problem Determination in *AIX Version 4.3 System Management Guide: Communications and Networks*.

sprayd Daemon

Purpose

Receives packets sent by the **spray** command.

Syntax

/usr/lib/netsvc/spray/rpc.sprayd

Description

The **rpc.sprayd** daemon is a server that records the packets sent by the **spray** command. The **rpc.sprayd** daemon is normally started by the **inetd** daemon.

UDP Performance

User Datagram Protocol (UDP) performance with the **spray** command and the **rpc.sprayd** daemon can be affected by the following factors:

- How memory buffers (mbufs) are tuned for system configuration.
- The incoming burst rate (that is, interframe gap) of UDP packets for the **spray** command.
- Other system activity. Since the **rpc.sprayd** daemon runs as a normal user process, other activity (such as the **init** process, or the **syncd** daemon) can affect the operation of the **rpc.sprayd** daemon.
- Priority of the **rpc.sprayd** daemon process. The **rpc.sprayd** daemon has a floating process priority that is calculated dynamically.
- The size of the receive socket buffer used by the **rpc.sprayd** daemon. Because various implementations use different socket buffer sizes, measuring UDP performance with the **spray** command and the **rpc.sprayd** daemon is difficult and inconclusive.

Files

/etc/inetd.conf TCP/IP configuration file that starts RPC daemons and other TCP/IP daemons.

Related Information

The **spray** command.

The **inetd** daemon.

List of NFS Commands.

Network File System (NFS) Overview for System Management in *AIX Version 4.3 System Management Guide: Communications and Networks*.

srcmstr Daemon

Purpose

Starts the System Resource Controller.

Syntax

srcmstr /usr/sbin/srcmstr [**−r**] [**−B**]

Description

The **srcmstr** daemon is the System Resource Controller (SRC). The **srcmstr** daemon spawns and controls subsystems, handles short subsystem status requests, passes requests on to a subsystem, and handles error notification.

The **srcmstr** daemon is normally started by using an **inittab** file entry.

Flags

−r Accepts remote requests if the daemon is started with the **−r** flag. If you start **srcmstr** without the **−r** flag, remote requests are ignored.

−B Specifies the **−B** flag that causes the **srcmstr** daemon to run as in previous releases (AIX 4.3.1 and earlier).

Notes:

- The **srcmstr** daemon is typically started from **inittab**. To add the **−r** or **−B** flags, edit **/etc/inittab** and run **init q** or reboot.
- The user must be running as root on the remote system. The local **/etc/hosts.equiv** file or the **/.rhosts** file must be configured to allow remote requests.

Security

Auditing Events: If the auditing subsystem has been properly configured and is enabled, the **srcmstr** command will generate the following audit record (event) every time the command is executed:

Event	Information
SRC_Start	Lists in an audit log the name of the subsystems being started.
SRC_Stop	Lists in an audit log the name of the subsystems being stopped.

See "Setting Up Auditing" in *AIX Version 4.3 System Management Guide: Operating System and Devices* for more details about how to properly select and group audit events, and how to configure audit event data collection.

Error Recovery

The default **/etc/inittab** specifies the **respawn** flag for the **srcmstr** daemon. If the **srcmstr** daemon terminates abnormally and the **/etc/inittab** specifies the **respawn** flag, the **srcmstr** daemon is restarted. It then determines which SRC subsystems were active during the previous invocation. The daemon re−establishes communication with these subsystems (if it existed previously), and initializes a private kernel extension and the **srcd** daemon to monitor the subsystem processes.

If a subsystem known to the previous invocation of **srcmstr** terminates, the SRC kernel extension notifies the **srcd** daemon. The **srcd** daemon sends a socket message to **srcmstr** and subsystem termination is handled as if the subsystem had been started by the current **srcmstr**. This function can be disabled by specifying the **−B** flag when the **srcmstr** daemon is started. The SRC kernel extension is in **/usr/lib/drivers/SRC_kex.ext**. The executable for **srcd** is **/usr/sbin/srcd**.

Files

/etc/inittab	Specifies stanzas read by the **init** command.
/etc/objrepos/SRCsubsys	Specifies the SRC Subsystem Configuration Object Class.
/etc/objrepos/SRCnotify	Specifies the SRC Notify Method Object Class.
/etc/hosts.equiv	Specifies that no remote requests will work if the specified host name is not in the **/etc/hosts.equiv** file.
/etc/services	Defines the sockets and protocols used for Internet services.
/dev/SRC	Specifies the **AF_UNIX** socket file.
/dev/.SRC−unix	Specifies the location for temporary socket files.
/dev/.SRC−unix/SRCD	Specifies the **AF_UNIX** socket file for the **srcd** daemon.
/var/adm/SRC/active_list	Contains a list of active subsystems. **Caution:** The structure of this file is internal to SRC and is subject to change.
/var/adm/SRC/watch_list	Contains a list of subsystem processes active during the previous invocation of the **srcmstr** daemon. **Caution:** The structure of this file is internal to SRC and is subject to change.
/.rhosts	Specifies remote machines and users (root only) that are allowed to request SRC function from this machine.

Related Information

The **auditpr** command, **init** command.

The System Resource Controller Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* gives an explanation of subsystems, subservers, and the System Resource Controller.

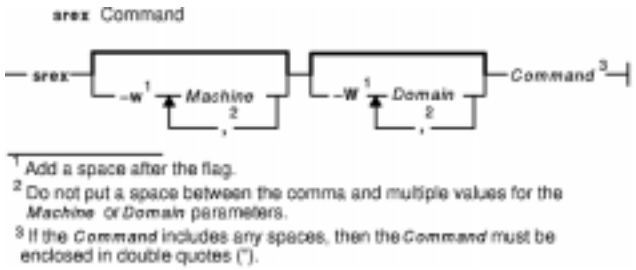
The Auditing Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

srex Command

Purpose

Starts the sequential daemon for the Distributed System Management Interface Tool (DSMIT).

Syntax



srex [**−w***Machine* [**,***Machine*] ...] [**−W***Domain* [**,***Domain*] ...] *Command*

> **Note:** Add a space after the **−w** and **−W** flags. Do not put a space between the comma and multiple values for the *Machine* or *Domain* parameters. If the *Command* includes any spaces, then the *Command* must be enclosed in double quotes (").

Description

The **srex** command allows you to bypass the DSMIT interface by entering commands at the command line. The command will be sent to all the machines specified in sequential mode. To send commands to machines specified in concurrent mode, see the **crex**command.

> **Note:** If the command contains any reserved shell characters such as a single quote ('), ampersand (&), or pipe symbol (|), use double quotes to enclose the command as in the example below. If the command contains double quotes ("), use single quotes to enclose the command.

To use the **srex** command, you must have your system configured correctly as a DSMIT server. To learn more about DSMIT and configuring your system, see the Distributed System Management Interface Tool (DSMIT) Overview, in the *Distributed SMIT 2.3 for AIX: Guide and Reference*.

Flags

−w*Machine* Specifies the machines to be in the working collective.

−W *Domain* Specifies the domains to be in the working collective.

Security

Access Control: You must be a registered DSMIT administrator to run this command.

Examples

1. To list the users on `M1` and `M2`, enter:

   ```
   srex −w M1,M2 "lsuser −c −a id home ALL | sed '/^#.*/d' | tr ':' '\011'"
   ```

2. To change all the guest passwords on Machine1 and all the machines in `D1`, enter:

   ```
   srex −w M1 −W D1 "passwd guest"
   ```

Files

/usr/share/DSMIT/domains	Contains the list of domains used by DSMIT.
/usr/share/DSMIT/dsmitos	Contains the list of operating systems of DSMIT clients.
/usr/share/DSMIT/hosts	Contains the list of machines with DSMIT installed that can run commands built by the DSMIT server.
/usr/share/DSMIT/security/v5srvtab	Stores the local machine's unique DSMIT principal key.
/usr/share/DSMIT/security/admin.cfg	Stores the DSMIT administrator's keys.
/usr/share/DSMIT/security/managing.cfg	Stores intermediate keys used by the managing systems.
/usr/share/DSMIT/security/managed.cfg	Stores the managed machine's DSMIT principal keys.
/usr/share/DSMIT/security/dsmit.ptr	Stores the name of the DSMIT configuration file server.

Related Information

The **crex** command, **dsmit** command.

Distributed System Management Interface Tool (DSMIT) Overview in the &BkSym.dsmitref;.

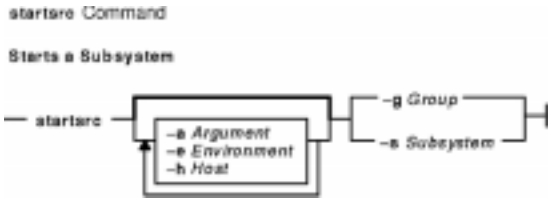
startsrc Command

Purpose

Starts a subsystem, a group of subsystems, or a subserver.

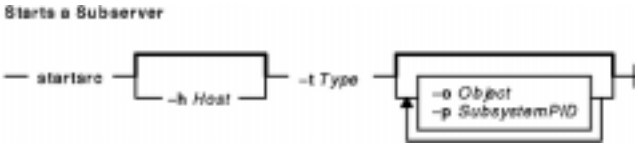
Syntax

To Start a Subsystem



startsrc [**−a***Argument*] [**−e***Environment*] [**−h***Host*] {**−s***Subsystem* |**−g***Group*}

To Start a Subserver



startsrc [**−h***Host*] **−t***Type* [**−o***Object*] [**−p***SubsystemPID*]

Description

The **startsrc** command sends the System Resource Controller (SRC) a request to start a subsystem or a group of subsystems, or to pass on a packet to the subsystem that starts a subserver.

If a start subserver request is passed to the SRC and the subsystem to which the subserver belongs is not currently active, the SRC starts the subsystem and transmits the start subserver request to the subsystem.

Flags

−a *Argument*
: Specifies an argument string that is passed to the subsystem when the subsystem is executed. This string is passed from the command line and appended to the command line arguments from the subsystem object class. The *Argument* string specified is a maximum of 1200 characters or the command is unsuccessful. The command argument is passed by the SRC to the subsystem, according to the same rules used by the shell. Quoted strings are passed as a single argument, and blanks outside a quoted string delimit an argument. Single and double quotes can be used.

−e *Environment*
: Specifies an environment string that is placed in the subsystem environment when the subsystem is executed. The *Environment* string specified is a maximum of 1200 characters, or the command is unsuccessful. Using the same rules that are used by the shell, the SRC sets up the environment for the subsystem.

  Quoted strings are assigned to a single environment variable and blanks outside quoted strings delimit each environmental variable to be set. For example: `-e "HOME=/tmp`

`TERM=dumb MESSAGE=\"Multiple word message\""`would set `HOME=/tmp` as the first, `TERM=dumb` as the second, and `MESSAGE="Multiple word message"` as the third environment variable for the subsystem.

−g *Group* Specifies a group of subsystems to be started. The command is unsuccessful if the *Group* name is not contained in the subsystem object class.

−h *Host* Specifies the foreign host on which this start action is requested. The local user must be running as "root". The remote system must be configured to accept remote System Resource Controller requests. That is, the **srcmstr** daemon (see **/etc/inittab**) must be started with the **−r** flag and the **/etc/hosts.equiv** or **.rhosts** file must be configured to allow remote requests.

−o *Object* Specifies that a subserver object is to be passed to the subsystem as a character string. It is the subsystems responsibility to determine the validity of the *Object* string.

−p *SubsystemPID* Specifies a particular instance of the subsystem to which the start subserver request is to be passed.

−s S*ubsystem* Specifies a subsystem to be started. The *Subsystem* can be the actual subsystem name or the synonym name for the subsystem. The command is unsuccessful if the *Subsystem* is not contained in the subsystem object class.

−t *Type* Specifies that a subserver is to be started. The command is unsuccessful if *Type* is not contained in the subserver object class.

Examples

1. To start a subsystem with arguments and environmental variables, enter:

   ```
   startsrc −s srctest −a "−D DEBUG" −e "TERM=dumb HOME=/tmp"
   ```

   This starts the `srctest` subsystem with `"TERM=dumb"`, `"HOME=/tmp"` in its environment and `"−D DEBUG"` as two arguments to the subsystem.

2. To start a subsystem group on a foreign host, enter:

   ```
   startsrc −g tcpip −h zork
   ```

   This starts all the subsystems in the subsystem `tcpip` group on the `zork` machine.

3. To start a subserver, enter:

   ```
   startsrc −t tester
   ```

   This sends a start subserver request to the subsystem that owns the `tester` subsystem.

4. To start a subsystem with command arguments, enter:

   ```
   startsrc −s srctest −a "−a 123 −b \"4 5 6\""
   ```

This places `"−a"` as the first argument, `"123"` as the second, `"−b"` as the third, and`"456"` as the fourth argument to the `srctest` subsystem.

Files

/etc/objrepos/SRCsubsys Specifies the SRC Subsystem Configuration Object Class.

/etc/objrepos/SRCsubsvr Specifies the SRC Subserver Configuration Object Class.

/etc/services Defines the sockets and protocols used for Internet services.

/dev/SRC Specifies the **AF_UNIX** socket file.

/dev/.SRC−unix Specifies the location for temporary socket files.

Related Information

The **stopsrc** command.

The System Resource Controller Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* gives an explanation of subsystems, subservers, and the System Resource Controller.

startup Command

Purpose

Turns on accounting functions at system startup.

Syntax

/usr/sbin/acct/startup

Description

The **startup** command turns on the accounting functions when the system is started, if called by the **/etc/rc** command file. See the **startup** example for the command to add to the **/etc/rc** file.

Security

Access Control: This command should grant execute (x) access only to members of the adm group.

Examples

To turn on the accounting functions when the system is started, add the following to the **/etc/rc** file:

```
/usr/bin/su - adm -c /usr/sbin/acct/startup
```

The **startup** shell procedure will then record the time and clean up the previous day's records.

Files

/usr/sbin/acct The path to the accounting commands.

Related Information

The **shutacct** command, **turnacct** command.

For more information about the Accounting System, the preparation of daily and monthly reports, and the accounting files, see the Accounting Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

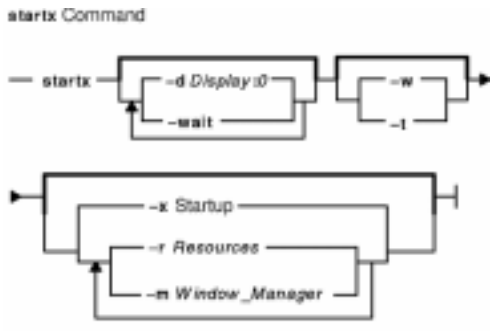
Setting Up an Accounting System in *AIX Version 4.3 System Management Guide: Operating System and Devices* explains the steps you must take to establish an accounting system.

startx Command

Purpose

Initializes an X session.

Syntax



startx [**−d** *Display***:***0*] [**−t** | **−w**] [**−x** Startup | [**−r** *Resources*] [**−m** *Window_Manager*]] [**−wait**]

Description

The **startx** command streamlines the process of starting an X session.

The command does the following:

- Sets the user's **DISPLAY** environment variable to identify the X server to the X clients
- When run from a workstation, starts the X server
- Starts the X clients.

The **startx** command redirects X server and X client error messages to the file specified by the user's **XERRORS** environment variable. This process is useful for debugging and gives the X server a clean startup and shutdown appearance on a workstation.

If a startup script file name is not given at the command line with the **−x** option, then the **startx** command searches for a file specified by the user's **XINITRC** environment variable. If the **XINITRC** environment variable is not set, then the **startx** command searches the user's home directory for a file called **.Xinit**, **.xinit**, **.Xinitrc**, **.xinitrc**, or **.xsession**, respectively, to begin the X client programs.

If a startup file is not found, the **startx** command runs the Window Manager indicated at the command line with the **−m** option, or invokes the window manager **mwm**, **twm**, **awm**, or **uwm** after finding the associated configuration file (**.mwmrc**, **.twmrc** , **.awmrc**, or **.uwmrc**, respectively). If a window manager configuration file is not found in the user's home directory, **startx** initiates an **Xterm** client and the **mwm** window manager.

When a startup file is not found, the **startx** command also instructs the loading of the resources file given at the command line with the **−r** option, or a file from the user's home directory called **.Xdefaults**, **.xdefaults**, **.Xresources**, or **.xresources**, respectively. If an X resources file is not found, then the X session will not be personalized.

If a startup file exists for a workstation and no resources are loaded by the user, then the **xinit** command within the **startx** command attempts to load an **.Xdefaults** file.

The use of a workstation is assumed when the X session is initiated from **/dev/lft***. If this is not the case, then the **−t** or **−w** option must be used.

Flags

−d*Display***:***0*	Specifies the display name of the X server to pass to the X clients during the process for startup.
−m*Window_Manager*	Starts the Window Manager when no startup script is found.
−r*Resources*	Loads the resources file when no startup script is found.
−t	Starts X clients for an X terminal.
−w	Starts the X server and X clients for an X window session on a workstation.
−wait	Prevents the X session from being restarted when the **xdm** command invokes **startx**.
−x Startup	Starts an X window session using the startup script.

> **Note:** You can use one or both of the **−m** and **−r** options, or the **−x** option, but you cannot use the **−x** option with the **−m** and **−r** options. In the startup script, it is the responsibility of the user to start a window manager session, load X resources, and spawn X clients.

Examples

1. To start an X session on a workstation, or an X terminal, enter:

   ```
   startx
   ```

2. To force start an X session on a workstation, enter:

   ```
   startx −w
   ```

3. To start an X session for an X terminal, and log off the user's telnet session, enter:

   ```
   startx; kill −9 $$
   ```

4. To start an X session using the **.xinitrc** script, enter:

   ```
   startx −x .xinitrc
   ```

5. To start an X session using the **mwm** window manager, enter:

   ```
   startx −m mwm
   ```

   However, if a startup script file is found, the **−w** option is ignored.

6. In the startup script, it is the responsibility of the user to start a window manager, load X resources, and spawn X clients. The following is an example of an **.xsession** script.

   ```
   #!/bin/csh
    (mwm &)
    xrdb −load .Xdefaults
    (xclock −g 75x75+0+0 &)
    (xbiff −g 75x75+101−0 &)
    if ("/dev/lft*" == "`tty`") then
       aixterm −g 80x24+0+0 +ut −C −T `hostname`
    else
       aixterm −g 80x24+0+0 +ut −T `hostname`
    endif
   ```

For a workstation, the last line in the startup script should be a foreground **aixterm** command with the **−C** option for console messages.

For an X terminal, the last line in the startup script should be a foreground **aixterm** command without the **−C** option. In addition, because some X terminals do not terminate the **telnet** session upon closing, the user must exit the current telnet session before using hot keys to switch to the X session.

Also, the **startx** command can be used by the **xdm** command in the **/usr/lib/X11/xdm/Xsession** file. This provides the **xdm** command with the features of the **startx** command.

Files

The following file names have been historically used for the startup of an X session.

$HOME/.xerrors	Where **startx** is to redirect error messages. By default, **startx** redirects errors to the **.xerrors** file in user's home directory.
$HOME/.Xinit,	
$HOME/.xinit,	
$HOME/.Xinitrc,	
$HOME/.xinitrc,	
$HOME/.xsession	Used as a Startup file containing shell commands to start a window manager, load X resources, and spawn X clients.
$HOME/.Xdefaults,	
$HOME/.xresources	Used as an X resources file loaded to set user preferences for X clients.
$HOME/.mwmrc	An **mwm** configuration file.
$HOME/.twmrc	A **twm** configuration file.
$HOME/.awmrc	An **awm** configuration file.
$HOME/.uwmrc	A **uwm** configuration file.
/dev/lft*	The terminal, or tty, interface of a workstation's initial **login** shell.

Related Information

The **mwm** command, **xinit** command, **xdm** command, **aixterm** command, **telnet, tn, or tn3270** command, **X** command, and **xrdb** command.

statd Daemon

Purpose

Provides crash and recovery functions for the locking services on NFS.

Syntax

/usr/sbin/rpc.statd

Description

The **statd** daemon interacts with the **lockd** daemon to provide crash and recovery functions for the locking services on Network File System (NFS). The **statd** daemon should always be started before the **lockd** daemon.

The **statd** daemon is started and stopped by the following SRC commands:

```
startsrc −s rpc.statd
```

```
stopsrc −s rpc.statd
```

The status monitor maintains information on the location of connections as well as the status in the **/etc/sm** directory, the **/etc/sm.bak** file, and the **/etc/state** file. When restarted, the **statd** daemon queries these files and tries to reestablish the connection it had prior to termination. To restart the **statd** daemon, and subsequently the **lockd** daemon, without prior knowledge of existing locks or status, delete these files before restarting the **statd** daemon.

Related Information

The **lockd** daemon.

List of NFS Commands.

Network File System (NFS) Overview for System Management in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

stem Command

Purpose

Allows insertion of instrumentation code at the entry and exit points of existing program and library subroutines.

Syntax

Options to Instrument Object Files:

stem { **−mf** *MapFile* | **−p** *Program* | **−pshm** *Program* } [**−ld_cmd**] [**−ld_options** *QuotedString*] [**−noleaf**][**−libdir** *Directory*] [**−exedir** *Directory*]

Options To Control the Shared−Memory Buffer:

stem { **−on** [**−noreset**] | **−off** | **−shm** [*Size*] | **−shmkill** }

Option To Produce Callgraph From Shared−Memory Buffer Contents

stem −cg *Program*

Description

The **stem** (Scanning Tunneling Encapsulating Microscope) command is a tool for inserting instrumentation code, either user−supplied or default routines provided with stem, in subroutines. The **stem** command operates on existing libraries and programs without requiring source code or recompilation of the libraries or programs.

The **stem** command places instrumentation at the entry and exit points of selected subroutines, known as *target routines*. Target routines can be in user programs or shared libraries. User−defined instrumentation is in the form of subroutines, known as *instrumentation routines*. Instrumentation routines are simple C subroutines readily created and tailored by you.

In addition to instrumenting target routines, the **stem** command can replace target routines with instrumentation routines.

> **Note:** The **stem** command does not require an installed C compiler.

Flags

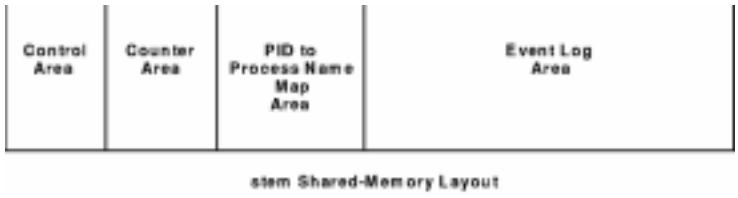
−cg	Produces a shared−memory callgraph. Use after entering **stem −pshm**.
−exedir*Directory*	Stores instrumented executables in the specified directory. The **stem** command stores instrumented executables in the default **/tmp/EXE** directory if you omit the **−exedir** flag.
−ld_cmd	Uses the **ld_cmd** file, if one exists. You can modify **ld_cmd** files to resolve **ld** errors. This flag requires either the **−mf**, **−p**, or **−pshm** flag to be used in the same command.
−ld_options "*QuotedString*"	Passes **ld** options, which you specify in the "*QuotedString*" parameter, to the **ld** command. This flag requires either the **−mf**, **−p**, or **−pshm** flag to be used in the same command.

−libdir*Directory*	Stores instrumented libraries in the specified directory. The **stem** command stores instrumented libraries in the default **/tmp/LIBS** directory if you omit the **−libdir** flag.
−mf *MapFile*	Uses the specified map file as a **stem** map file.
−noleaf	Ignores and does not instrument, leaf routines. Leaf routines, the final process in a sequence of subroutines, do not make calls to other routines. This flag requires either the **−mf**, **−p**, or **−pshm** flag to be used in the same command.
−noreset	Does not reset the pointer to the beginning of the shared−memory buffer. Use this flag only with the **−on** flag.
−off	Turns off the **ON/OFF** flag in the shared−memory buffer.
−on	Turns on the **ON/OFF** flag in the shared−memory buffer.
−p *Program*	Instruments all routines in the specified program. This is the easy way to instrument a program. The **stem** command uses **Stem_Standard_entry()** and **Stem_Standard_exit()** (in the **stem_samples.o** file) for instrumentation routines.
−pshm*Program*	Instruments all routines in the specified program. This flag is a slight variation on the **−p** flag. The **stem** command uses **Stem_ShmEnter()** and **Stem_ShmExit()** (in the **stem_samples.o** file) for instrumentation routines.
−shm*[Size]*	Sets the size of the shared−memory buffer, using the number of bytes specified in the *Size* variable. If *Size* is not specified, the **stem** command displays the current shared−memory buffer size. When the **−shm** flag is omitted and a shared−memory buffer does not exist, the **stem** command creates a shared−memory buffer of size 40,960 bytes. **Note:** This flag pins the shared−memory buffer; be careful not to make the buffer too large.
−shmkill	Destroys an existing shared−memory segment. This flag causes the **stem** command to ignore other flags.

Shared−Memory Buffer

Though **stem** instrumentation routines can print to files and call the **trace** daemon, another output mechanism is a specially constructed shared−memory segment. The shared−memory segment enables shared−library and user−program target routines to share one output stream.

The current buffer design includes four sections, as shown in the "**stem** Shared−Memory Layout" illustration.



stem Shared-Memory Layout

The **/usr/lpp/stem/headers/stem_shm.h** header file defines the buffer structure and event types.

Control Area

The Control Area includes the **ON/OFF** flag, pointers defining the boundaries of the other areas, and the **WithinInstrumentation** flag. **WithInstrumentation** determines the present instrumentation state and prevents *infinite instrumentation loops*.

Infinite instrumentation loops result from instrumenting a target routine and then directly or indirectly calling the target routine from the instrumentation routine. Assume the following **Stem_instr_routine()**:

```
Stem_instr_routine()  {
         printf("Call results in an infinite instrumentation
             loop\n");
         printf("if the printf routine is instrumented\n);
}
```

instruments the target routine **printf()**.

The instrumented program encounters an infinite instrumentation loop because calls to **printf()** are directed to **Stem_instr_routine()** and **printf()** is directly called by **Stem_instr_routine()**. The **WithinInstrumentation** flag in the shared−memory buffer and a test of the flag in each instrumentation routine prevents infinite instrumentation loops. Properly coded, **Stem_instr_routine()** appears as follows:

```
Stem_instr_routine()  {
        if ( shmbuf[ WithinInstrumentation ] )  return();
        shmbuf[ WithinInstrumentation ] = 1;  /* turn on flag */
        printf("not an infinite instrumentation loop\n");
        shmbuf[ WithinInstrumentation ] = 0; /* turn off flag */
}
```

Of course the preceding segment assumes addressability to the shared−memory buffer has already been established by global variable **shmbuf**, an integer pointer. See the file **/usr/samples/perfagent/stem/stem_samples.c** for example instrumentation routines and usage of the **WithinInstrumentation** flag.

> **Note:** There exists only one **WithinInstrumentation** flag within the shared−memory buffer. You require multiple flags if more than one **stem** instrumented program runs at the same time.

Counter Area

The Counter Area is reserved for a counter−based implementation. For example, rather than producing an event for each file open, instrumentation routines can add to a counter or set of counters.

PID to Process Name Map Area

The PID to Process Name Map Area contains the process IDs and process names for all processes running on the system at the time of the last **stem −on** call.

Event Log Area

The Event Log Area houses events of any type and any length. The **stem** command defines some event types in the **/usr/samples/perfagent/stem/stem_shm.h** header file. Each event has the structure shown in the stem Event Structure illustration .

stem Event Structure

The first word of the event stores the type and length markers. The length represents the number of 32−bit words, including the first word.

Shared Libraries and the LIBPATH environment variable

Both instrumented and uninstrumented programs can make use of instrumented libraries by setting the **LIBPATH** environment variable. The **stem** command prints a message showing what setting to use for **LIBPATH** whenever a shared library is instrumented. Unless the **−libdir** flag specifies another value, the **LIBPATH** environment variable should be set to **/tmp/LIBS**. The following command makes use of an instrumented library located in **/tmp/LIBS**:

```
LIBPATH=/tmp/LIBS   /bin/ps −eaf
```

The preceding uninstrumented **/bin/ps** program will use whatever instrumented libraries exist in the directory **/tmp/LIBS**.

Directories and FilesCreated by the stem Command

The **stem** command creates the **./stemdir** subdirectory in the current working directory. Under the **./stemdir** directory, **stem** creates additional subdirectories, one directory for each executable or library to instrument. For the following map file,

```
StemAll  /bin/some_program      .Stem_ShmEnter   .Stem_ShmExit stem_samples.o
StemAll  /usr/lib/libc.a:shr.o  .Stem_ShmEnter   .Stem_ShmExit stem_samples.o
.main    runit                  .Stem_ShmEnter   No_Exit       stem_samples.o
```

the **stem** command creates the following three directories:

- **./stemdir/_bin_some_program**
- **./stemdir/_usr_lib_libc.a_shr.o**
- **./stemdir/runit**

Notice that each slash (/) in the second column of the map file has been substituted with an underline character (_). Two files in each **stem**−created directory are of particular interest: **instrumented** and **not_instrumented**. The **instrumented** file contains the list of target routines instrumented by **stem**. The **not_instrumented** file contains the list of target routines not instrumented by **stem** and a brief reason why they were not instrumented.

The **stem** command also creates directories to store instrumented libraries and instrumented programs. The **stem** command places:

- instrumented programs in the **/tmp/EXE/** directory by default, or in the directory specified by the **−exedir** flag,
- instrumented libraries in the **/tmp/LIBS** directory by default, or in the directory specified by the **−libdir** flag, and
- special instrumentation libraries created for stripped programs in **/tmp/LIBS** , or in the directory specified by the **−libdir** flag.

Stripped and Un−Stripped Programs

A program is stripped if the **strip** command removed its symbol table. The **stem** command can instrument both stripped and un−stripped programs, but it only instruments un−stripped libraries.

The instrumentation method for stripped and un−stripped programs are different. For un−stripped programs, **stem** uses the **ld** command, the linkage editor, to combine the program and the instrumentation object file. The **ld** command is responsible for resolving external references and searching libraries. The **stem** command tries to provide the necessary parameters to **ld** but sometimes is unable to do so. Such cases require user intervention.

The **stem** command flags **−ld_options** and **−ld_cmd** assist user intervention. The **−ld_options** flag enables you to pass **ld** the necessary options required to resolve external references. The **−ld_cmd** flag is similar though more direct. The **stem** command creates a shell procedure, named **ld_cmd**, in the stem−created directory for the program. You can change this shell procedure, and then use **stem** with the **−ld_cmd** flag to run it.

The **stem** command instruments stripped programs in an entirely different manner because the **ld** command does not operate on stripped programs. Therefore, the instrumentation object file cannot be combined with the stripped program. To instrument stripped programs, **stem** creates a dependency between the stripped program and a library containing the instrumentation object code. This is all done automatically by **stem.**

> **Note:** The **ld** command is not used by **stem** on stripped programs. If **ld** problems continue when instrumenting an un−stripped program, try stripping the program before having **stem** instrument it.

Instrumentable Target Routines

Instrumentable target routines represent most of the routines within programs and libraries. However, **stem** cannot instrument certain assembler language routines and a few compiler generated routines. Before instrumenting a target routine, **stem** carefully evaluates the sequence of instructions within the routine. If the routine does not appear to be following normal linkage conventions, **stem** will not instrument it.

The **stem** command uses the **stripnm** program to locate the addresses of routines within programs and libraries. Though uncommon but not impossible, if **stripnm** cannot find a routine, **stem** will not be able to instrument it.

Limitations and Cautions

The **stem** command operates by inserting instrumentation routines in existing programs and libraries, and by changing the program flow. The instrumentation routines can perform most operations but there are limitations. Instrumentation routines are being inserted within the framework of an existing program or library and must conform to the execution context of that environment, which can be restrictive.

> **Note:** The **stem** command does not work on non−archived libraries and does not support programs or libraries with more than 6200 subroutines.

For example, instrumentation routines can typically open files and write to them without incident. However, C++ programs with exception handling are not supported. Also, some processes, **aixterm** for example, occasionally close all open files. This presents a problem to the instrumentation routines. They must detect the fact that one of their open files has been closed before performing any input or output operations on the file. Programming to detect this behavior is in the standard instrumentation routines in the **/usr/samples/perfagent/stem/stem_samples.c** file.

The **stem** command has been tested on a number of programming languages, C, C++, and Fortran, though C has received most of the testing. Most C and C++ programs and libraries can be instrumented without incident. Fortran programs, however, do pose one problem. The Fortran programming language supports multiple entry points to subroutines. The **stem** command cannot currently detect multiple entry points. Therefore, the **stem** output cannot properly represent entry and exit events from some Fortran routines.

The **stem** command does not work on programs containing LISP routines and cannot detect if a program was built with LISP routines in it.

Instrumentation Routine Parameters and Return Values

The **stem** command does not modify the parameters to or return values from target routines. Instrumentation routines can, however, access these without exception. The only difficulty is discerning the types of these parameters and return values. For example, consider the following simple instrumentation routine, **Stem_simple_entry()**.

```
Stem_simple_entry(a, b, c)
unsigned a;
long b;
int c;
{
        printf(" %u  %x  %d\n", a, b, c);
}
```

With **stem**, it is possible to send hundreds or thousands of target routines to this instrumentation routine, most of which are not likely to have three parameters. Fewer still will have three parameters whose types match those in the preceding example. This does not pose a problem for the preceding instrumentation code. The **stem** command passes parameters to routines in registers 3 through 10. In the preceding example, register 3's contents is placed in parameter **a**, even if **a**'s type does not match the type being passed in to the target routine. Parameter **b**, contains the contents of register 4, and **c** has the contents of register 5. So, even if a routine has only one parameter, **b** and **c** will still be assigned the contents of registers 4 and 5 respectively.

Conventions mandate that return values from target routines be placed within register 3. Therefore, in the preceding routine, **a** would contain the return value from any routines that used **Stem_simple_entry()** as their exit point instrumentation routine.

> **CAUTION:** The preceding instrumentation routine does not make use of any pointers to structures or pointers to character strings. Programs can encounter a segmentation error if an instrumentation routine attempts to reference with a parameter when the parameter was not a valid pointer. Consider the following:

```
Stem_dangerous_entry(a, b, c)
char *a;
long b;
int c;
{
        printf(" %s  %x  %d\n", a, b, c);
}
```

Stem_dangerous_entry() is dangerous because the instrumentation routine tries to print a string pointed to by **a.** The preceding only works if the first parameter to target routines instrumented with **Stem_dangerous_entry()** is always a pointer to a character string.

Security

Access Control: You must be root or a member of the perf group to run this command.

Examples

1. Simple Program Instrumentation (The −p Flag)

To get started, simply choose a program to instrument and run the following commands:

```
stem −p Program
/tmp/EXE/Program
/bin/pg stem_out_*
```

The **−p** flag instructs **stem** to instrument all instrumentable target routines of *Program* with the routines **Stem_Standard_entry()** and **Stem_Standard_exit()**. The "Instrumentable Target Routines" section of this article contains a complete definition of instrumentable.

By default, the **stem** command puts the instrumented executable in the **/tmp/EXE/***Program* file and leaves the original *Program* file unaltered. Include the full path after the **−p** flag if *Program* does not reside in the current directory.

Instrumentation routines **Stem_Standard_entry()** and **Stem_Standard_exit()** produce one line of output each time a routine is entered or exited as **/tmp/EXE/***Program* runs. The **stem** command pairs and indents the routine entry and exit events to reflect the nesting level within the program.

The **stem** output from running **/tmp/EXE/***Program* is directed to one or more output files, (*stem_out_001, stem_out_002, stem_out_003,..., stem_out_nnn)*. The numeric suffix *_nnn,* designates the execution context of the instrumented program. An execution context can be signal handling code, or different threads. Many instrumented programs have just one execution context and only create the file *stem_out_001*. This article refers to execution contexts and numeric suffixes as threads and thread IDs (TID), respectively.

If *Program* in the preceding were /bin/ps, output in *stem_out_001* would appear as:

```
  Seconds.usecs  TID  Routine Names & Seconds.usecs since entering routine.
766011575.869091   1 ->main
766011575.906776   1         ->setlocale
766011575.907578   1         <-setlocale 0.000802
766011575.907741   1         ->catopen
766011575.907881   1         <-catopen 0.000140
766011575.908019   1         ->setbuf
766011575.908217   1         <-setbuf 0.000198
766011575.908366   1         ->parseberkeley
766011575.908497   1         <-parseberkeley 0.000131
766011575.908636   1         ->getpagesize
766011575.908861   1         <-getpagesize 0.000225
766011575.909093   1         ->ttyname
766011575.909698   1         <-ttyname 0.000605
766011575.909873   1         ->strncmp
766011575.910186   1         <-strncmp 0.000313
.....
766011576.642508   1         ->prcom
766011576.642635   1                 ->getthreaddata
766011576.642766   1                         ->getthrds
766011576.642927   1                         <-getthrds 0.000161
766011576.643067   1                         ->malloc
766011576.643206   1                         <-malloc 0.000139
766011576.643350   1                         ->getthrds
766011576.643513   1                         <-getthrds 0.000163
766011576.643654   1                 <-getthreaddata 0.001019
766011576.643794   1                 ->getprocftdata
766011576.643924   1                         ->malloc
766011576.644061   1                         <-malloc 0.000137
766011576.644204   1                 <-getprocftdata 0.000410
766011576.644344   1                 ->gettty
766011576.644475   1                 <-gettty 0.000131
766011576.644616   1         <-prcom 0.002108
766011576.644752   1         ->fclose
766011576.644910   1         <-fclose 0.000158
766011576.645051   1         ->done
766011576.645180   1                 ->exit
```

The first column contains the time (seconds and microseconds) of the enter or exit event. The second column has the stem thread ID (TID), which always corresponds to the suffix of the particular *stem_out_nnn* file. The first and second columns appear as they do so a sorted merge of all *stem_out_** files will produce an easy−to−read stream of output, clearly showing the order of routine entries and exits across thread IDs.

The indented routine name, prefixed with either the routine entry symbol (−>) or the routine exit symbol (<−), appears to the right of the TID column. The indentation is meant to reflect the calling sequence or *callgraph* of the instrumented program. For exit events, one additional column appears and includes the elapsed time (seconds and microseconds) since entering the routine.

The preceding sample output contains one additional noteworthy characteristic. The routine named **exit()** has an enter event but no corresponding exit event. In fact, no additional enter or exit events appear in the output. The **Stem_Standard_entry()** instrumentation routine specifically recognizes the **exit()** routine because **exit()** begins the process of destroying the data structures and files used by the **Stem_Standard_entry()** and **Stem_Standard_exit()**. Upon encountering **exit()**, **Stem_Standard_entry()** sets a flag to avoid logging data for all subsequent enter and exit events.

The instrumentation source file, **/usr/samples/perfagent/stem/stem_samples.c**, shows the **Stem_Standard_entry()**, and **Stem_Standard_exit()** routines. You can easily modify these routines. The **−p** flag copies the **stem_samples.c** file to the current directory unless a **stem_samples.c** file exists in the current directory.

2. Shared−Memory Callgraphs (The −pshm and −cg Flags)

The **−pshm** flag is similar to the **−p** flag. The difference is in the instrumentation routines and the output mechanisms. The **−pshm** flag uses the **Stem_ShmEnter()** and **Stem_ShmExit()** routines to instrument each target routine within *Program*.

Stem_ShmEnter() and **Stem_ShmExit()** do not open files to log output, they log output in a specially−made shared−memory buffer. There are a few advantages to this approach.

- Logging data within the shared−memory buffer is faster than writing to files.
- Logging together of events from multiple instrumented processes in one output stream is possible.
- The**−on** and **−off** shared−memory buffer flags readily control the logging of events.

A disadvantage is post processing. You must run **stem** again using the **−cg** flag to view the data. The following is an example sequence of instructions using the **−pshm** and **−cg** flags:

```
stem −pshm /bin/ps
stem −on
/tmp/EXE/ps
stem −cg /tmp/EXE/ps
  PID         ElapsedTime    DeltaSecs    IAR    NAME
  9500 Enter    0.000000     0.000000  10000178   1  main
  9500 Enter    0.000290     0.000290  1000a7d0   1 . setlocale
  9500 Exit     0.001086     0.000796  1000a7d0   1 . setlocale
  9500 Enter    0.001120     0.000034  1000a7f4   1 . catopen
  9500 Exit     0.001223     0.000103  1000a7f4   1 . catopen
  9500 Enter    0.001249     0.000026  1000a818   1 . setbuf
  9500 Exit     0.001388     0.000139  1000a818   1 . setbuf
  9500 Enter    0.001422     0.000034  10008ad0   1 . parseberkeley
  9500 Exit     0.001448     0.000026  10008ad0   1 . parseberkeley
  9500 Enter    0.001474     0.000026  1000afd4   1 . getpagesize
  9500 Exit     0.001591     0.000117  1000afd4   1 . getpagesize
  9500 Enter    0.001673     0.000082  1000b040   1 . ttyname
  9500 Exit     0.002156     0.000483  1000b040   1 . ttyname
  9500 Enter    0.002193     0.000037  1000a8cc   1 . strncmp
  9500 Exit     0.002297     0.000104  1000a8cc   1 . strncmp
```

The instrumentation source file, **/usr/samples/perfagent/stem/stem_samples.c** shows the **Stem_ShmEnter()**, and **Stem_ShmExit()** routines. You can easily modify these routines. The **−pshm** flag copies the **stem_samples.c** file to the current directory unless a **stem_samples.c** file exists in the current directory.

3. Stem Map File (The −mf flag)

Use the **−p** and **−pshm** flags to instrument all instrumentable target routines within a specified program with the standard instrumentation routines. Use the **−mf** flag to instrument library target routines or to better control the instrumentation of program target routines.

The *Map_File* parameter specifies a file you create whose contents describe which target routines in user programs and/or shared libraries to instrument. Map files contain the names and locations of target routines and of the entry and exit instrumentation routines. The following example shows the format of a **stem** map file:

```
Target     Target                    Entry               Exit               Instrumentation
Routine    Routine                Instrumentation      Instrumentation    Object File
           File                   Routine              Routine
```

The following entry is an example of a **stem** map file written as one line.

```
.malloc   /usr/lib/libc.a:shr.o   .Stem_malloc_entry   .Stem_malloc_exit   stem_samples.o
```

Specify target and instrumentation routine names with a . (dot) as the first character. This is how they are reported by the **/usr/bin/nm** and **/usr/bin/stripnm** programs.

To avoid name collisions with routines in the target files, **stem** mandates that all instrumentation routines begin with the "Stem_" prefix. In fact, all routines in an instrumentation object file must begin with the "Stem_" prefix. Instrumentation object files in column 5 can have any name, but they should reside within the current directory.

The preceding map file changes the program flow of target routine **malloc()**. After instrumentation, calls to target routine **malloc()** are directed first through instrumentation routine **Stem_malloc_entry()**, then through **malloc()**, and finally through instrumentation routine **Stem_malloc_exit()**. The calling sequence becomes:

```
Stem_malloc_entry()  --->  malloc()  --->  Stem_malloc_exit()
```

Target routine **malloc()** must exist within the shared library target file **/usr/lib/libc.a**. Because the library archive **/usr/lib/libc.a** contains several object files, such as **shr.o** and **meth.o**, the stem map file must identify the correct object because target routine **malloc()** could exist in more than one of the archive's object files. A colon (:) separates the object file from the archive name.

Instrumentation routines **Stem_malloc_entry()** and **Stem_malloc_exit()** must exist in instrumentation object file **stem_samples.o**. Also, **stem_samples.o** must reside in the current directory. If the file **Make.Stem** exists in the current directory, **stem** tries to create the instrumentation object file by running the command:

```
make −f Make.Stem stem_samples.o.
```

If the file **Make.Stem** does not exist, **stem** tries to create the instrumentation object file by running the command:

```
make stem_samples.o.
```

> **Note:** The **stem** software includes a sample instrumentation source file, **stem_samples.c**, in the directory **/usr/samples/perfagent/stem**. It contains several example instrumentation routines to use as is, or they can be modified to suit your specific requirements.

4. Map File Keywords (The StemAll, No_Exit, and Replace Flags)

To instrument all routines in a program or shared library, list each routine separately in the map file, or use the keyword **StemAll**. In the following map file example, all instrumentable target routines in **/bin/some_program** and **/usr/lib/libc.a:shr.o** are directed through the instrumentation routines

Stem_ShmEnter() and **Stem_ShmExit()**.

> **Note:** Instrumentable target routines represent most routines. The **stem** command cannot instrument certain assembler−language routines. The **stem** command creates an "instrumented" and not a "not_instrumented" file for each library or user program specified in column 2 of the map file. See the "Instrumentable Target Routines" section of this article for additional details.

```
StemAll  /bin/some_program  .Stem_ShmEnter  .Stem_ShmExit stem_samples.o
StemAll  /usr/lib/libc.a:shr.o  .Stem_ShmEnter .Stem_ShmExit stem_samples.o
```

To instrument only the entry point of a target routine, use the keyword **No_Exit** in column 4, the exit instrumentation routine column. In the following example map file, all routines in **/bin/some_program** have their entry points instrumented but not their exit points.

```
StemAll  /bin/some_program  .Stem_ShmEnter  No_Exit  stem_samples.o
```

To replace a target routine with an instrumentation routine, use the keyword **Replace** in column 4, the exit instrumentation column. In the following example map file, all calls to the **some_targ()** routine in **/bin/some_program** will be directed to the instrumentation routine **Stem_replace_example()**.

```
.some_targ /bin/some_program  .Stem_replace_example  Replace stem_samples.o
```

> **Note:** Target routine replacement is inherently dangerous. To avoid unpredictable results, the replacement routines must adhere to all pre−conditions and post−conditions of the target routines. For example, you can replace a sorting routine with another as long as the output is properly sorted upon exiting the replacement routine.

5. Shared−Memory (The −shm Size, −shmkill, −on, −off, −noreset Flags)

The **stem** command uses a shared−memory segment to control program flow, to assist communication between user programs and libraries, and to store data in the form of events. The **stem** command requires the existence of the shared−memory buffer and will create one even if no shared−memory flags are specified. The "Shared−Memory Buffer" section describes the layout of the shared−memory buffer.

Multiple postprocessing programs can process the shared−memory buffer at the same time. All stem−instrumented user programs and shared libraries can write to the shared−memory buffer. Use the**−shm** flag to specify the size of the buffer. You can increase the buffer size but not reduce it. The **−shm** flag without the *Size* parameter displays the current size of the shared−memory.

> **Note:** The **−shmkill** flag can destroy the buffer.

The flags **−on**, **−off**, and **noreset**, control the logging of events. Logging of events only occurs if the shared−memory buffer's **ON/OFF** flag is set to ON and the buffer is not full. When created, the buffer's current pointer is set to full and the **ON/OFF** flag is OFF. The command:

```
stem −on
```

resets the buffer pointer and turns on the **ON/OFF** flag. This command also stores the process IDs and process names of all running processes in the shared−memory buffer.

> **CAUTION:** Since the commands stem and syscalls share the same buffer, do not run them at the same time.

Files

/usr/bin/stem Contains the **stem** command.

Related Information

The **ld** command, **strip** command, **stripnm** command, and, **syscalls** command.

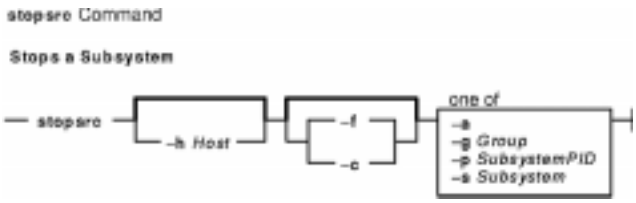
stopsrc Command

Purpose

Stops a subsystem, a group of subsystems, or a subserver.

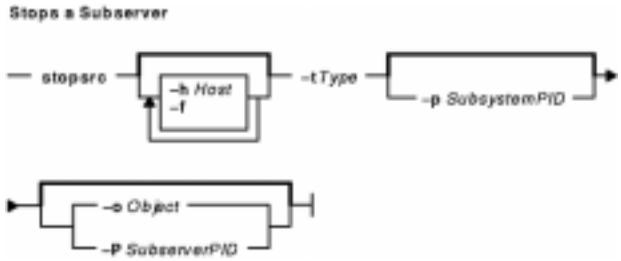
Syntax

To Stop a Subsystem



stopsrc [**−h** *Host*] [**−f** | **−c**] {**−a** | **−g***Group* | **−p***SubsystemPID* | **−s***Subsystem* }

To Stop a Subserver



stopsrc [**−h***Host*] [**−f**] **−t***Type* [**−p***SubsystemPID*] [**−P***SubserverPID* | **−o***Object*]

Description

The **stopsrc** command sends a request to the System Resource Controller (SRC) to stop a subsystem, a group of subsystems, or all subsystems. The **stopsrc** command sends the System Resource Controller a subsystem request packet that is forwarded to the subsystem for a stop subserver request.

In the absence of the **−f** (stop force) flag, a normal stop action is assumed. A normal stop requests that a subsystem or subserver complete all current processing, release resources when all application activity has been completed, and then end. No new requests for work should be accepted by the subsystem.

A forced stop requests that a subsystem or subserver end quickly, releasing all resources, but not wait for application activity to complete.

A cancel action stops the subsystem after the subsystem's resources are released and after a grace period. This grace period is specified in the subsystem object class. The cancel stop is used only for subsystem stops and is always sent to the subsystem as the **SIGTERM** signal. The subsystem should catch this signal, perform subsystem clean up operations, and end. If the subsystem does not end within the wait time period, specified in the subsystem object class, the subsystem is sent a **SIGKILL** signal to ensure that the subsystem stops.

If the subsystem uses sockets or message queues for communication, a packet is constructed and sent to the subsystem. If the subsystem uses signals for communication, the subsystem is sent the appropriate signal

from the subsystem object class.

Flags

−a	Specifies that all subsystems are to be stopped.
−c	Specifies that the stop request is a canceled stop request. For a cancel stop request, a **SIGTERM** signal is sent to the subsystem. After the wait time contained in the subsystem object class has passed, if the subsystem has not yet ended, the subsystem is sent a **SIGKILL** signal.
−f	Specifies a forced stop request.
−g *Group*	Specifies that a group of subservers is to be stopped. The command is unsuccessful if the *Group* name is not contained in the subsystem object class.
−h*Host*	Specifies the foreign *Host* machine on which this stop action is requested. The local user must be running as "root". The remote system must be configured to accept remote System Resource Controller requests. That is, the **srcmstr** daemon (see **/etc/inittab**) must be started with the **−r** flag and the **/etc/hosts.equiv** or **.rhosts** file must be configured to allow remote requests.
−o *Object*	Specifies that a subserver *Object* value is to be passed to the subsystem as a character string.
−p *SubsystemPID*	Specifies a particular instance of the subsystem to stop, or a particular instance of the subsystem to which the stop subserver request is to be passed.
−P*SubserverPID*	Specifies that a subserver PID is to be passed to the subsystem as a character string.
−s *Subsystem*	Specifies a subsystem to be stopped. The *Subsystem* parameter can be the actual subsystem name or the synonym name for the subsystem. The **stopsrc** command stops all currently active instances of the subsystem. The command is unsuccessful if the *Subsystem* name is not contained in the subsystem object class.
−t *Type*	Specifies that a subserver is to be stopped. The **stopsrc** command is unsuccessful if the *Type* specified is not contained in the subserver object class.

Examples

1. To stop force a subsystem on a foreign host, enter:

   ```
   stopsrc −h zork −s srctest −f
   ```

   This forces a stop on all the instances of the `srctest` subsystem on the `zork` machine.

2. To stop cancel a subsystem group, enter:

   ```
   stopsrc −g tcpip −c
   ```

   This activates a stop cancel on all the subsystems in the `tcpip` group.

3. To stop a subserver, enter:

   ```
   stopsrc −t tester −p 1234
   ```

   This stops the `tester` subserver that belongs to the `srctest` subsystem with a subsystem PID of `1234`.

4. To stop all subsystems, enter:

   ```
   stopsrc −a
   ```

This stops all the active subsystems on the local machine.

Files

/etc/objrepos/SRCsubsys	Specifies the SRC Subsystem Configuration Object Class.
/etc/objrepos/SRCsubsvr	Specifies the SRC Subserver Configuration Object Class.
/etc/services	Defines the sockets and protocols used for Internet services.
/dev/SRC	Specifies the **AF_UNIX** socket file.
/dev/.SRC−unix	Specifies the location for temporary socket files.

Related Information

The **startsrc** command.

The System Resource Controller Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* gives an explanation of subsystems, subservers, and the System Resource Controller.

stpinet Method

Purpose

Disables the inet instance.

Syntax

stpinet [**−l"***Interface ...*"] [**−t***Time*]

Description

If **stpinet** is started with a list of network interfaces specified with the **−l** option, then this method only stops those IFs. Otherwise, **stpinet** informs users of the impending demise of TCP/IP, using the **wall** command, and invokes the **ifconfig** command to mark each configured IF as **down**. If no network interfaces are specified, the status flag of the inet instance is set to DEFINED.

Flags

−l "*Interface ...*"	Specifies the name of the interface to be disabled.
−t*Time*	Specifies the time in minutes until the inet instance is stopped.

Examples

The following example disables the inet instance `tr0` five minutes from the time the method is executed:

```
stpinet −l "tr0" −t 5
```

Related Information

The **ifconfig** command, **rmdev** command, **wall** command.

The **odm_run_method** subroutine.

Writing a Device Method in *AIX Version 4.3 Kernel Extensions and Device Support Programming Concepts*.

Object Data Manager (ODM) Overview for Programmers in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

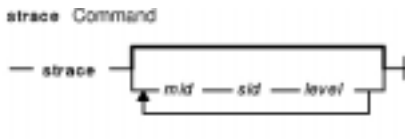
Understanding Network Interfaces in *AIX Version 4.3 System Management Guide: Communications and Networks*.

strace Command

Purpose

Prints STREAMS trace messages.

Syntax



strace [*midsidlevel*] ...

Description

The **strace** command without parameters writes all STREAMS event trace messages from all drivers and modules to its standard output. These messages are obtained from the STREAMS log driver. If parameters are provided, they must be in triplets. Each triplet indicates that tracing messages are to be received from the given module or driver, subID (usually indicating minor device), and priority level equal to or less than the given level. The `all` token may be used for any member to indicate no restriction for that attribute.

Parameters

mid Specifies a STREAMS module ID number.

sid Specifies a subID number.

level Specifies a tracing priority level.

Output Format

The format of each trace message output is:

```
<seq> <time> <ticks> <level> <flags> <mid> <sid> <text>
```

`<seq>` Trace sequence number

`<time>` Time of message in *hh:mm:ss*

`<ticks>` Time of message, in machine ticks, since system was started

`<level>` Tracing priority level

`<flags>` Has one of the following values:

E Message is also in the error log

F Indicates a fatal error

N Mail was sent to the system administrator

`<mid>` Module ID number of source

`<sid>` SubID number of source

`<text>` Formatted text of the trace message

On multiprocessor systems, `<text>` is composed of two parts:

- the number of the processor where the owner of the message has sent it,
- the formatted text itself.

Once initiated, the **strace** command continues to execute until terminated by the user.

> **Note:** Due to performance considerations, only one **strace** command is permitted to open the STREAMS log driver at a time. The log driver has a list of the triplets specified in the command invocation, and compares each potential trace message against this list to decide if it should be formatted and sent up to the **strace** process. Hence, long lists of triplets have a greater impact on overall STREAMS performance. Running the **strace** command has the most impact on the timing of the modules and drivers generating the trace messages that are sent to the **strace** process. If trace messages are generated faster than the **strace** process can handle them, some of the messages will be lost. This last case can be determined by examining the sequence numbers on the trace messages output.

Examples

1. To output all trace messages from the module or driver whose module ID is `41`, enter:

   ```
   strace 41 all all
   ```

2. To output those trace messages from driver or module ID `41` with sub−IDs `0`, `1`, or `2`:

   ```
   strace 41 0 1 41 1 1 41 2 0
   ```

Messages from sub−IDs `0` and `1` must have a tracing level less than or equal to 1. Those from sub−ID `2` must have a tracing level of 0.

Related Information

List of Streams Commands, STREAMS Overview, Understanding the log Device Driver in *AIX Version 4.3 Communications Programming Concepts*.

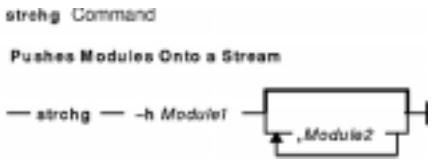
strchg Command

Purpose

Changes stream configuration.

Syntax

To push modules onto a stream:



strchg −h*Module1* [**,***Module2* ...]

To pop modules off a stream:

strchg −p [**−a** | **−u***Module*]

To push and pop modules to conform to the configuration file:

strchg −f*File*

Description

The **strchg** command is used to alter the configuration of the stream associated with the user's standard input. The **strchg** command pushes modules on the stream, pops modules off of the stream, or both. Only the root user or owner of a STREAMS device can alter the configuration of that stream. If another user attempts to alter the configuration, the **strchg** command will not succeed.

Note: If modules are pushed in the wrong order, the stream might not function as expected.

Flags

−a	Pops all modules above the topmost driver off of a stream. The **−p** flag must be used in front of the **−a** flag.
−f*File*	Pushes and pops the necessary modules to conform the stream to the configuration given in the specified file. The **−h**, **−p**, and **−f** flags are mutually exclusive.
−h *Module1*	Pushes modules onto a stream. The modules are listed on the command line in the order they are to be pushed.
−p	Pops a module off of a stream. Used alone, the **−p** flag pops the topmost module from the

stream.

−u *Module* Pops all modules above the specified module off of a stream. The **−p** flag must be used in front of the **−u** flag.

The **−a** and **−u** flags are mutually exclusive.

Parameters

Module1 Specifies the module to be pushed onto a stream. (Used by the **−h** flag.)

Module Specifies the topmost module to remain on a stream. All modules above this module are popped off of the stream. (Used by the **−u** flag.)

File Contains a list of modules representing the desired configuration of the stream. Each module name must appear on a separate line, where the first name represents the topmost module and the last name represents the module that is closest to the driver.

Return Values

On successful completion, the **strchg** command returns a value of 0. Otherwise, it returns a nonzero value and prints an error message indicating usage error, a bad module name, too many modules to push, failure of an **ioctl** operation on the stream, or failure to open the file specified by the *File* parameter.

Examples

1. To push the `ldterm` module on the stream, enter:

   ```
   strchg −h ldterm
   ```

2. To pop the topmost module from the stream associated with the `/dev/term/24` device, enter:

   ```
   strchg −p < /dev/term/24
   ```

   The user must be the owner of this device or the root user.

3. If the `fileconf` file contains the following:

   ```
   compat
   ldterm
   ptem
   ```

   the following command configures the stream so that the `ptem` module is pushed over the driver, followed by the `ldterm` module, and the `compat` module is pushed closest to the stream head.

   ```
   strchg −f fileconf
   ```

Related Information

The **strconf** command.

List of Streams Commands, STREAMS Overview in *AIX Version 4.3 Communications Programming Concepts*.

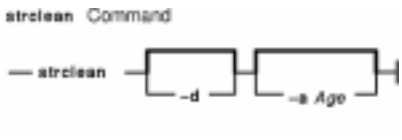
The **streamio** operations in *AIX Version 4.3 Technical Reference: Communications Volume 2*.

strclean Command

Purpose

Cleans up the STREAMS error logger.

Syntax



strclean [**−d**] [**−a***Age*]

Description

The **strclean** command is used to clean up the STREAMS error−logger directory on a regular basis: for example, by using the **cron** daemon. By default, all files with names matching **error.*** in the **/var/adm/streams** directory that have not been modified in the last three days are removed.

> **Note:** The **strclean** command is typically run using the **cron** deamon on a daily or weekly basis.

Flags

−a*Age* Specifies the maximum age, in days, for a log file.

−d Specifies a directory other than the default directory.

Examples

The following example has the same result as running the **strclean** command with no parameters.

```
strclean −d /var/adm/streams −a 3
```

Files

/var/adm/streams/error.* Contains the STREAMS error log.

Related Information

The **cron** daemon, and **strerr** daemon.

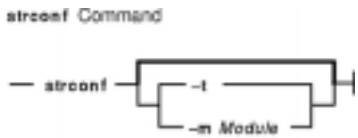
List of Streams Commands, STREAMS Overview in *AIX Version 4.3 Communications Programming Concepts*.

strconf Command

Purpose

Queries stream configuration.

Syntax



strconf [**−t**| **−m***module*]

Description

The **strconf** command is used to query the configuration of a stream. When used without any flags, it prints a list of all the modules in the stream as well as the topmost driver. The list is printed with one name per line, where the first name printed is the topmost module on the stream and the last item printed is the name of the driver.

Note: The **strconf** command only reads from standard input.

Flags

−m *Module* Determines if the specified module is present on the stream. If the module is present, the **strconf** command prints the message `yes` and returns a value of 0. If it is not present, the **strconf** command prints the message `no` and returns a nonzero value.

The **−t** and **−m** flags are mutually exclusive.

−t Prints only the topmost module of the stream (if one exists).

Parameter

Module Specifies the module for which to look.

Examples

1. For a stream that has only the `ldterm` module pushed above the `ports` driver, the **strconf** command (with no flags) would produce the following output:

   ```
   ldterm
   ports
   ```

2. Entering the following command asks if the `ldterm` module is on the stream:

   ```
   strconf −m ldterm
   ```

   The command produces the following output while returning an exit status of 0:

```
yes
```

Related Information

The **strchg** command.

The **streamio** operations in *AIX Version 4.3 Technical Reference: Communications Volume 1*.

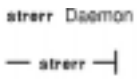
List of Streams Commands, STREAMS Overview in *AIX Version 4.3 Communications Programming Concepts*.

strerr Daemon

Purpose

Receives error log messages from the STREAMS log driver.

Syntax



strerr

Description

The **strerr** daemon receives error log messages from the STREAMS log driver and appends them to a log file. The error log files produced reside in the directory **/var/adm/streams**, and are named **error.***mm−dd*, where *mm* is the month and *dd* is the day of the messages contained in each log file.

The format of an error log message is:

```
<seq> <time> <ticks> <flags> <mid> <sud> <text>
```

These fields are defined as follows:

`<seq>` Error sequence number

`<time>` Time of message in *hh:mm:ss*

`<ticks>` Time of message in machine ticks since boot priority level

`<flags>` Has one of the following values:

T The message was also sent to a tracing process

F Indicates a fatal error

N Send mail to the person who administers your system

`<mid>` Module ID number of source

`<sid>` Sub−ID number of source

`<text>` Formatted text of the error message

On multiprocessor systems, `<text>` is composed of two parts:

- the number of the processor where the owner of the message has sent it,
- the formatted text itself.

Messages that appear in the error log are intended to report exceptional conditions that require the attention of the person who administers your system. Those messages indicating the total failure of a STREAMS driver or module should have the **F** flag set. Those messages requiring the immediate attention of the administrator should have the **N** flag set, which causes the error logger to send the message to that person by way of the **mail** command. The priority level usually has no meaning in the error log, but does have meaning if the

message is also sent to a tracer process.

Once initiated, the **strerr** daemon continues to execute until terminated by the user. Usually, the **strerr** daemon is executed asynchronously.

> **Note:** Only one **strerr** daemon at a time is permitted to open the STREAMS log driver. If a module or driver is generating a large number of error messages, running the error logger causes a degradation in STREAMS performance. If a large number of messages are generated in a short time, the log driver may not be able to deliver some of the messages. This situation is indicated by gaps in the sequence numbering of the messages in the log files.

Files

/var/adm/streams/error.*mm−dd*

Related Information

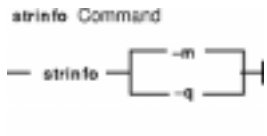
List of Streams Commands, STREAMS Overview**,** Understanding the log Device Driver in *AIX Version 4.3 Communications Programming Concepts*.

strinfo Command

Purpose

Displays administrative information about STREAMS activity.

Syntax



strinfo −m | −q

Description

The **strinfo** command displays information for debugging purposes about STREAMS, drivers and modules, or stream heads and the STREAMS run queue.

Flags

−m Displays information on drivers and modules present in STREAMS.

−q Displays informations on active stream heads, and on the run queue which holds the STREAMS module and driver service procedures.

Examples

1. To display information about STREAMS drivers and modules in use, enter:

   ```
   strinfo −m
   ```

   This produces a listing similar to the following:

   ```
   Device: 'sad', dcookie 0xf, flags:0x4, str 0x19a69e8
   Device: 'slog', dcookie 0x10, flags:0x4, str 0x19a6c18
   Device: 'rs', dcookie 0x11, flags:0x2, str 0x19bcb00
   Module: 'bufcall', flags:0x1, str 0x19a5c00
   Module: 'ldterm', flags:0x0, str 0x19cc858
   ```

   In this example `dcookie` indicates the major number, `flags` indicates the flags configuration, and `str` is the STREAMS table address.

2. To display information about active stream heads and the STREAMS run queue, enter:

   ```
   strinfo −q
   ```

   This produces a listing similar to the following:

   ```
   Active Stream Heads
   sth      sth_dev  sth_rq   sth_wq   sth_flag rq->q_first
   05a7ee00 00110001 05ad7000 05ad7074 00000818 00000000
   ```

```
STREAMS Service Queue
Queue 0x5ad7000 Flags 0x10
```

File

/usr/sbin/strinfo Contains the **strinfo** command.

Related Information

List of Streams Commands in *AIX Communications Programming Concepts*.

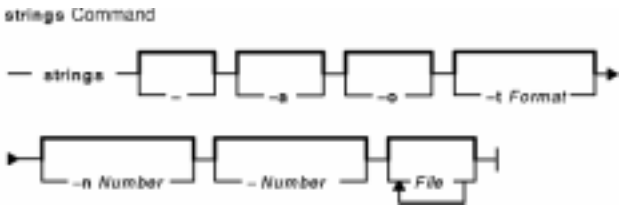
STREAMS Overview in *AIX Communications Programming Concepts*.

strings Command

Purpose

Finds the printable strings in an object or binary file.

Syntax



strings [**−a**] [**−**] [**−o**] [**−t** *Format*] [**−n** *Number*] [*−Number*] [*File ...*]

Description

The **strings** command looks for printable strings in an object or binary file. A string is any sequence of 4 or more printable characters that end with a new−line or a null character. The **strings** command is useful for identifying random object files.

Flags

−a or **−** Searches the entire file, not just the data section, for printable strings.

−n*Number* Specifies a minimum string length other than the default of 4 characters. The maximum value of a string length is 4096. This flag is identical to the *−Number* flag.

−o Lists each string preceded by its octal offset in the file. This flag is identical to the **−t o** flag.

−t*Format* Lists each string preceded by its offset from the start of the file. The format is dependent on the character used as the *Format* variable.

d Writes the offset in decimal.

o Writes the offset in octal.

x Writes the offset in hexadecimal.

Note: When the **−o** and the **−t***Format* flags are defined more than once on a command line, the last flag specified controls the behavior of the **strings** command.

−Number Specifies a minimum string length other than the default of 4 characters. The maximum value of a string length is 4096. This flag is identical to the **−n** *Number* flag.

File Binary or object file to be searched.

Exit Status

This command returns the following exit values:

0 Specifies that the command ran successfully.

>0 Specifies that an error occurred.

Examples

1. To search a file, enter:

   ```
   strings strings
   ```

   The **string** command displays:

   ```
   @(#)56
   1.17 com/cmd/scan/strings.c, cdmscan, bos320 5/7/92 10:21:20
   Standard input
   strings.cat
   /usr/mbin/strings
   Usage: strings [-a | -] [-o] [-t format] [-n | -#] [file...]
   Usage: strings [-a | -] [-o] [-t format] [-n | -#] [file...]
   Usage: strings [-a | -] [-o] [-t format] [-n | -#] [file...]
   Usage: strings [-a | -] [-o] [-t format] [-n | -#] [file...]
   Usage: strings [-a | -] [-o] [-t format] [-n | -#] [file...]
   %7o
   %7d
   %7x
   %7o
   %7d
   ```

2. To search for strings at least 12 characters long, enter:

   ```
   strings -12 strings
   ```

   The **string** command displays:

   ```
   1.17 com/cmd/scan/strings.c, cdmscan, bos320 5/7/92 10:21:20
   Standard input
   /usr/mbin/strings
   Usage: strings [-a | -] [-o] [-t format] [-n | -#] [file...]
   Usage: strings [-a | -] [-o] [-t format] [-n | -#] [file...]
   Usage: strings [-a | -] [-o] [-t format] [-n | -#] [file...]
   Usage: strings [-a | -] [-o] [-t format] [-n | -#] [file...]
   Usage: strings [-a | -] [-o] [-t format] [-n | -#] [file...]
   ```

3. To search for strings at least 20 characters long and show the offset in hexadecimal, enter:

   ```
   strings -t x -n 20 strings
   ```

   The **string** command displays:

   ```
   1017 1.17 com/cmd/scan/strings.c, cmdscan, bos320 5/7/92 10:21:20
   108c Usage: strings [-a | -] [-o] [-t format] [-n | -#] [file...]
   10d8 Usage: strings [-a | -] [-o] [-t format] [-n | -#] [file...]
   1124 Usage: strings [-a | -] [-o] [-t format] [-n | -#] [file...]
   1170 Usage: strings [-a | -] [-o] [-t format] [-n | -#] [file...]
   11bc Usage: strings [-a | -] [-o] [-t format] [-n | -#] [file...]
   ```

Related Information

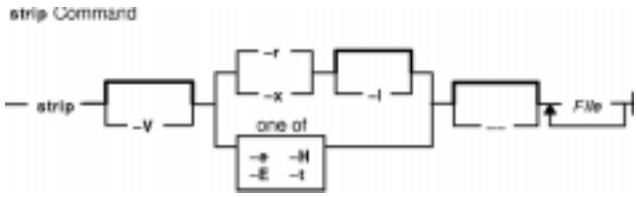
The **od** command.

strip Command

Purpose

Reduces the size of an Extended Common Object File Format (XCOFF) object file by removing information used by the binder and symbolic debug program.

Syntax



strip [**−V**] [**−r** [**−l**] | **−x** [**−l**] | **−t** | **−H** | **−e** | **−E**] [**−X** {**32**|**64**|**32_64**}] [**−−**] *File ...*

Description

The **strip** command reduces the size of XCOFF object files. The **strip** command optionally removes the line number information, relocation information, the debug section, the typchk section, the comment section, file headers, and all or part of the symbol table from the XCOFF object files. Once you use this command, symbolic debugging of the file is difficult; therefore, you should normally use the **strip** command only on production modules that you have debugged and tested. Using the **strip** command reduces the storage overhead required by an object file.

For each object module, the **strip** command removes information as specified by the supplied options. For each archive file, the **strip** command removes the global symbol table from the archive.

You can restore a stripped symbol table to an archive or library file by using the **ar −s** command.

The **strip** command with no options removes the line number information, relocation information, symbol table, the debug section, and the typchk section, and the comment section.

Flags

−e Sets the **F_LOADONLY** flag in the optional header of the object file. If the object file is placed in an archive, this flag indicates to the binder (**ld** command) that symbols in the object file should be ignored when linking with the archive.

−E Resets (turns off) the **F_LOADONLY** bit in the optional header of the object file. (See **−e** flag).

−H Removes the object file header, any optional header, and all section headers.
Note: Symbol Table information is not removed.

−l (Lowercase L) Strips the line number information from the object file.

−r Removes all symbol table information except those entries for external and static symbols. Does not remove the relocation information. Also removes the debug and typchk sections. This option produces an object file that can still be used as input to the linkage editor (**ld** command).

−t Removes most symbol table information but does not remove function symbols or line number information.

−V Prints the version number of the **strip** command.

−x Removes the symbol table information but does not remove static or external symbol information. The **−x** flag also removes relocation information, therefore linking to the file would not be possible.

−X*mode* Specifies the type of object file **strip** should examine. The *mode* must be one of the following:

32 Processes only 32−bit object files

64 Processes only 64−bit object files

32_64 Processes both 32−bit and 64−bit object files

The default is to process 32−bit object files (ignore 64−bit objects). The *mode* can also be set with the **OBJECT_MODE** environment variable. For example, **OBJECT_MODE=64** causes **strip** to process any 64−bit objects and ignore 32−bit objects. The **−X** flag overrides the **OBJECT_MODE** variable.

−− (Double hyphen) Interprets all arguments following this flag as file names. This allows you to strip files whose names start with a hyphen.

Exit Status

This command returns the following exit values:

0 Successful completion.

>0 An error occurred.

Examples

1. To remove the symbol table and line number information from the **a.out** file, enter:

   ```
   strip a.out
   ```

2. To remove the object file header of the **a.out** file, enter:

   ```
   strip -H a.out
   ```

3. To remove both the 32−bit and 64−bit symbol tables from **lib.a**, enter:
   ```
   strip -X 32-64 lib.a
   ```

Files

/usr/ccs/bin/strip Contains the **strip** command.

Related Information

The **ar** command, **as** command, **dump** command, **ld** command, **size** command.

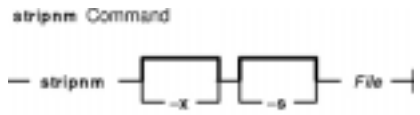
The **ar** file, **a.out** file.

stripnm Command

Purpose

Displays the symbol information of a specified object file.

Syntax



stripnm [**−x**] [**−s**] *File*

Description

The **stripnm** command (when run without the **−s** flag) prints the symbol table of a specified object file to standard output. The file specified by the *File* parameter can be a single object file or an archive library of object files. If the file specified by the *File* parameter is an archive, a listing for each object file in the archive is produced. If the symbol table has been stripped from the object file, the **stripnm** command extracts symbol names from the traceback tables (even if the **−s** flag is not specified). If the traceback tables do not exist, an error message is displayed.

> **Note:** The **stripnm** command does not list all symbols from the symbol table. Only file names and named and unnamed external symbols are reported. When used with the **−s** flag, the **stripnm** command will also suppress printing of some duplicated symbols in the symbol table.

Each symbol name is followed by its address (a series of blanks if the address is undefined) and the type of class and section type. The address field can be displayed as a decimal (the default value) or hexadecimal (if the **−x** flag is specified).

When run using the **−s** flag, the **stripnm** command extracts routine names first from the traceback tables and then from the symbol table, if it exists. Traceback tables are found at the end of routines, and contain the symbolic names of these routines. Routines defined as static do not appear in the symbol table, but may have traceback tables.

The **stripnm** command can also search for the glue code. The glue code is a set of executable instructions in the object file. In the Text section of the object file, the glue code is composed of the following sequence of instructions:

8182xxxx
: xxxx is offset in the table of contents (TOC), and can be any string.

90410014
: /* st r2, 14(r1) */

800c0000
: /* 1 r0, 0(r12) 8*/

804c004
: /* 1 r2, 4(r12) */

7c0903a6
: /* mtctr r0 */

4e800420
/* bctr */

The **stripnm** command searches the Text section from beginning to end for this sequence. If the command finds a sequence of instructions that matches, it is reported as glue code.

The **stripnm** command can also be used to search for symbol information in the **/unix** file. If the **/unix** file does not correspond to the currently running kernel, a warning message is displayed.

Note: Only the root user and members of the security group should have execute (x) access to this command.

Flags

−x	Prints symbol address values in hexadecimal format.
−s	If the symbol table does not exist, this flag displays symbol names from the traceback tables. If the symbol table exists, the **−s** flag first displays the symbol names from the traceback tables, and then displays symbol names found in the symbol table but not found in the traceback tables.

Examples

1. To list the symbols of the **a.out** object file, enter:
   ```
   stripnm a.out
   ```
2. To list the symbols address values of the **a.out** object file in hexadecimal mode, enter:
   ```
   stripnm −x a.out
   ```
3. To list symbols from the traceback tables and symbol table (if it exists) of the **a.out** object file, enter:
   ```
   stripnm −s a.out
   ```

Related Information

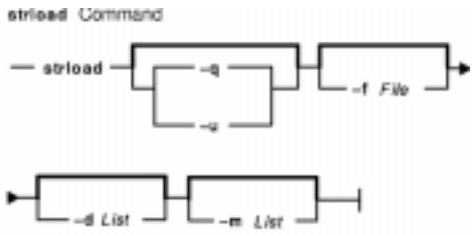
The **strip** command.

strload Command

Purpose

Loads and configures Portable Streams Environment (PSE).

Syntax



strload [**−u** | **−q**] [**−f***File*] [**−d***List*] [**−m***List*]

Description

The **strload** command enables the system administrator to load and unload drivers and modules and to query the load status of PSE and its dependents.

By default, the **strload** command loads PSE according to the **/etc/pse.conf** file. The **−f** flag allows the administrator to use an alternate configuration file. The **−d** and **−m** flags are used to specify drivers and modules that are not present in the configuration files (such as when new drivers are being developed). The **−q** flag reports on the system load status (kernel existence) of the referenced drivers and modules.

Configuration File

The configuration file is a flat ASCII, line−oriented database. Comments are introduced by a # (pound sign), and continue until the end of the line. Blank lines are ignored. The form for each record is:

```
attributes filename [argument [node [minor ...] ] ]
```

Fields are separated by spaces, tabs, or both. A − (dash) can be specified as the field value, indicating that the default value is to be used. The fields are defined as follows:

`attributes` Describes the extension to load. The acceptable values are:

- **d** Specifies a driver.
- **m** Specifies a module.
- **s** Creates the node as a standard (not cloned) device.
- **+** Specifies that the extension can be configured more than once. This value must be specified for all lines containing the extension file name.

`filename` Specifies the object file containing the extension. If the command is issued with a "/" (slash) in the filename of the driver or module to be loaded, unloaded or queried, the **strload** command uses the value in the filename field explicitly. If there is no "/" in the filename entry, the **strload** command will first look for a copy of the driver or module in the current directory. If the driver or module is not in the current directory, **strload** will look for the driver or module in the **/usr/lib/drivers/pse** directory.

Note: It is recommended that the **strload** command be issued from the root directory (/). The **strload** command for load, unload, and query should always be issued from the same directory.

The AIX kernel extension loader REQUIRES that the pathnames used be identical in load, unload and queries. This, coupled with the way the filename is determined by **strload**, could cause problems. Every byte in the pathname used by the **strload** command must EXACTLY match every positionally corresponding byte in the pathname used by the AIX kernel extension loader because the kernel does a **strcmp()** on the filename when looking for matches. If the **strload** command is issued from a different directory to unload the module or driver, one of the following will occur:

- If the **strload** command does not find a copy of the driver or module in the new current directory, **strload** will attempt to unload the driver or module in the **/usr/lib/drivers/pse** directory. However, this pathname may not be the same as the pathname that the loader has logged for that driver or module. If the pathname is not the same, the **strload** command will fail.
- If the **strload** command finds another copy of the module or driver in the new current directory, then the pathnames will be the same, and the loader will correctly unload the driver or module that was loaded. Thus, the **strload** command succeeds, but the results may not be as the user intended.

For example:

The following scenario (NOT recommended) will cause "spx", also known as "A", to be unloaded. This is probably not the desired effect.

```
mkdir /tmp/foo /tmp/bar
cp /usr/lib/drivers/pse/spx /tmp/foo/A
cp /bin/ls /tmp/bar/A
cd /tmp/foo
strload −d A      # The loader knows the path and filename as
                  # "A" because "A" is found in the current
                  # directory
cd /tmp/bar
strload −q −d A   # Reports "yes" because there is "A" in the
                  # current directory. Note that the file "A"
                  # in /tmp/bar is NOT the same file "A" in
                  # /tmp/foo, but the loader does not care
                  # because it identifies the file by
                  # pathname.
strload −u −d A   # Unloads spx (also known as "A")!
```

The following is an error scenario:

```
mkdir /tmp/foo2 /tmp/bar2
cp /usr/lib/drivers/pse/spx /tmp/foo2/A
cd /tmp/foo2
strload −d A     # The loader knows the path and filename as
                 # "A"because "A" is found in the current
                 # directory.
cd /tmp/bar2
strload −q −d A  # Answers "no". There is no filename
                 # in /tmp/bar2 that matches "A", so strload
                 # prepends pathname "/usr/lib/drivers/pse" to
                 # "A". "/usr/lib/drivers/pse/A" is not found,
                 # so strload answers "no".
strload −u −d A  # Fails − "A" does not exist.
```

The following is an error scenario:

```
cd /usr/lib/drivers/pse
```

```
strload -d spx   # The loader knows the path and filename as
                 # "spx" because "spx" is found in the
                 # current directory.
cd /
strload -q -d spx # Answers "no". There is no filename in /
                  # that matches "spx", so strload prepends
                  # the pathname "/usr/lib/drivers/pse" to
                  # "spx". "/usr/lib/drivers/pse/spx" is found
                  # since it exists, so strload gives
                  # "/usr/lib/drivers/pse/spx" to the loader.
                  # The strcmp() fails since
                  # "/usr/lib/drivers/pse/spx" and "spx" do
                  # not match exactly.
strload -u -d spx # Fails - "spx" does not exist.
```

`argument` Has no meaning for the **strload** command. This field is optional. It is passed to the extension when its configuration routine is called. Its interpretation is specific to that extension. The default argument is the value of the `filename` field.

`node` Specifies the name of the node to create. This field is optional. It applies only to drivers and is used as the created node name when the driver is loaded. By default, the created node is **/dev/**`filename`.

`minor` Specifies additional, non clone nodes to create for this driver. This field is optional. The node names are created by appending the minor number to the cloned driver node name. No more than five minor numbers can be given (from 0 to 4), and a node is created for each one.

The **-d** and **-m** flags cause the configuration file to be ignored, unless it is explicitly named on the command line, as follows:

```
strload -f /tmp/my.conf -d newdriver
```

> **Note:** The **-d** and **-m** flags do not override the configuration file. That is, if driver **dgb** is loaded by using the configuration file, the **-d** flag will attempt to reload it but will fail. The configuration file is processed before the **-d** and **-m** flags.

The *List* variable for the **-d** and **-m** flags is a comma-separated list of file names, each of which contains a single PSE driver or module. The configuration process proceeds as if a line of one of the following forms was found in the configuration file:

```
d filename
```

```
m filename
```

Flags

-d *List* Lists PSE device drivers to load or unload. The *List* variable specifies a comma-separated list of driver object names.

-f *File* Configures PSE according to the configuration information contained in the file indicated by the *File* variable. The default configuration file is **/etc/pse.conf**.

-m *List* Lists PSE modules to load or unload. The *List* variable specifies a comma-separated list of module object names.

-q Reports load status of extensions.

-u Unloads extensions.

Examples

1. Entering the following command loads PSE (if not already loaded), the `dgb` and `ssb` drivers from the **/usr/lib/drivers/pse/** directory, and the `aoot` module from the current directory, but does not use the configuration file:

   ```
   root# strload −d dgb,ssb −m ./aoot
   ```

2. To unload the `aoot` module only, enter:

   ```
   root# strload −u −m ./aoot
   ```

3. Entering the following command asks if the `spx` driver exists:

   ```
   root# strload −q −d
   spx
   ```

   and produces the following output if not:

   ```
   spx
   : no
   ```

4. The following is an example configuration file:

   ```
   #example configuration file
   d       dgb                                     #line 1
   d       mux    −       −       0                #line 2
   ds      foo                                     #line 3
   d+      xtiso  tcp     /dev/xti/tcp             #line 4
   d+      xtiso  udp     /dev/xti/udp             #line 5
   m       aoot                                    #line 6
   ```

   Line 1 loads the `dgb` driver extension as a cloned device named `/dev/dgb`. The argument passed to the `dgb` configuration routine is `dgb`.

   Line 2 loads the `mux` driver extension as a cloned device named `/dev/mux` and also creates a standard device name `/dev/mux0` with a minor number of 0 (zero). (No more than five device names can be created with minor numbers from 0 to 4.)

   Line 3 loads the `foo` driver extension as a standard device (not cloned) named `/dev/foo`. The minor number is 0.

   Lines 4 and 5 load the `xtiso` driver extension, and configure it twice: once as `tcp` and once as `udp`. The clone nodes created are `/dev/xti/tcp` and `/dev/xti/udp`. The configuration routine of `xtiso` is called twice: once with the argument `tcp`, and once with `udp`.

   Line 6 loads the `aoot` module extension. No node is created, and the configuration routine is passed the value `aoot`.

Files

/usr/lib/drivers/pse/*	Contains PSE kernel extensions.
/etc/pse.conf	Default PSE configuration file.
/**usr**/**sbin**/**strload**	Contains the **strload** command.

Related Information

The **slibclean** command, **strerr** command.

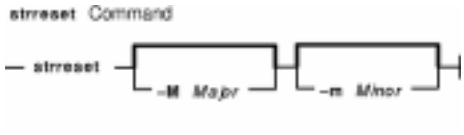
Configuring Drivers and Modules in the Portable Streams Environment (PSE), List of Streams Commands, STREAMS Overview in *AIX Version 4.3 Communications Programming Concepts*.

strreset Command

Purpose

Resets a stream. This command only applies to AIX Version 4.2 or later.

Syntax



strreset [**−M** *Major*] [**−m** *Minor*]

Description

The **strreset** command resets an open stream by generating an M_FLUSH message to the stream head. You use it mainly to reset blocked streams. When it is impossible to reopen the stream, issue an I_FLUSH ioctl(), or equivalent command. This situation may happen with a process sleeping in a module's close routine, when signals can not be sent to the process (a zombie process exiting, for example).

Flags

−M *Major* Specifies the major number for the special file associated with the stream to be reset.
−m *Minor* Specifies the minor number for the special file associated with the stream to be reset.

Exit Status

This command returns the following exit values:

0 Successful completion.
>0 An error occurred.

Security

Access Control: You must have root authority to run this command.

Auditing Events: N/A

Files

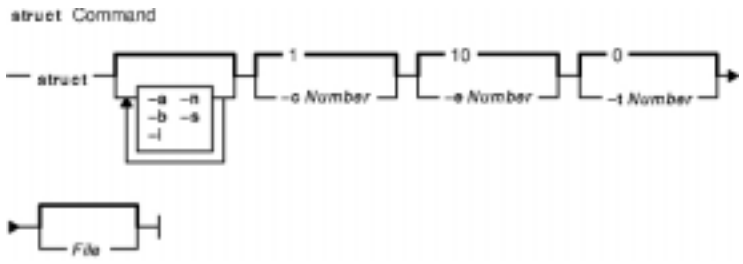
/usr/sbin/strreset Contains the **strreset** command.

struct Command

Purpose

Translates a FORTRAN program into a RATFOR program.

Syntax



struct [**−s**] [**−i**] [**−a**] [**−b**] [**−n**] [**−t***Number*] [**−c***Number*] [**−e***Number*] [*File*]

Description

The **struct** command translates the FORTRAN program specified by *File* (standard input default) into a RATFOR program. Wherever possible, RATFOR control constructs replace the original FORTRAN. Statement numbers appear only where still necessary. Cosmetic changes are made, including changing Hollerith strings into quoted strings and relational operators into symbols (for example, **.GT.** into >). The output is appropriately indented.

The **struct** command knows FORTRAN 66 syntax, but not full FORTRAN 77. If an input FORTRAN program contains identifiers that are reserved words in RATFOR, the structured version of the program will not be a valid RATFOR program. The labels generated cannot go above 32767. If you get a **goto** statement without a target, try using the **−e** flag.

Flags

−a

Turn sequences of **else−if** statements into a non−RATFOR switch of the form:

```
switch
{   case pred1: code
    case pred2: code
    case pred3: code
    default: code
         }
```

The **case** predicates are tested in order. The code appropriate to only one case is executed. This generalized form of **switch** statement does not occur in RATFOR.

−b

Generates **goto** statements instead of multilevel **break** statements.

−c*Number*

Increments successive labels in the output program by the nonzero integer *Number*. The default is 1. Do not insert a space between **−c** and *Number*.

−e*Number*

If *Number* is 0 (default), places code within a loop only if it

	can lead to an iteration of the loop. Do not insert a space between **−e** and *Number*.
−i	Do not turn computed **goto** statements into switches. (RATFOR does not turn switches back into computed **goto** statements.)
−n	Generates **goto** statements instead of multilevel **next** statements.
−s	Input is accepted in standard format. Comments are specified by a **c**, **C**, or ***** in column 1, and continuation lines are specified by a nonzero, nonblank character in column 6. Input is in the form accepted by the **f77** command.
−t*Number*	Makes the nonzero integer *Number* the lowest valued label in the output program. The default is 10. Do not insert a space between **−t** and *Number*.

If *Number* is nonzero, admits small code segments to a loop if otherwise the loop would have exits to several places including the segment, and the segment can be reached only from the loop. In this case, small is close to, but not equal to, the number of statements in the code segment. Values of *Number* under 10 are suggested.

Examples

To translate the `test.f` FORTRAN program into the `newtest.ratfor` RATFOR program, enter:

```
struct −s −i −n −t2 test.f > newtest.ratfor
```

Files

/tmp/struct*	Temporary files used during processing of the **struct** command.
/usr/lib/struct/structure	File that handles processing for the **struct** command.
/usr/lib/struct/beautify	File that handles processing for the **struct** command.
/usr/ucb/struct	Contains the **struct** command.

Related Information

The **asa** or **fpr** command, **fsplit** command.

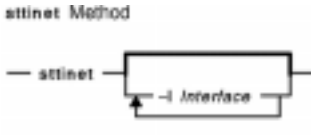
The Commands Overview in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

sttinet Method

Purpose

Enables the inet instance.

Syntax



sttinet [**−l***Interface* ...]

Description

The **sttinet** method enables the inet instance by calling the **ifconfig** command and sets the status flag of the inet instance to AVAILABLE.

> **Note:** The **sttinet** method is a programming tool and should not be executed from the command line.

Flags

−l*Interface* ... Specifies which specific interface to enable. If no interfaces are specified, then all configured interfaces are started.

Examples

The following method enables the inet instance:

```
sttinet −l tr0 −l tr1
```

Related Information

The **ifconfig** command, **mkdev** command.

The **odm_run_method** subroutine.

Writing a Device Method in *AIX Version 4.3 Kernel Extensions and Device Support Programming Concepts*.

Object Data Manager (ODM) Overview for Programmers in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

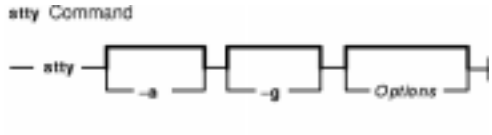
Understanding Network Interfaces in *AIX Version 4.3 Communications Programming Concepts*.

stty Command

Purpose

Sets, resets, and reports workstation operating parameters.

Syntax



stty [**−a**] [**−g**] [*Options*]

Description

The **stty** command sets certain I/O options for the device that is the current standard input. This command writes output to the device that is the current standard output.

This version of AIX uses the standard XPG4 interface to control the terminals, maintaining a compatibility with POSIX and BSD interfaces. The **stty** command supports both POSIX and BSD compliant options, but the usage of POSIX options is strongly recommended. A list of obsolete BSD options, with the corresponding POSIX options, is also provided.

When you redirect standard input from a tty device by entering

```
stty −a </dev/ttyx
```

the **stty** command (POSIX) will hang while waiting for the **open()** of that tty until the RS−232 carrier detect signal has been asserted. Exceptions to this rule occur if the **clocal** or **forcedcd** (128−port only) option is set.

Flags

−a Writes the current state of all option settings to standard output.

−g Writes option settings to standard output in a form usable by another **stty** command.

Options

The **stty** command supports following categories of options:

- Control Modes
- Input Modes
- Output Modes
- Local Modes
- Hardware Flow Control Modes
- Control Character Assignments
- Combination Modes
- Window Size

Control Modes

clocal	Assumes a line without modem control.
−clocal	Assumes a line with modem control.
cread	Enables the receiver.
−cread	Disables the receiver.
cstopb	Selects 2 stop bits per character.
−cstopb	Selects 1 stop bit per character.
cs5, **cs6**, **cs7**, **cs8**	Selects character size.
hup, **hupcl**	Hangs up dial−up connection on the last close.
−hup, **−hupcl**	Does not hang up dial−up connection on the last close.
parenb	Enables parity generation and detection.
−parenb	Disables parity generation and detection.
parodd	Selects odd parity.
−parodd	Selects even parity.
0	Hangs up phone line immediately.
speed	Sets the workstation input and output speeds to the specified *speed* number of bits per second. All speeds are not supported by all hardware interfaces. Possible values for *speed* are: **50**, **75**, **110**, **134**, **200**, **300**, **600**, **1200**, **1800**, **2400**, **4800**, **9600**, **19200**, **19.2**, **38400**, **38.4**, **exta**, and **extb**. **Note: exta**, **19200**, and **19.2** are synonyms; **extb**, **38400**, and **38.4** are synonyms.
ispeed*speed*	Sets the workstation input speed to the specified *speed* number of bits per second. All speeds are not supported by all hardware interfaces, and all hardware interfaces do not support this option. Possible values for *speed* are the same as for the *speed* option.
ospeed*speed*	Sets the workstation output speed to the specified *speed* number of bits per second. All speeds are not supported by all hardware interfaces, and all hardware interfaces do not support this option. Possible values for *speed* are the same as for the *speed* option.

Input Modes

brkint	Signals INTR on break.
−brkint	Does not signal INTR on break.
icrnl	Maps CR to NL on input.
−icrnl	Does not map CR to NL on input.
ignbrk	Ignores BREAK on input.
−ignbrk	Does not ignore BREAK on input.
igncr	Ignores CR on input.
−igncr	Does not ignore CR on input.
ignpar	Ignores parity errors.
−ignpar	Does not ignore parity errors.
inlcr	Maps NL to CR on input.
−inlcr	Does not map NL to CR on input.
inpck	Enables parity checking.
−inpck	Disables parity checking.
istrip	Strips input characters to 7 bits.
−istrip	Does not strip input characters to 7 bits.
iuclc	Maps uppercase alphabetic characters to lowercase.
−iuclc	Does not map uppercase alphabetic characters to lowercase.
ixany	Allows any character to restart output.
−ixany	Allows only the START (the Ctrl−Q key sequence) to restart output.

ixoff	Sends START/STOP characters when the input queue is nearly empty/full.
−ixoff	Does not send START/STOP characters.
ixon	Enables START/STOP output control. Once START/STOP output control has been enabled, you can pause output to the workstation by pressing the Ctrl−S key sequence and resume output by pressing the Ctrl−Q key sequence.
−ixon	Disables START/STOP output control.
imaxbel	Echoes the BEL character and discards the last input character if input overflows.
−imaxbel	Discards all input if input overflows.
parmrk	Marks parity errors.
−parmrk	Does not mark parity errors.
Output Modes	
bs0, **bs1**	Selects style of delay for backspaces (**bs0** siginifes no delay).
cr0, **cr1**, **cr2**, **cr3**	Selects style of delay for CR characters (**cr0** siginifes no delay).
ff0, **ff1**	Selects style of delay for form feeds (**ff0** siginifes no delay).
nl0, **nl1**	Selects style of delay for NL characters (**nl0** siginifes no delay).
ofill	Uses fill characters for delays.
−ofill	Uses timing for delays.
ocrnl	Maps CR characters to NL characters.
−ocrnl	Does not map CR characters to NL characters.
olcuc	Maps lowercase alphabetic characters to uppercase on output.
−olcuc	Does not map lowercase alphabetic characters to uppercase on output.
onlcr	Maps NL characters to CR−NL characters.
−onlcr	Does not map NL characters to CR−NL characters.
onlret	On the terminal, NL performs the CR function.
−onlret	On the terminal, NL does not perform the CR function.
onocr	Does not output CR characters at column zero.
−onocr	Outputs CR characters at column zero.
opost	Processes output.
−opost	Does not process output; that is, ignores all other output options.
ofdel	Uses DEL characters for fill characters.
−ofdel	Uses NUL characters for fill characters.
tab0, **tab1**, **tab2**	Selects style of delay for horizontal tabs (**tab0** siginifes no delay).
tab3	Expands tab character to variable number of spaces.
vt0, **vt1**	Selects style of delay for vertical tabs (**vt0** siginifes no delay).
Local Modes	
echo	Echoes every character typed.
−echo	Does not echo characters.
echoctl	Echoes control characters as ^X (Ctrl−X), where X is the character given by adding 100 octal to the code of the control character.
−echoctl	Does not echo control characters as ^X (Ctrl−X).
echoe	Echoes the ERASE character as the "backspace space backspace" string. **Note:** This mode does not keep track of column position, so you can get unexpected results when erasing such things as tabs and escape sequences.
−echoe	Does not echo the ERASE character, just backspace.
echok	Echoes a NL character after a KILL character.

−echok	Does not echo a NL character after a KILL character.
echoke	Echoes the KILL character by erasing each character on the output line.
−echoke	Just echoes the KILL character.
echonl	Echoes the NL character.
−echonl	Does not echo the NL character.
echoprt	Echoes erased characters backwards with / (slash) and \ (backslash).
−echoprt	Does not echo erased characters backwards with / (slash) and \ (backslash).
icanon	Enables canonical input (canonical input allows input−line editing with the ERASE and KILL characters). See the discussion about canonical mode input in Line Discipline Module (ldterm) *AIX Version 4.3 Communications Programming Concepts*.
−icanon	Disables canonical input.
iexten	Specifies that implementation−defined functions shall be recognized from the input data. Recognition of the following control characters requires **iexten** to be set: **eol2, dsusp, reprint, discard, werase, lnext.** The functions associated with these modes also require **iexten** to be set: **imaxbel**, **echoke**, **echoprt**, and **echoctl**.
−iexten	Specifies that implementation−defined functions shall not be recognized from the input data.
isig	Enables the checking of characters against the special control characters INTR, SUSP and QUIT special control characters.
−isig	Disables the checking of characters against the special control characters INTR, SUSP and QUIT special control characters.
noflsh	Does not clear buffers after INTR, SUSP, or QUIT control characters.
−noflsh	Clears buffers after INTR, SUSP, or QUIT control characters.
pending	Causes any input that is pending after a switch from raw to canonical mode to be re−input the next time a read operation becomes pending or the next time input arrives. Pending is an internal state bit.
−pending	No text is pending.
tostop	Signals **SIGTOU** for background output.
−tostop	Does not signal **SIGTOU** for background output.
xcase	Echoes uppercase characters on input, and displays uppercase characters on output with a preceding \ (backslash).
−xcase	Does not echo uppercase characters on input.

Hardware Flow Control Modes

These options are extensions to the XPG4 standard.

cdxon	Enables CD hardware flow control mode on output.
−cdxon	Disables CD hardware flow control mode on output.
ctsxon	Enables CTS hardware flow control mode on output.
−ctsxon	Disables CTS hardware flow control mode on output.
dtrxoff	Enables DTR hardware flow control mode on input.
−dtrxoff	Disables DTR hardware flow control mode on input.
rtsxoff	Enables RTS hardware flow control mode on input.
−rtsxoff	Disables RTS hardware flow control mode on input.

Control Assignments

To assign a control character to a character string, enter:

```
stty Control String
```

where the `Control` parameter may be the INTR, QUIT, ERASE, KILL, EOF, EOL, EOL2, START, STOP, SUSP, DSUSP, REPRINT, DISCARD, WERASE, LNEXT, MIN, or TIME character. (Use the MIN and TIME characters with the **−icanon** option.)

> **Note:** The values for MIN and TIME are interpreted as integer values, not as character values.

The `String` parameter may be any single character such as c. An example of this control assignment is:

```
stty STOP c
```

Another way of assigning control characters is to enter a character sequence composed of a \^ (backslash, circumflex) followed by a single character. If the single character after the ^ (circumflex) is one of the characters listed in the ^c (circumflex c) column of the following table, the corresponding control character value will be set. For example, to assign the DEL control character by using the ? (question mark) character, enter the string `\^?` (backslash, circumflex, question mark), as in:

```
stty ERASE \^?
```

Circumflex Control Characters in stty					
^c	**Value**	**^c**	**Value**	**^c**	**Value**
a, A	<SOH>	**l, L**	<FF>	**w, W**	<ETB>
b, B	<STX>	**m, M**	<CR>	**x, X**	<CAN>
c, C	<ETX>	**n, N**	<SO>	**y, Y**	<EM>
d, D	<EOT>	**o, O**	<SI>	**z, Z**	<SUB>
e, E	<ENQ>	**p, P**	<DLE>	**[**	<ESC>
f, F	<ACK>	**q, Q**	<DC1>	\	<FS>
g, G	<BEL>	**r, R**	<DC2>	**]**	<GS>
h, H	<BS>	**s, S**	<DC3>	^	<RS>
i, I	<HT>	**t, T**	<DC4>	**_**	<US>
j, J	<LF>	**u, U**	<NAK>	**?**	<DEL>
k, K	<VT>	**v, V**	<SYN>	**@**	<NUL>

Combination Modes	
cooked	See the **−raw** option.
ek	Sets ERASE and KILL characters to the Ctrl−H and Ctrl−U key sequences, respectively.
evenp	Enables **parenb** and **cs7**.

−evenp	Disables **parenb** and sets **cs8**.
lcase, **LCASE**	Sets **xcase**, **iuclc**, and **olcuc**. Used for workstations with uppercase characters only.
−lcase, **−LCASE**	Sets **−xcase**, **−iuclc**, and **−olcuc**.
nl	Sets **−icrnl** and **−onlcr**.
−nl	Sets **icrnl**, **onlcr**, **−inlcr**, **−igncr**, **−ocrnl**, and **−onlret**.
oddp	Enables **parenb**, **cs7**, and **parodd**.
−oddp	Disables **parenb** and sets **cs8**.
parity	See the **evenp** option.
−parity	See the **−evenp** option.
sane	Resets parameters to reasonable values.
raw	Allows raw mode input (no input processing, such as erase, kill, or interrupt); parity bit passed back.
−raw	Allows canonical input mode.
tabs	Preserves tabs.
−tabs, **tab3**	Replaces tabs with spaces when printing.

Window Size

cols*n*, **columns***n*	The terminal (window) size is recorded as having *n* columns.
rows*n*	The terminal (window) size is recorded as having *n* rows.
size	Prints the terminal (window) sizes to standard output (first rows and then columns).

Obsolete Options

The following BSD options are supported by the **stty** command. For each of them, the recommended POSIX option is given.

all	Use the **stty −a** command to display all current settings.
crt	Use the **sane** option to reset parameters to reasonable values.
crtbs	Use the **−echoe** option.
crterase	Use the **echoe** option.
−crterase	Use the **−echoe** option.
crtkill	Use the **echoke** option.
−crtkill	Use the **echok** and **−echoke** options.
ctlecho	Use the **echoctl** option.
−ctlecho	Use the **−echoctl** option.
decctlq	Use the **−ixany** option.
−decctlq	Use the **ixany** option.
even	Use the **evenp** option.
−even	Use the **−evenp** option.
everything	Use the **stty −a** command to display all current settings.
litout	Use the **−opost** option.
−litout	Use the **opost** option.
odd	Use the **oddp** option.
−odd	Use the **−oddp** option.
pass8	Use the **−istrip** option.
−pass8	Use the **istrip** option.
prterase	Use the **echoprt** option.

speed Use the **stty** command to display current settings.

tandem Use the **ixoff** option.

−tandem Use the **−ixoff** option.

Examples

1. To display a short listing of your workstation configuration, enter:

   ```
   stty
   ```

   This lists settings that differ from the defaults.

2. To display a full listing of your workstation configuration, enter:

   ```
   stty −a
   ```

3. To enable a key sequence that stops listings from scrolling off the screen, enter:

   ```
   stty ixon ixany
   ```

   This sets **ixon** mode, which lets you stop runaway listing by pressing the Ctrl−S key sequence. The **ixany** flag allows you to resume the listing by pressing any key. The normal workstation configuration includes the **ixon** and **ixany** flags, which allows you to stop a listing with the Ctrl−S key sequence that only the Ctrl−Q key sequence will restart.

4. To reset the configuration after it has been messed up, enter:

   ```
   Ctrl−J stty sane Ctrl−J
   ```

   Press the Ctrl−J key sequence before and after the command instead of the Enter key. The system usually recognizes the Ctrl−J key sequence when the parameters that control Enter key processing are messed up.

   Sometimes the information displayed on the screen may look strange, or the system will not respond when you press the Enter key. This can happen when you use the **stty** command with parameters that are incompatible or that do things you don't understand. It can also happen when a screen−oriented application ends abnormally and does not have a chance to reset the workstation configuration.

   Entering the **stty sane** command sets a reasonable configuration, but it may differ slightly from your normal configuration.

5. To save and restore the terminal's configuration:

   ```
   OLDCONFIG=`stty −g`          # save configuration
   stty −echo                   # do not display password
   echo "Enter password: \c"
   read PASSWD                  # get the password
   stty $OLDCONFIG              # restore configuration
   ```

   This command saves the workstation's configuration, turns off echoing, reads a password, and restores the original configuration.

   Entering the **stty**−*echo* command turns off echoing, which means that the password does not appear on the screen when you type it at the keyboard. This action has nothing to do with the **echo** command, which displays a message on the screen.

File

/usr/bin/stty Contains the **stty** command.

Related Information

The **terminfo** file, **tty** special file **termios.h** header file.

National Language Support Overview for Programming and TTY Subsystem Overview in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

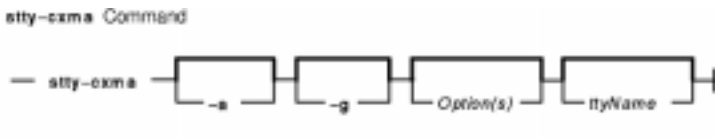
ldterm Line Discipline in *AIX General Programming Concepts: Writing and Debugging Programs*.

stty−cxma Command

Purpose

Sets and reports the terminal options for a TTY configuration of the 128−port asynchronous subsystem.

Syntax



stty−cxma [**−a**] [**−g**] [*Option(s)*] [*ttyName*]

Description

If no flags or options are specified, the **stty−cxma** command reports all 128−port special driver settings and modem signals, as well as all standard parameters reported by the **stty** command for the tty device that is the current standard input.

The *ttyName* parameter can be specified to set or report options for a tty device for other than the standard input. The *ttyName* parameter can be a simple tty name, such as **tty0**, or can be prefixed by **/dev/**, such as **/dev/tty0**. This option may be used on a modem control line when no carrier is present.

Further options can be specified to change flow control settings, set transparent print options, force modem control lines, and display all tty settings. Unrecognized options are passed to the **stty** command for interpretation.

Flags

−a Writes all the unique 128−port settings as well as all the standard tty settings reported by **stty −a** to standard output.

−g Writes option settings to standard output in a form usable by another stty command.

Options

The following options specify transient actions to be performed immediately:

break	Sends a 250 MS break signal out on the tty line.
flush	Discards tty input and output immediately.
flushin	Discards tty input only.
flushout	Discards tty output only.

The actions specified by the following options are in effect until the device is closed. The next time the device is opened, default values are used.

dtr	Raises the DTR modem control line, unless DTR hardware flow control is selected.
−dtr	Drops the DTR modem control line, unless DTR hardware flow control is selected.

rts Raises the RTS modem control line, unless RTS hardware flow control is selected.

−rts Drops the RTS modem control line, unless RTS hardware flow control is selected.

startin Releases flows control to resume stopped input.

startout Restarts stopped output exactly as if an XON character was received.

stopin Activates flow control to stop input.

stopout Stops output exactly as if an XOFF character was received.

2200flow Enables 2200 style flow control on the port. The 2200 terminals support an attached printer and use the following four flow control characters:

0xF8 terminal XON

0xF9 printer XON

0xFA terminal XOFF

0xFB printer XOFF

−2200flow Disables 2200 style flow control on the port.

2200print Runs flow control for the terminal and flow control for the transparent print device (as set by the **2200flow** option) independently.

−2200print Runs terminal and printer flow control (as set by the **2200flow** option) together. So if either the terminal or the printer XOFF character is received, all output is paused until the matching XON character is received.

altpin Switches the location of the DSR and DCD inputs on the modular connector, so that DCD is available when using an 8−pin RJ45 connector instead of the 10−pin RJ45 connector.

−altpin Restores the availability of DSR when using the 10−pin RJ45 connector.

aixon Enables auxiliary flow control, so that two unique characters are used for XON and XOFF. If both XOFF characters are received, transmission will not resume until both XON characters are received.

−aixon Disables auxiliary flow control.

astartc*c* Sets auxiliary XON flow control character. The character may be given as a decimal, octal, or hexadecimal number.

astopc*c* Sets auxiliary XOFF flow control character. The character may be given as a decimal, octal, or hexadecimal number.

bufsize*n* Sets the driver's estimate of the size of the transparent printer's input buffer. After a period of inactivity, the driver bursts this many characters to the transparent printer before reducing to the maximum CPS rate specified by the **maxcps** option rate selected above. The default value is 100 characters.

ctspace Enables CTS hardware output flow control, so local transmission pauses when CTS drops.

−ctspace Disables CTS hardware output flow control.

dcdpace Enables DCD hardware output flow control, so local transmission pauses when DCD drops.

−dcdpace Disables DCD hardware output flow control.

dsrpace Enables DSR hardware output flow control, so local transmission pauses when DSR drops.

−dsrpace Disables DSR hardware output flow control.

dtrpace Enables DTR hardware input flow control, so DTR drops to pause remote transmission.

−dtrpace Disables DTR hardware input flow control.

edelay*n* Sets the rate at which the128−port asynchronous adapter wakes up the driver on input. The adapter wakes the driver every *n* milliseconds. The default value is 100 milliseconds.

fastbaud Alters the baud rate table, so 50 baud becomes 57600 baud.

−fastbaud Restores the baud rate table, so 57500 baud becomes 50 baud.

fastcook Performs cooked output processing on the128−port asynchronous adapter to reduce host CPU

usage and increase raw mode input performance.

−fastcook Disables cooked output processing.

forcedcd Disables carrier sense, so the tty may be opened and used even when the carrier is not present.

−forcedcd Reenables carrier sense.

maxchar*n* Sets the maximum number of transparent print characters the driver places in the output queue. Reducing this number increases system overhead; increasing this number delays operator keystroke echo times when the transparent printer is in use. The default value is 50 characters.

maxcps*n* Sets the maximum CPS (characters per second) rate at which characters are output to the transparent print device. The rate chosen should be just below the average print speed. If the number is too low, printer speed is reduced. If the number is too high, the printer resorts to flow control, and user entry on the CRT is impaired accordingly. The default value is 100 CPS.

offstr*s* Sets the CRT escape sequence to turn transparent print off. An arbitrary octal character *xxx* may be given as `\xxx`.

onstr*s* Sets the CRT escape sequence to turn transparent print on. An arbitrary octal character *xxx* may be given as `\xxx`.

rtspace Enables RTS hardware input flow control, so RTS drops to pause remote transmission.

−rtspace Disables RTS hardware input flow control.

startc*c* Sets the XON flow control character. The character may be given as a decimal, octal, or hexadecimal number.

stopc*c* Sets the XOFF flow control character. The character may be given as a decimal, octal, or hexadecimal number.

term*t* Sets the transparent printer on and off strings to values specified in the internal default table. Internal defaults are used for the following terminals:**adm31**, **ansi**, **dg200**, **dg210**, **hz1500**, **mc5**, **microterm**, **multiterm**, **pcterm**, **tvi**, **vp−a2**, **vp−60**, **vt52**, **vt100**, **vt220**, **wyse30**, **wyse50**, **wyse60**, or **wyse75**. If the terminal type is not found in the internal default table, the transparent print on and off strings are set to the values specified by the **po** and **pf** attributes in the **termcap** file.

Examples

1. To display all the unique 128−port settings as well as all the standard tty settings for a tty port configured on a 128−port asynchronous controller as `/dev/tty0`, enter:

   ```
   stty−cxma −a tty0
   ```

2. To make DCD available when using an 8−pin RJ45 connector for a tty port configured on a 128−port asynchronous controller as `/dev/tty3`, enter:

   ```
   stty−cxma altpin tty3
   ```

This command interchanges the location of the DSR and DCD inputs on the modular connector.

Files

/usr/ebin/tty/stty−cxma Contains the **stty−cxma** command.

Related Information

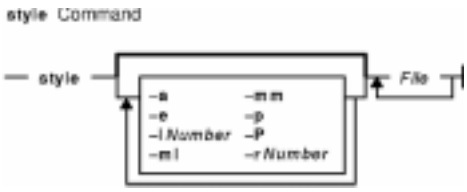
The **stty** command.

style Command

Purpose

Analyzes surface characteristics of a document.

Syntax



style [**−a**] [**−e**] [**−l***Number*] [**−ml**] [**−mm**] [**−p**] [**−P**] [**−r***Number*] *File ...*

Description

The **style** command analyzes the surface characteristics of the writing style of an English−language document. It reports on readability, sentence length and structure, word length and usage, verb type, and sentence openers. Because the **style** command runs the **deroff** command before looking at the text, header files that contain appropriate formatting information should be included as part of the input.

Note: The use of nonstandard formatting macros may cause incorrect sentence breaks.

Flags

−a	Prints all sentences with their length and readability index.
−e	Prints all sentences that begin with an expletive such as "`There are`".
−l*Number*	Prints all sentences longer than the number of words specified by the parameter *Number*.
−ml	Causes the **deroff** command to skip lists; should be used if a document contains many lists of nonsentences.
−mm	Overrides the default **ms** macro package.
−p	Prints all sentences that contain a passive verb.
−P	Prints parts of speech of the words in the document.
−r*Number*	Prints all sentences whose readability index is greater than *Number*.

Related Information

The **diction** command, **deroff** command.

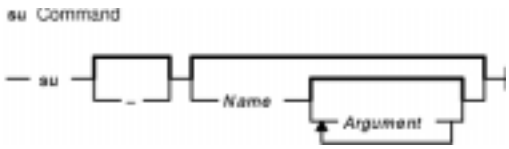
The **ms** macro package.

su Command

Purpose

Changes the user ID associated with a session.

Syntax



su [**−**] [*Name* [*Argument* ...]]

Description

The **su** command changes user credentials to those of the root user or to the user specified by the *Name* parameter, and then initiates a new session. The user name may include a DCE cell specification.

> **Note:** The root user is not required to satisfy the Distributed Computing Environment (DCE) authentication when switching to a DCE user. In this case, the user's DCE credentials are not gained.

Any arguments, such as flags or parameters, that are specified by the *Arguments* parameter must relate to the login shell defined for the user specified by the *Name* parameter. These arguments are passed to the specified user's login shell. For example, if the login shell for user `Fred` is **/usr/bin/csh**, you can include any of the flags for the **csh** command, such as the **−f** flag. When the **su** command runs, it passes the **−f** flag to the **csh** command. When the **csh** command runs, the **−f** flag omits the **.cshrc** startup script.

The following functions are performed by the **su** command:

account checking	Validates the user account to be certain it exists, that it is enabled for the **su** command, that the current user is in a group permitted to switch to this account with the **su** command, and that it can be used from the current controlling terminal.
user authentication	Validates the user's identity, using the system−defined primary authentication methods for the user. If a password has expired, the user must supply a new password.
credentials establishment	Establishes initial user credentials, using the values in the user database. These credentials define the user's access rights and accountability on the system.
session initiation	If the **−** flag is specified, the **su** command initializes the user environment from the values in the user database and the **/etc/environment** file. When the **−** flag is not used, the **su** command does not change the directory.

These functions are performed in the sequence shown. If one function is unsuccessful, the succeeding functions are not done. Refer to the **ckuseracct**, **ckuserID**, **authenticate**, **setpcred**, and **setpenv** subroutines for the semantics of these functions.

To restore the previous session, enter `exit` or press the Ctrl−D key sequence. This action ends the shell called by the **su** command and returns you to the previous shell, user ID, and environment.

If the **su** command is run from the **/usr/bin/tsh** shell, the trusted shell, you exit from that shell. The **su** command does not change the security characteristics of the controlling terminal.

Each time the **su** command is executed an entry is made in the **/var/adm/sulog** file. The **/var/adm/sulog** file records the following information: date, time, system name, and login name. The **/var/adm/sulog** file also records whether or not the login attempt was successful: a + (plus sign) indicates a successful login, and a − (minus sign) indicates an unsuccessful login.

Flags

− Specifies that the process environment is to be set as if the user had logged in to the system using the **login** command. Nothing in the current environment is propagated to the new shell.

Security

Access Control: All users should have execute (x) access to this command. The command should be setuid to the root user to access authentication information, and have the trusted computing base attribute.

Files Accessed:

Mode	File
r	**/etc/passwd**
r	**/etc/group**
r	**/etc/environment**
r	**/etc/security/user**
r	**/etc/security/passwd**
r	**/etc/security/limits**
r	**/etc/security/environ**
w	**/var/adm/sulog**

Auditing Events:

Event	Information
USER_Su	user name

Examples

1. To obtain root user authority, enter:

   ```
   su
   ```

   This command runs a subshell with the effective user ID and privileges of the root user. You will be asked for the root password. Press End Of File, the Ctrl−D key sequence, to end the subshell and return to your original shell session and privileges.

2. To obtain the privileges of the `jim` user, enter:

   ```
   su jim
   ```

This command runs a subshell with the effective user ID and privileges of Jim.

3. To set up the environment as if you had logged in as the `jim` user, enter:

   ```
   su - jim
   ```

   This starts a subshell using Jim's login environment.

4. To run the backup command with root user authority and then return to your original shell, enter:

   ```
   su root "-c /usr/sbin/backup -9 -u"
   ```

This runs the **backup** command with root user authority within root's default shell. You must give the correct root password when queried for the command to execute.

Files

/usr/bin/su	Contains the **su** command.
/etc/environment	Contains user environment values.
/etc/group	Contains the basic group attributes.
/etc/passwd	Contains the basic user attributes.
/etc/security/user	Contains the extended attributes of users.
/etc/security/environ	Contains the environment attributes of users.
/etc/security/limits	Contains the process resource limits of users.
/etc/security/passwd	Contains password information.
/var/adm/sulog	Contains information about login attempts.

Related Information

The **bsh** command, **csh** command, **getty** command, **ksh** command, **login** command, **setgroups** command, **setsenv** command, **tsh** command, and **tsm** command.

The **ckuseracct** subroutine, **ckuserID** subroutine, **setpcred** subroutine, **setpenv** subroutine, **authenticate** subroutine.

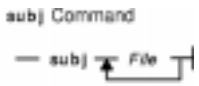
For more information about the identification and authentication of users, discretionary access control, the trusted computing base, and auditing, refer to Security Administration in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

subj Command

Purpose

Generates a list of subjects from a document.

Syntax



subj [*File* ...]

Description

The **subj** command searches one or more English−language files for subjects that might be appropriate in a subject−page index and prints the list of subjects on the standard output. The document should contain formatting commands (from the **nroff**, **troff**, and **mm** commands, among others) to make the best use of the **subj** command.

The **subj** command selects sequences of capitalized words as subjects, except for the first word in each sentence. Thus, if a sentence begins with a proper noun, the capitalization rule does not select this word as a subject. However, since each sentence is expected to begin on a new line, the first word of a sentence that begins in the middle of a line may be erroneously selected. Also, the **subj** command selects modifier−noun sequences from the abstract, headings, and topic sentences (the first sentence in each paragraph). Thus, occasionally a word is incorrectly categorized as a noun or adjective.

The output of the **subj** command may not be appropriate for your needs and should be edited accordingly.

Parameters

File Specifies the English−language files that the **subj** command searches for appropriate subjects for indexing.

Related Information

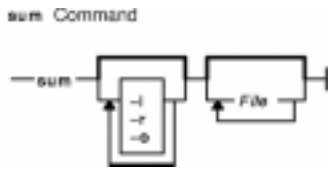
The **mm** command, **ndx** command, **nroff** command, **troff** command.

sum Command

Purpose

Displays the checksum and block count of a file.

Syntax



sum [**−i**] [**−r**] [**−o**] [*File* ...]

Description

The **sum** command reads the file specified by the *File* parameter and calculates a checksum and the number of 1024−byte blocks in that file. If no options are specified, a byte−by−byte algorithm, such as the BSD 4.3 default algorithm, is used. If no files are named, the standard input is read. The checksum and number of 1024−byte blocks are written to standard output. The **sum** command is generally used to determine if a file that has been copied or communicated over transmission lines is an exact copy of the original.

Flags

−i Allows the user to compute the checksum without including header information, if the input file is a binary file. If the input file is not a binary file, the checksum includes header information.

−o Uses the word−by−word algorithm to compute the checksum. The **sum** command with the **−o** flag is compatible with the Version 2 **sum** command in terms of the checksum, but not the number of blocks.

−r Uses a byte−by−byte algorithm to compute the checksum. Using the **−r** flag is the same as using no options.

> **Note:** The default is no longer the word−by−word computation algorithm; it is the BSD 4.3 default algorithm.

Exit Status

This command returns the following exit values:

0 Successful completion.

>0 An error occurred.

Examples

To display the checksum of, and the number of 1024−byte blocks in, the `file1` and `file2` files, enter:

```
sum file1 file2
```

If the checksum of the `file1` file is 32830, the checksum of the `file2` file is 32481, and the `file1` file

contains one block, and the `file2` contains four blocks, the **sum** command displays:

```
32830           1       file1
32481           4       file2
```

Files

/usr/bin/sum Contains the **sum** command.

Related Information

The **cksum** command, **wc** command.

The File Systems Overview for Systems Management in *AIX Version 4.3 System Management Concepts: Operating System and Devices* gives an explanation of what a file system is and why to use one.

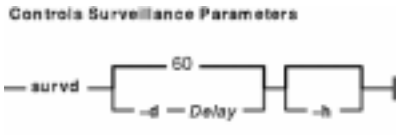
survd Daemon

Purpose

Controls the surveillance daemon.

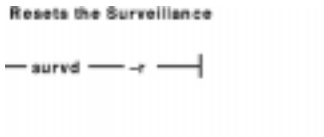
Syntax

Controls Surveillance Parameters



survd [**−d***Delay*] [**−h**]

Resets the Surveillance



survd −r

Description

The **survd** command enables a user with root authority to control the surveillance daemon. You can choose the frequency of the signals that AIX sends to the bring−up microprocessor (BUMP), using the **−d***Delay* flag. You can also choose the way the BUMP will reboot the system (hardware or software reboot) if it does not receive a signal from AIX within the given delay. (The **−h** flag stands for hardware reboot required). Finally, you can decide to turn the surveillance off, using the **−r** flag.

The **survd** daemon works only on multiprocessor systems with Micro Channel I/O. For IBM systems, this includes the IBM 7012 Model G Series, the IBM 7013 Model J Series, and the IBM 7015 Model R Series.

> **Attention:** Do not use the **kill** command to stop the surveillance: in this case the BUMP would no longer receive signals from the daemon and would reboot the system.

Flags

−d*Delay* Specifies the period of the signals that AIX sends to the BUMP. The *Delay* parameter indicates the period in seconds. The minimal value authorized for the *Delay* parameter is 10 seconds. If this flag is not specified, the frequency will be 60 seconds.

−h Indicates that the BUMP will execute a hardware reboot if it does not receive any signal from AIX within the defined delay. If this flag is not specified a software reboot will be executed.

−r Turns off the surveillance and kills the surveillance daemon. This flag cannot be used with **−d** or **−h** flags, and it has no action if the daemon was not running.

Security

Access Control: Only the root user can run this command.

Examples

1. To set the signal period to 70 seconds, enter:

   ```
   survd −d 70
   ```

   Note that if no signal is received by the BUMP within 70 seconds, the BUMP will execute a software reboot, since the **−h** flag is not specified.

2. To turn the surveillance off, enter:

   ```
   survd −r
   ```

Related Information

The *AIX Version 4.3 Problem Solving Guide and Reference*.

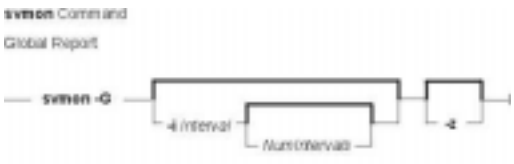
svmon Command

Purpose

Captures and analyzes a snapshot of virtual memory.

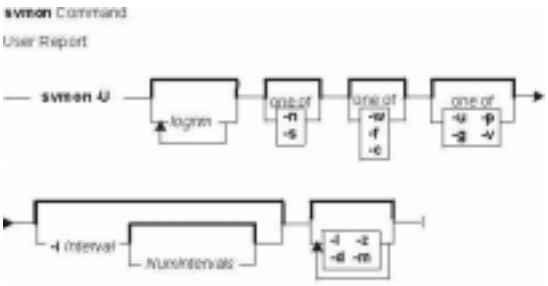
Syntax

Global Report



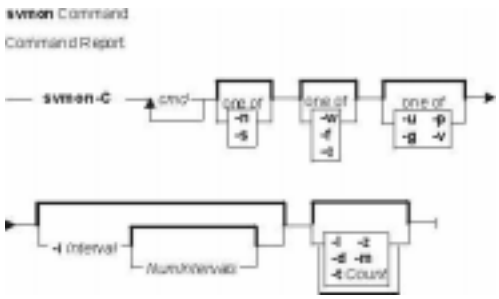
svmon−G [−**i***Interval* [*NumIntervals*]] [−**z**]

User Report



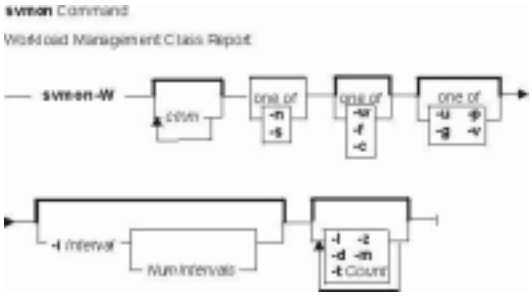
svmon−U [*lognm1...lognmN*] [−**n** | −**s**] [−**w** | −**f** | −**c**] [−**t***Count*] [−**u** | −**p** | −**g** | −**v**] [−**i***Interval* [*NumIntervals*]] [−**l**] [−**d**] [−**z**] [−**m**]

Command Report



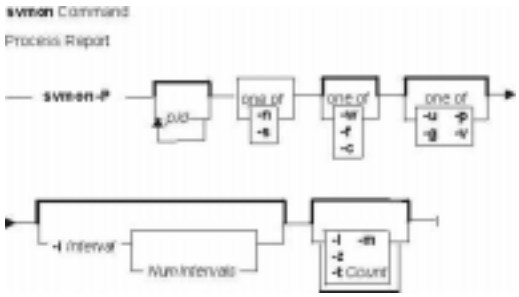
svmon−C*cmd1...cmdN* [−**n** | −**s**] [−**w** | −**f** | −**c**] [−**t***Count*] [−**u** | −**p** | −**g** | −**v**] [−**i***Interval* [*NumIntervals*]] [−**l**] [−**d**] [−**z**] [−**m**]

Workload Management Class Report



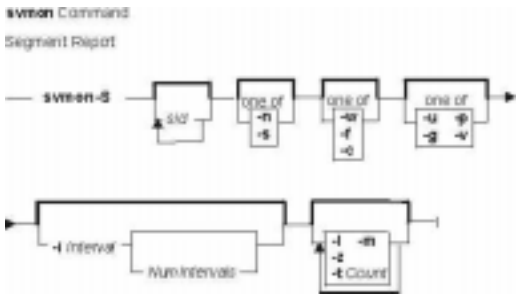
svmon−**W** [*clnm1*...*clnmN*] [−**n** | −**s**] [−**w** | −**f** | −**c**] [−**t***Count*] [−**u** | −**p** | −**g** | −**v**] [−**i***Interval* [*NumIntervals*]] [−**l**] [−**d**] [−**z**] [−**m**]

Process Report



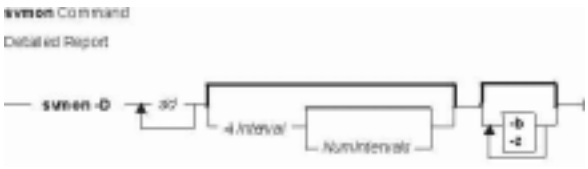
svmon−**P** [*pid1*... *pidN*] [−**n** | −**s**] [−**w** | −**f** | −**c**] [−**t***Count*] [−**u** | −**p** | −**g** | −**v**] [−**i***Interval* [*NumIntervals*]] [−**l**] [−**z**] [−**m**]

Segment Report



svmon−**S** [*sid1*...*sidN*] [−**n** | −**s**] [−**w** | −**f** | −**c**] [−**t***Count*] [−**u** | −**p** | −**g** | −**v**] [−**i***Interval* [*NumIntervals*]] [−**l**] [−**z**] [−**m**]

Detailed Report



svmon−**D***sid1*..*sidN* [−**b**] [−**i***Interval* [*NumIntervals*]] [−**z**]

Description

The **svmon** command displays information about the current state of memory. The displayed information does *not* constitute a true snapshot of memory, because the **svmon** command runs at user level with interrupts enabled.

Reports

Reports

The svmon command creates seven types of reports:

- **global**
- **user**
- **command**
- **workload managment class**
- **process**
- **segment**
- **detailed segment**

Each report type is described here. Unless otherwise noted, all statistics are in units of 4096−byte pages.

Global Report

The global report is printed when the **−G** flag is specified. The column headings in a global report are:

memory
Specifies statistics describing the use of real memory, including:
size
Number of real memory frames (size of real memory)
Note: This includes any free frames that have been made unusable by the memory sizing tool, the rmss command.
inuse
Number of frames containing pages
free
Number of frames free of all memory pools
pin
Number of frames containing pinned pages
virtual
Number of pages allocated in the system virtual space
stolen
Number of frames stolen by rmss and maked unusable by the VMM

in use
Specifies statistics on the subset of real memory in use, including:
work
Number of frames containing working segment pages
pers
Number of frames containing persistent segment pages
clnt
Number of frames containing client segment pages

pin
Specifies statistics on the subset of real memory containing pinned pages, including:
work
Number of frames containing working segment pinned pages
pers
Number of frames containing persistent segment pinned pages
clnt
Number of frames containing client segment pinned pages

pg space
Specifies statistics describing the use of paging space.
size
Size of paging space

inuse
Number of paging space pages used
Note: A frame is a 4K block of real memory

User Report

The user report is printed when the **−U** flag is specified. The column headings in a user login report are:

User
Indicates the user name
Inuse
Indicates the total number of pages in real memory in segments that are used by the user.
Pin
Indicates the total number of pages pinned in segments that are used by the user.
Pgsp
Indicates the total number of pages reserved or used on paging space by segments that are used by the user.
Virtual
Indicates the total number of pages allocated in the process virtual space.

Once this columns heading is displayed, **svmon** displays (if the **−d** flag is specified) information about all the processes run by the specified login user name. It only contains the column heading of the processes as described in **Process Report**.

Then **svmon** displays information about the segments used by those processes. This set of segments is separated into three categories:

1. The segments that are flagged **system** that are basically shared by all processes
2. The segments that are only used by the set of processes
3. The segments that are shared between several users

If **−l**flag is specified, then for each segment in the last category, the list of process identifiers that use the segment is displayed. Beside the process identifier, the login user name that executes it is also displayed. See the **−l** flag description for special segments processing.

Command Report

The command report is printed when the **−C** flag is specified. The column headings in a command report are:

Command
Indicates the command name.
Inuse
Indicates the total number of pages in real memory in segments that are used by the command (all process running the command).
Pin
Indicates the total number of pages pinned in segments that are used by the command (all process running the command).
Pgsp
Indicates the total number of pages reserved or used on paging space by segments that are used by the command.
Virtual
Indicates the total number of pages allocated in the virtual space of the command.

Once this columns heading is displayed, svmon displays (if the **−d** flag is specified) information about all the processes running the specified command. It only contains the column heading of the processes as described in **Process Report**.

Then **svmon** displays information about the segments used by those processes. This set of segments is separated into three categories:

1. The segments that are flagged **system** that are basically shared by all processes.
2. The segments that are only used by the set of processes
3. The segments that are shared between several command names

If the **−l** flag is specified, then for each segment in the last category, the list of process identifiers that use the segment is displayed. Beside the process identifier, the command name it runs is also displayed. See the **−l** flag description for special segments processing.

Workload class Report

The workload class report is printed when the **−W** flag is specified. The column headings in a workload class report are:

Class
Indicates the workload class name.

Inuse
Indicates the total number of pages in real memory in segments belonging to the workload class.

Pin
Indicates the total number of pages pinned in segments belonging to the workload class.

Pgsp
Indicates the total number of pages reserved or used on paging space by segments belonging to the workload class.

Virtual
Indicates the total number of pages allocated in the virtual space of the workload class.

Once this columns heading is displayed,svmon displays information about the segments belonging to the workload class.

If −l option is specified, then for each segment, the list of process identifiers that use the segment is displayed. Beside the process identifier, the workload class the process belongs to is also displayed. See also −l flag description for special segments processing.

Note: a process belongs to the workload class, if its initial thread belongs to it.

Process Report

The process report is printed when the **−P** flag is specified. The column headings in a process report are:

Pid
Indicates the process ID.

Command
Indicates the command the process is running.

Inuse
Indicates the total number of pages in real memory in segments that are used by the process.

Pin
Indicates the total number of pages pinned in segments that are used by the process.

Pgsp
Indicates the total number of pages reserved or used on paging space by segments that are used by the process.

Virtual
Indicates the total number of pages allocated in the process virtual space.

64−bit
Indicates if the process is a 64 bit process (Y) or a 32 bit process (N

Mthrd
Indicates if the process is multi−threaded (Y) or not (N)

Once process information are displayed, **svmon** displays information about all the segments the process uses. Information about segment are described in the paragraph **Segment Report**

Segment Report

The segment report is printed when the **−S** flag is specified. The column headings in a segment report are:

Vsid
Indicates the virtual segment ID. Identifies a uniq segment in the VMM.

Esid
Indicates the effective segment ID. When provided, it indicates how the segment is used by the process. If the vsid segment is mapped by several processes but with different esid values then this field contains '−'. In that case, the exact esid values can be obtained through −P option applied on each process identifiers using the segment.

Type
Identifies the type of the segment: pers indicates a persistent segment, work indicates a working segment, clnt indicates a client segment, map indicates a mapped segment and rmap indicates a real memory mapping segment.

Description
Specifies a textual description of the segment. The value of this column depends on the segment type. If the segment is a persistent segment and is not associated with a log, then the device name and i−node number of the associated file are displayed, separated by a colon. (The device name and i−node can be translated into a file name with the ncheck command.) If the segment is the primary segment of a large file, then the words large file are prepended to the description.

If the segment is a persistent segment and is associated with a log, then the string log is displayed. If the segment is a working segment, then the **svmon** command attempts to determine the role of the segment. For instance, special working segments such as the kernel and shared library are recognized by the **svmon** command. If the segment is the private data segment for a process, then private is printed out. If the segment is the code segment for a process, and the segment report is printed out in response to the **−P** flag, then the word code is prepended to the description.

If the segment is mapped by several processes and used in a different way (for example, a process private segment mapped as shared memory by an other process), then the description is empty. The exact description can be obtained through **−P**flag applied on each process identifier using the segment.

If a segment description is too large to fit in the description space then the description is truncated. The truncated part can be obtained through the **−S** flag (without **−l**) on given segment.

Inuse
Indicates the number of pages in real memory in this segment.

Pin
Indicates the number of pages pinned in this segment.

Pgsp
Indicates the number of pages used on paging space by this segment. This field is relevant only for working segments.

Virtual
Indicates the number of pages allocated for the virtual space of the segment. (Only for working segments).

Note: VMM manages this value for statistics purpose. It may happened it is not updated. Then its value may be less than the inuse counters.

Address Range

Specifies the range(s) within the segment pages have been allocated. Working segment may have two ranges because pages are allocated by starting from both ends and moving towards the middle.

If the **−l** flag is present, the list of process identifiers that use that segment is displayed. See the **−l** flag description for special segments processing.

Detailed Report

The detailed report is printed when the **−D** flag is specified. The column headings in a detailed report are:

Same segment information as described in the segment report and it prints for each frame:

Page
Relative page number to the virtual space. This page number can be higher than the number of frame in a segment (65532) in the virtual space is larger than a single segment (large file).

Frame
Frame number in the real memory

Pin
Indicates if the frame is pinned or not

Ref
Indicates if the frame has been referenced by a process (−b option only).

Mod
Indicates if the frame has been modified by a process (−b option only).

Statistic values

A segment can be used by multiple processes. Each page from such a segment is accounted for in the inuse, pin, virtual or pgspace fields for each process that uses the segment. The total of the inuse, pin, virtual and pgspace fields over all active processes may exceed the total number of pages in memory or on paging space.

VMM manages virtual counter for statistics purpose. It may happen if it is not updated. Then its value may be less than the inuse counters.

Flags

If no command line flag is given, then the **−G flag** is implicit.

−G
Displays a global report.

−P *[pid1... pidN]*
Displays memory usage statistics for process *pid1...pidN*. *pid* is a decimal value. If no list of process IDs (PIDs) is supplied memory usage statistics are displayed for all active processes.

−S *[sid1...sidN]*
Displays memory−usage statistics for segments *sid1...sidN*. *sid* is a hexadecimal value. If no list of segment IDs (SIDs) is supplied memory usage statistics are displayed for all defined segments.

−U *[lognm1...lognmN]*
Displays memory usage statistics for the login name *lognm1...lognmN*. *lognm* is a string, it is an exact login name. If no list of login identifier is supplied, memory usage statistics are displayed for all defined login identifiers.

−C*cmd1...cmdN*
Displays memory usage statistics for the processes running the command name *cmdnm1...cmdnmN*. *cmdnm* is a string. It is the exact basename of an executable file.

−W *[clnm1...clnmN]*
Displays memory usage statistics for the workload management class *clnm1...clnmN*. *clnm* is a string. It is the exact name of a class. If no list of class name is supplied, memory usage statistics are displayed for all defined class names.

−D*sid1...sidN*
Displays memory−usage statistics for segments *sid1...sidN*, and a detail status of all frames of each segment.

−n
Indicates that only non−system segments are to be included in the statistics. By default all segments are analyzed.

−s
Indicates that only system segments are to be included in the statistics. By default all segments are analyzed.

−w
Indicates that only working segments are to be included in the statistics. By default all segments are analyzed.

−f
Indicates that only persistent segments (files) are to be included in the statistics. By default all segments are analyzed.

−c
Indicates that only client segments are to be included in the statistics. By default all segments are analyzed.

−u
Indicates that the objects to be printed are sorted in decreasing order by the total number of pages in real memory. It is the default sorting criteria if none of the following flags are present: **−p**, **−g** and **−v**.

−p
Indicates that the object to be printed are sorted in decreasing order by the total number of pages pinned.

−g
Indicates that the object to be printed are sorted in decreasing order by the total number of pages reserved or used on paging space. This flag in conjunction with the segment report shifts the non−working segment at the end of the sorted list.

−v
Indicates that the object to be printed are sorted in decreasing order by the total number of pages in virtual space. This flag in conjunction with the segment report shifts the non−working segment at the end of the sorted list.

−b
Shows the status of the reference and modified bits of all the displayed frames (detailed report **−D**). Once shown the reference bit of the frame is reset. When used with the **−i** flag it detects which frames are accessed between each interval.

Note: This flag should be used with caution because of its performance impacts.

−l
Shows, for each displayed segment, the list of process identifiers that use the segment and, according to the type of report, the entity name (login, command or class) the process belong to. For special segments a label is displayed instead of the list of process identifiers.

System segment
This label is displayed for segments that are flagged system

Unused segment
This label is displayed for segments that are not used by any existing processes.

Shared library text
This label is displayed for segments that contain text of shared library, and that may be used by most of the processes (libc.a). This is to prevent the display of a long list of process.

−z
Displays the maximum memory size dynamically allocated (malloc) by **svmon** during its execution.

−m
Displays information about source segment rather than mapping segment when a segment is mapping a source segment.

−d
Displays for a given entity, the memory statistics of the processes belonging to the entity.

−t*Count*

Displays memory usage statistics for the top *Count* object to be printed

−i*Interval [NumIntervals]*

Instructs the **svmon** command to print statistics out repetitively. Statistics are collected and printed every *Interval* seconds. *NumIntervals* is the number of repetitions; if not specified, **svmon** runs until user interruption, Ctrl−C.

Notes:

- Because it may take a few seconds to collect statistics for some options, the observed interval may be larger than the specified interval.
- If none of the **−u**, **−p**, **−g**, and **−v** flags are specified, **−u** is implicit.

Security

Access Control: You must have root authority to run this command.

Examples

1. To print out global statistics, enter:
   ```
   svmon −G
   ```

2. To print out global statistics each minute during 1/2 hour, enter:
   ```
   svmon −G −i 60 30
   ```

3. To print out the memory usage statistics for the users root and steve taking into account working segments, enter:
   ```
   svmon −U root steve −w
   ```

4. To print out the top 10 users of the paging space blocks, enter:
   ```
   svmon −U −g −t 10
   ```

5. To print out memory usage statistics for the commands oracle, xemacs and cc, enter:
   ```
   svmon −C oracle xemacs cc
   ```

6. To print out the memory usage statistics for the workload class default and developer, enter:
   ```
   svmon −W default developer
   ```

7. To print out the memory usage statistics for processes 6746 and 10078 taking into account working and persistent segments, enter:
   ```
   svmon −P 6746 10078 −wf
   ```

8. To print out the memory usage statistics of the top 10 process according to the number of pinned pages, enter
   ```
   svmon −P −t 10 −p
   ```

9. To print out the memory usage statistics of all the process taking into account non−system working segments and sorting the process by the number of virtual pages, enter:
   ```
   svmon −P −n −w −v
   ```

10. To print out the memory usage statistic of segments e00e and 15015, result sorted by the number of reserved paging space blocks, enter:
    ```
    svmon −S e00e 15015 −g
    ```

11. To print out the memory usage statistics of the top 5 working segment according to the number of virtual pages , enter:
    ```
    svmon −S −t 5 −w −v
    ```

12. To sort system segments by the number of pages in real memory, and print out the top 10 system segments of the resulting list, enter:
    ```
    svmon −S −s −u −t 10
    ```

13. To print out the frames belonging to the segment e00e, enter:
    ```
    svmon -D e00e
    ```

14. To print out the frames belonging to the segment e00e with the status bit of each frame, enter:
    ```
    svmon -D e00e -b
    ```

Related Information

The **ncheck** command, **rmss** command.

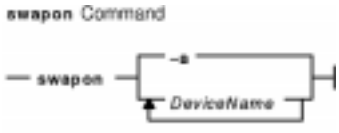
Logical Volume Storage Overview, Paging Space Overview in *AIX Version 4 System Management Guide: Operating System and Devices*.

swapon Command

Purpose

Specifies additional devices for paging and swapping.

Syntax



swapon −a | *DeviceName* ...

Description

The **swapon** command is used to specify additional devices on which paging and swapping are to take place. It is used during early system initialization to make the initial paging space available. During a later phase of system initialization, the **swapon −a** command is used to make other devices available so that paging and swapping activity is interleaved across several devices. The **swapon −a** command makes all devices specified in the **/etc/swapspaces** file available. Calls to the **swapon** command normally occur in the system multiuser initialization **/etc/rc** file.

The *DeviceName* parameter specifies a specific device to be made available. The second form gives individual block devices as given in the system swap configuration table. The call makes this space and other defined spaces available to the system for paging and swap allocation. The system swap configuration table is the set of all devices specified in the **/etc/swapspaces** file.

You can use the Web−based System Manager Devices application (**wsm devices** fast path) to run this command. You could also use the System Management Interface Tool (SMIT) **smit swapon** fast path to run this command.

Note: The maximum number of active paging spaces is 16.

Flags

−a Causes all devices present in the **/etc/swapspaces** file to be made available.

Examples

1. To cause all devices present in the **/etc/swapspaces** file to be made available, enter:

   ```
   swapon -a
   ```

   All devices present in the **/etc/swapspaces** file are now available.

2. To cause the **/dev/paging03** and **/dev/paging04** devices to be available for paging and swapping, enter:

   ```
   swapon /dev/paging03 /dev/paging04
   ```

The **/dev/paging03** and **/dev/paging04** devices are now available.

Files

/etc/rc	System multiuser initialization
/dev/paging	Device entries for paging/swap space
/etc/swapspaces	Contains a list of swap devices.

Related Information

The **mkps** command, **chps** command.

The **swapon** subroutine.

The Paging Space Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains paging space and its allocation policies.

Setting up and running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

The System Management Interface Tool (SMIT) Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains the structure, main menus, and tasks that are done with SMIT.

swcons Command

Purpose

Redirects, temporarily, the system console output to a specified device or file.

Syntax

swcons [**−p***log_file*] [**−s***log_size*] [**−t***tag_verbosity*] [**−v***log_verbosity*] *PathName*

Description

The **swcons** command temporarily switches the system console output to a different target during system operation. This command only switches system informational−, error−, and intervention−required message output to the specified destination. The **swcons** command does not affect the operation of the system console device that is providing a login by way of the **getty** command.

The device or file specified when using this command remains the target for console output until changed by another **swcons** command, until the next start of the system, or until the console driver detects an error when accessing the designated device or file. If an open or write error is detected on the device or file specified by the **swcons** command, the console device driver switches all output back to the device or file that provided console support when the system was last started.

The *PathName* parameter must be a fully qualified path name to a device or file that is to receive system console message output. If the *PathName* parameter specifies a file that does not exist, the **swcons** command creates the file. If the file does exist, the **swcons** command appends any new console message output to the contents of the file.

> **Attention:** Use of the **swcons** command to switch console output to an NFS mounted file system may cause the operating system to hang.

Flags

−p*log_file*	Specifies the full path name to use for the console output log file.
−s*log_size*	Specifies the size, in bytes, of the console output log file.
−t*tag_verbosity*	Specifies the verbosity level for console output tagging. Zero disables tagging; 1 through 9 enable tagging. For additional information about console output logging and tagging, see the console Special File in the *AIX Version 4.3 Files Reference* book.
−v*log_verbosity*	Specifies the verbosity level for console output logging. Zero disables logging; 1 through 9 enable logging.

Examples

1. To change the system console message output to a file called `console.out` in the `/tmp` directory, enter:

   ```
   swcons /tmp/console.out
   ```

2. To change the system console message output to a terminal with the logical name `tty3`, enter:

```
swcons /dev/tty3
```

3. To change the system−console message output back to the device or file that supported the console output at system start time, enter:

   ```
   swcons
   ```

Files

/dev/console Specifies the special file for system console access.
/usr/sbin/swcons Contains the **swcons** command file.

Related Information

The **chcons** command, **lscons** command.

The **console** special file.

sync Command

Purpose

Updates the i−node table and writes buffered files to the hard disk.

Syntax

sync

Description

The **sync** command runs the **sync** subroutine. If the system must be stopped, run the **sync** command to ensure file system integrity. The **sync** command writes all unwritten system buffers to disk including modified i−nodes, delayed block I/O, and read−write mapped files.

> **Note:** The writing, although scheduled, is not necessarily complete upon return from the **sync** subroutine.

Related Information

The **sync** subroutine.

synclvodm Command

Purpose

Synchronizes or rebuilds the logical volume control block, the device configuration database, and the volume group descriptor areas on the physical volumes.

Syntax

synclvodm [**−v**] *VolumeGroup* [*LogicalVolume ...*]

Description

During normal operations, the device configuration database remains consistent with the logical volume manager information in the logical volume control blocks and the volume group descriptor areas on the physical volumes. If for some reason the device configuration database is not consistent with Logical Volume Manager information, the **synclvodm** command can be used to resynchronize the database. The volume group must be active for the resynchronization to occur (see **varyonvg**). If logical volume names are specified, only the information related to those logical volumes is updated.

> **Attention:** Do not remove the **/dev** entries for volume groups or logical volumes. Do not change the device configuration database entries for volume groups or logical volumes using the object data manager.
>
> **Note:** To use this command, you must either have root user authority or be a member of the **system** group.

Flags

−v verbose

Examples

To synchronize the device configuration database with the logical volume manager information for rootvg, enter the following:

```
synclvodm rootvg
```

Files

/usr/sbin/synclvodm Contains the **synclvodm** command.

Related Information

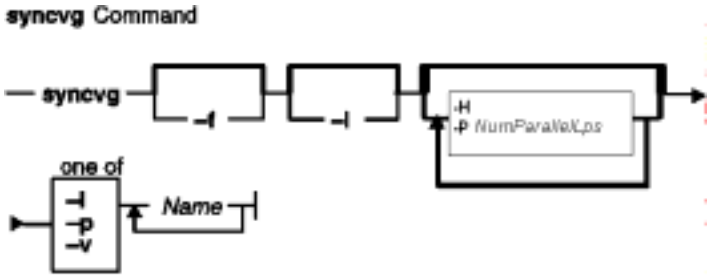
The **varyonvg** command, **varyoffvg** command.

syncvg Command

Purpose

Synchronizes logical volume copies that are not current.

Syntax



syncvg [**−f**] [**−i**] [**−H**] [**−P***NumParallelLps*] { **−l** | **−p** | **−v** } *Name* ...

Description

The **syncvg** command synchronizes the physical partitions, which are copies of the original physical partition, that are not current. The **syncvg** command can be used with logical volumes, physical volumes, or volume groups, with the *Name* parameter representing the logical volume name, physical volume name, or volume group name. The synchronization process can be time consuming, depending on the hardware characteristics and the amount of data.

When the **−f** flag is used, a good physical copy is chosen and propagated to all other copies of the logical partition, whether or not they are stale. Using this flag is necessary in cases where the logical volume does not have the mirror write consistency recovery.

Unless disabled, the copies within a volume group are synchronized automatically when the volume group is activated by the **varyonvg** command.

> **Note:** For the **sycnvg** command to be successful, at least one good copy of the logical volume should be accessible, and the physical volumes that contains this copy should be in ACTIVE state. If force option is used, the above condition applies to all mirror copies.

Flags

−f	Specifies a good physical copy is chosen and propagated to all other copies of the logical partition, whether or not they are stale.
−H	Postpones writes for this volume group on other active concurrent cluster nodes until this sync operation is complete. When using the −H flag, the −P flag does not require that all the nodes on the cluster support the −P flag. This flag is ignored if the volume group is not varyed on in concurrent mode.
−i	Reads the names from standard input.
−l	Specifies that the *Name* parameter represents a logical volume device name.
−p	Specifies that the *Name* parameter represents a physical volume device name.
−P*NumParallelLps*	Numbers of logical partitions to by syncronized in parallel. The valid range for

NumParallelLps is 1 to 32. *NumParallelLps* must be tailered to the machine, disks in the volume group, system resources, and volume group mode.

When a volume group is varried on in concurrent mode all other cluster nodes that have this volume group varried must be at least AIX 4.3.0 otherwise syncvg will ignore this option and continue.

−v Specifies that the *Name* parameter represents a volume group device name.

Examples

1. To synchronize the copies on physical volumes `hdisk04` and `hdisk05`, enter:

   ```
   syncvg −p hdisk04 hdisk05
   ```

2. To synchronize the copies on volume groups `vg04` and `vg05`, enter:

   ```
   syncvg −v vg04 vg05
   ```

Files

/usr/sbin/syncvg	Contains the **syncvg** command.
/tmp	Directory where the temporary files are stored and while the command is running.

Related Information

The **varyonvg** command.

The Logical Volume Storage Overview in *AIX Version 4.3 System Management Guide: Operating System and Devices* explains the Logical Volume Manager, physical volumes, logical volumes, volume groups, organization, ensuring data integrity, and allocation characteristics.

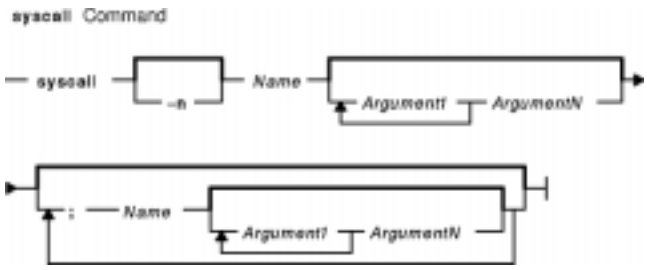
The System Management Interface Tool (SMIT) Overview in *AIX Version 4.3 System Management Guide: Operating System and Devices* explains the structure, main menus, and tasks that are done with SMIT.

syscall Command

Purpose

Performs a specified subroutine call.

Syntax



syscall [**−n**] *Name* [*Argument1 ... ArgumentN*] [**;** *Name* [*Argument1 ... ArgumentN*]] ...

Description

The **syscall** command executes a system call interface program, which performs the subroutine call specified by the *Name* parameter. If you specify the **−n** flag, the **syscall** command performs the call **n** times. Arguments specified by the *Argument* parameter are passed to the subroutine without error checking. The *Argument* parameter can be expressed in the following formats:

0x *nnn*	Hexadecimal constant *nnn*.
0 *nnn*	Octal constant *nnn*.
nnn	Decimal constant *nnn*.
+*nnn*	Decimal constant *nnn*.
−*nnn*	Decimal constant *nnn*.
"*string*	The character string "*string*".
'*string*	The character string "*string*".
string	The character string "*string*".
#*string*	The length of the character string "*string*".
&&*n*	The address of the *n*th argument to this subroutine. (*n*=0 is the subroutine name.)
&*n*	The address of the *n*th byte in an internal 10KB buffer.
$*n*	The result of the *n*th subroutine. (*n*=0 is the first subroutine.)
string	Anything else is a literal character string.

The **syscall** command prints a message and exits for unknown subroutines and for subroutines that return a value of −1.

> **Note:** The **syscall** command understands the **sleep** subroutine as a special case subroutine.

Flags

−n Specifies the number of times the **syscall** command performs the specified subroutine.

; Separates multiple subroutines (up to a maximum of 20) issued by the same invocation of the

syscall command.

Security

Access Control: You must have root authority to run this command.

Examples

To simulate the C program fragment:

```
output=open("x", 401, 0755);

write(output, "hello", strlen("hello"));
```

enter:

```
syscall open x 401 0755 \; write \$0 hello \#hello
```

Note: Special shell characters must be escaped.

Files

/usr/bin/syscall Contains the **syscall** command.

Related Information

The **bsh** command, **Rsh** command, **csh** command, **ksh** command, **sh** command.

The **open** subroutine, **sleep** subroutine.

Shells Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

syscalls Command

Purpose

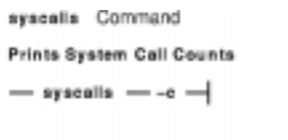
Provides system call tracing and counting for specific processes and the system.

Syntax

To Create or Destroy Buffer:

syscalls [[**−enable** *bytes*]| **−disable**]

To Print System Call Counts:



syscalls−c

To Print System Call Events or Start Tracing:

syscalls [**−o** *filename*] [**−t**] { [[**−p***pid*] **−start** | **−stop**] | **−x** *program* }

Description

The **syscalls** (system call tracing) command, captures system call entry and exit events by individual processes or all processes on the system. The **syscalls** command can also maintain counts for all system calls made over long periods of time.

Notes:

1. System call events are logged in a shared−memory trace buffer. The same shared memory identifier may be used by other processes resulting in a collision. In such circumstances, the **−enable** flag needs to be issued.
2. The **syscalls** command does not use the **trace** daemon.
3. The system crashes if **ipcrm −M sharedmemid** is run after syscalls has been run. Run **stem −shmkill** instead of running **ipcrm −M** to remove the shared memory segment.

Flags

−c	Prints a summary of system call counts for all processes. The counters are not reset.
−disable	Destroys the system call buffer and disables system call tracing and counting.
−enable*bytes*	Creates the system call trace buffer. If this flag is not used, the **syscalls** command creates a buffer of the default size of 819,200 bytes. Use this flag if events are not being logged in the buffer. This is the result of a collision with another process using the same shared memory buffer ID.
−o *filename*	Prints output to *filename* rather than standard out.

−p *pid* When used with the **−start** flag, only events for processes with this *pid* will be logged in the **syscalls** buffer. When used with the **−stop** option, **syscalls** filters the data in the buffer and only prints output for this *pid.*

−start Resets the trace buffer pointer. This option enables the buffer if it does not exist and resets the counters to zero.

−stop Stops the logging of system call events and prints the contents of the buffer.

−t Prints the time associated with each system call event alongside the event.

−x *program* Runs *program* while logging events for only that process. The buffer is enabled if needed.

Security

Access Control: You must be root or a member of the perf group to run this command.

Examples

1. To collect system calls for a particular program, enter:
   ```
   syscalls −x /bin/ps
   ```

   Output similar to the following appears:

   ```
     PID    TTY  TIME CMD
   19841  pts/4  0:01 /bin/ksh
   23715  pts/4  0:00 syscalls −x /bin/ps
   30720  pts/4  0:00 /bin/ps
   34972  pts/4  0:01 ksh
     PID   System Call
   30720           .kfork  Exit , return=0  Call preceded tracing.
   30720          .getpid  () = 30720
   30720       .sigaction  (2, 2ff7eba8, 2ff7ebbc) = 0
   30720       .sigaction  (3, 2ff7eba8, 2ff7ebcc) = 0
   30720     .sigprocmask  (0, 2ff7ebac, 2ff7ebdc) = 0
   30720       .sigaction  (20, 2ff7eba8, 2ff7ebe8) = 0
   30720           .kfork  () = 31233
   30720        .kwaitpid  (2ff7ebfc, 31233, 0, 0) = 31233
   30720       .sigaction  (2, 2ff7ebbc, 0) = 0
   30720       .sigaction  (3, 2ff7ebcc, 0) = 0
   30720       .sigaction  (20, 2ff7ebe8, 0) = 0
   30720     .sigprocmask  (2, 2ff7ebdc, 0) = 0
   30720         .getuidx  (4) = 0
   30720         .getuidx  (2) = 0
   30720         .getuidx  (1) = 0
   30720         .getgidx  (4) = 0
   30720         .getgidx  (2) = 0
   30720         .getgidx  (1) = 0
   30720           ._load  NoFormat, (0x2ff7ef54, 0x0, 0x0, 0x2ff7ff58) = 537227760
   30720            .sbrk  (65536) = 537235456
   30720          .getpid  () = 30720
   ```

2. To produce a count of system calls made by all processes, enter:
   ```
   syscalls −start
   ```

   followed by entering:

   ```
   syscalls −c
   ```

   Output similar to the following appears:

   ```
   System Call Counts for all processes
         5041      .lseek
   ```

```
4950      .kreadv
 744      .sigaction
 366      .close
 338      .sbrk
 190      .kioctl
 120      .getuidx
 116      .kwritev
 108      .kfcntl
 105      .getgidx
  95      .kwaitpid
  92      .gettimer
  92      .select
  70      .getpid
  70      .sigprocmask
  52      .execve
  51      ._exit
  51      .kfork
  35      .open
  35      ._load
  33      .pipe
  33      .incinterval
  28      .sigreturn
  27      .access
  16      .brk
  15      .times
  15      .privcheck
  15      .gettimerid
  10      .statx
   9      .STEM_R10string
   4      .sysconfig
   3      .P2counters_accum
   3      .shmget
   3      .shmat
   2      .setpgid
   2      .shmctl
   2      .kioctl
   1      .Patch_Demux_Addr_2
   1      .Patch_Demux_Addr_High
   1      .STEM_R3R4string
   1      .shmdt
   1      .Stem_KEX_copy_demux_entry
   1      .STEM_R3R4string
   1      .Patch_Demux_Addr_1
   1      .pause
   1      .accessx
```

Files

/usr/bin/syscalls Contains the **syscalls** command.

Related Information

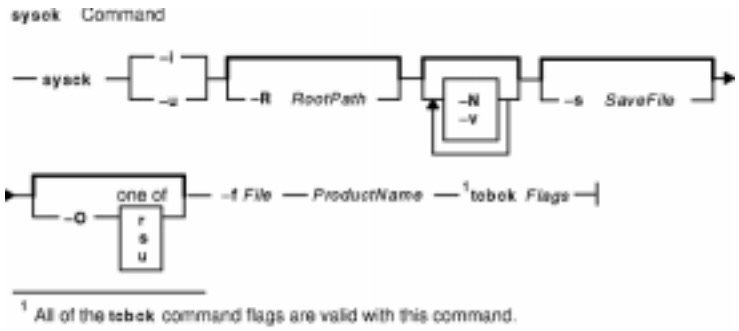
The **stem** command.

sysck Command

Purpose

Checks the inventory information during installation and update procedures.

Syntax



sysck { **−i** | **−u** } [**−R***RootPath*] [**−N**] [**−v**] [**−s***SaveFile*] [**−O** { **r** | **s** | **u** }] **−f***FileProductName* { **tcbck** *Flags* }

1All of the **tcbck** command flags are valid with this command.

Description

> **Attention:** The **sysck** command DOES NOT support checking files that are greater than 2 gigabytes. If a product needs to ship a file that is greater than 2 gigabytes, set size and checksum values in their *Fileset*.inventory to **VOLATILE**, so the **sysck** command will not try to acces the file.
>
> **Note:** All of the **tcbck** command flags are valid with the **sysck** command. This feature provides compatibility with AIX Version 3.1. For more information on the **tcbck** command and a complete listing of its flags, refer to *AIX Version 4.3 Commands Reference*.

The **sysck** command checks file definitions against the extracted files from the installation and update media and updates the Software Vital Product Data (SWVPD) database. The **sysck** command does not recognize the following special characters in file names: `, ', \, ", ^, ( ), |, {}, [], <>,and : . If a file name contains one of these characters, the **sysck** command fails.

The **sysck** command is primarily used during the installation and update of software products.

When invoked with the **−i** flag, the **sysck** command checks the attributes of an extracted file with its file definitions, updates the SWVPD, and attempts to fix some errors if they exist.

The *File* parameter is the name of the stanza file that contains the file definitions. An example of such a file is the **/etc/security/sysck.cfg** file, although the **syschk** command does not use this file. The **sysck** command checks the size, links, symlinks, owner, group, and mode attributes of a file for which the type attribute is set to **FILE**. When invoked with the **−v** flag as well as the **−i** flag, **sysck** also checks the checksum value of a file.

The **sysck** command updates the file name, product name, type, checksum, and size of each file in the SWVPD database.

To fix errors, the **sysck** command resets the attribute of the installed or updated file to the defined value in the *File* stanza file, except for some attributes as described in "Fixing Errors" .

When invoked with the **−u** flag, the **sysck** command removes the entry from the SWVPD database for each file that is part of the software product *ProductName*. The **sysck** command also deletes any hard links and symbolic links for each file, as defined in the SWVPD database.

Flags

−f*File* Specifies the name of the stanza file that contains the file definitions.

−i Checks for the correct installation of a software product's files. Updates the SWVPD database with the file definitions, and attempts to fix some errors if found.

−N Specifies that the SWVPD database should not be updated.

−O {**r**|**s**|**u**} Specifies which part of the SWVPD is to be updated, as follows:

- **r** Specifies the root part of the SWVPD.
- **s** Specifies the **/usr/share** part of the SWVPD.
- **u** Specifies the **/usr** part of the SWVPD (default).

−R*RootPath* Use *RootPath* as root instead of "/".

−s*SaveFile* Takes a snapshot of what is currently in the VPD and saves it in stanza format to the file specified by *SaveFile*. Called with the **−u** option. No action is taken in the database with this flag. Must be used with the **−f** option. For example:

```
sysck −i −s /tmp/save.inv −f /tmp/real.inv bos.rte.shell
```

−u Deletes file entries from the SWVPD and deletes hard links and symbolic links.

−v Verifies that the checksum is correct.

ProductName Specifies the installable software product or option that is being checked.

Environment Variables

INUTREE The environment variable **INUTREE** has only the following four valid values:

- **NULL** Same as **INUTREE** not being set.
- **M** Specifies the root part of the SWVPD.
- **S** Specifies the **/usr/share** part of the SWVPD.
- **U** Specifies the **/usr** part of the SWVPD (default).

INUTREE can be used instead of the **−O***Tree* flag.

INUNOVPD The environment variable **INUNOVPD** can be null or can be set to 1. If it is set to 1 then **sysck** does not update the SWVPD. **INUNOVPD** can be used instead of the **−N** flag.

INUVERIFY If the environment variable **INUVERIFY** is set to 1 **sysck** verifies that the checksum attributes in the stanza file are correct. **INUVERIFY** can be used instead of the **−v** flag.

File Definitions

acl The access control list for the file. If the value is blank, the **acl** attribute is removed. If no value is specified, the command computes a value, according to the format described in Access Control Lists.

This attribute should grant x (execute) access only to the root user and members of the security group. The command should **setuid** to the root user and have the trusted computing base attribute.

class The logical group of the file. A value must be specified because it cannot be computed. The value is *ClassName* [*ClassName*].

checksum The checksum of the file. If the value is blank, the **checksum** attribute is removed. If no value is specified, the command computes a value, according to the format given in the **sum** command. The value is the output of the **sum −r** command, including spaces.

group The file group. If the value is blank, the **group** attribute is removed. If no value is specified, the command computes a value, which can be a group ID or a group name.

mode The file mode. If the value is blank, the **mode** attribute is removed. If no value is specified, the command computes a value, which can be an octal number or a string (**rwx**), and have the **TCB**, **SUID**, **SGID**, and **SVTX** attributes.

owner The file owner. If the value is blank, the **owner** attribute is removed. If no value is specified, the command computes a value, which can be a user ID or a user name.

size The size of the file in bytes. If the value is blank, the **size** attribute is removed. A **VOLATILE** value in the size field indicates that the file size will change (so no checksum value can be given). A **NOSIZE** value indicates that the file has 0 length. If no value is specified, the command computes a value, which is a decimal number.

target Allows symbolic links and hard links to exist as separate stanzas in the inventory. The **target** file definition refers to the full path name of the source of the link, for example:

```
/etc/foo −−> /usr/bar
```

The **target** is `/usr/bar`.

type The type of file. This value cannot be blank. If no value is specified, the command computes a value, which can be the **FILE**, **DIRECTORY**, **FIFO**, **BLK_DEV**, **CHAR_DEV**, **LINK**, **MPX_DEV**, and **SYMLINK** keywords.

xacl An addition to the extended−access control list. A value must be specified as a single entry in an extended−access control list because the value cannot be computed. This attribute is valid only if the **−i** flag is used. For information about the format, see the **acl** file definition above.

Fixing Errors

To fix errors, the **sysck** command resets the attribute of the installed or updated file to the defined value defined in the *File* stanza file except for the following attributes, for which the **sysck** command acts as described:

links Creates any missing hard links. If a link exists to another file that is not listed in this definition, the link is deleted.

program If this attribute is included in the *File* stanza file, **sysck** invokes the program. A message is printed if an error occurs, but no additional action is taken.

symlinks Creates any missing symbolic links. If a link exists to another file that is not listed in this definition, the link is deleted.

Security

Privilege Control: Only the root user can run this command.

Examples

1. A product that uses the **installp** command to install ships an inventory file in its image. To add the definitions to the inventory database and check permissions, links, checksums, etc., enter:

   ```
   sysck −i −f dude.rte.inventory dude.rte
   ```

   where `dude.rte.inventory` would look like the following:

```
/usr/bin/dude.exec:
   class = apply,inventory,dude.rte
   owner = bin
   group = bin
   mode = 555
   type = FILE
   size = 2744
   checksum = "04720        3"
```

2. To remove any links to files for a product that has been removed from the system and remove the files from the inventory database, enter:

   ```
   sysck -u -f dude.rte.inventory dude.rte
   ```

Files

/etc/objrepos/inventory	Specifies names and locations of files in a software product on the root.
/usr/lib/objrepos/inventory	Specifies names and locations of files in a software product on the **/usr** file system.
/usr/share/lib/objrepos/inventory	Specifies names and locations of files in a software product on the **/usr/share** file system.

Related Information

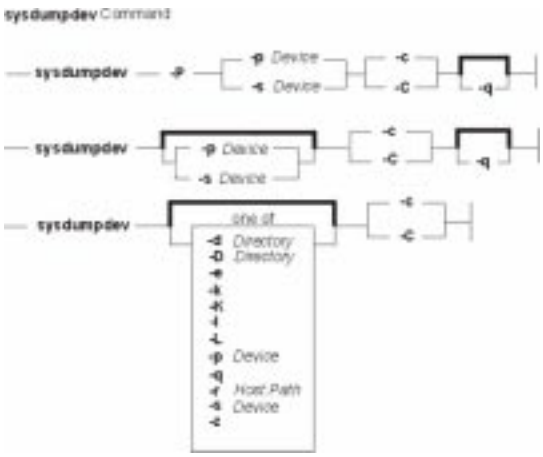
The **installp** command, **sum** command, **tcbck** command.

sysdumpdev Command

Purpose

Changes the primary or secondary dump device designation in a running system.

Syntax



sysdumpdev [**−c** | **−C**] **−P** { **−p** *Device* | **−s** *Device* } [**−q**]

sysdumpdev [**−c** | **−C**] [**−p** *Device* | **−s** *Device*] [**−q**]

sysdumpdev [**−c** | **−C**] [**−d***Directory* | **−D***Directory* | **−e** | [**−k** | **−K**] | **−l**|**−L** | **−p** *Device* | **−q** |**−r***Host***:***Path* | **−s** *Device* | **−z**]

Description

The **sysdumpdev** command changes the primary or secondary dump device designation in a system that is running. The primary and secondary dump devices are designated in a system configuration object. The new device designations are in effect until the **sysdumpdev** command is run again, or the system is restarted.

If no flags are used with the **sysdumpdev** command, the dump devices defined in the **SWservAt** ODM object class are used. The default primary dump device is **/dev/hd6**. The default secondary dump device is **/dev/sysdumpnull**.

Notes:

1. A mirrored paging space may be used as a dump device.
2. Do not use a diskette drive as your dump device.
3. If you use a paging device, only use hd6, the primary paging device. AIX Version 4.2.1 or later supports using any paging device in the root volume group (rootvg) as the secondary dump device.

You can use the Web−based System Manager Devices application (**wsm devices** fast path) to run this command. You could also use the System Management Interface Tool (SMIT) **smit sysdumpdev** fast path to run this command.

You can also use the **sysdumpdev** command to specify whether or not dumps should be compressed before

writing them to the dump device. Compressing dumps reduces the size needed for dump devices, but may cause the dump process to take longer.

Notes:

1. The **savecore** command should be used to copy a compressed dump from the dump device to a file.
2. The dump compression feature only applies to AIX Version 4.3.2 and later versions.

Running sysdumpdev in Non−rootvg Volume Groups

You can use a dump logical volume outside the root volume group, if it is not a permanent dump device. For example, if the **−P** flag is not specified. However, if you choose a paging space, we cannot copy the dump device unless it is in rootvg. During the time we must copy the dump device, only rootvg is active before paging is started.

The primary dump devices must always be in the root volume group for permanent dump devices. The secondary device may be outside the root volume group unless it is a paging space.

Configuring Remote Dump Devices with sysdumpdev

The **sysdumpdev** command can also be used to configure remote dump devices. The following conditions must be met before a remote dump device can be configured:

- The local and the remote host must have Transmission Control Protocol/Internet Protocol (TCP/IP) installed and configured.
- The local host must have Network File System (NFS) installed.
- The remote host must support NFS.
- The remote host must be operational and on the network. This condition can be tested by issuing the **ping** command.
- The remote host must have an NFS exported directory defined such that the local host has read and write permissions as well as root access to the dump file on the remote host.
- The remote host cannot be the same as the local host.

The network device driver must support remote dump. Drivers which support remote dump include the drivers for these network devices:

- Integrated Ethernet MCA Adapter
- IBM 10/100 Mbps Ethernet TX MCA Adapter (8f62)
- IBM PCI Ethernet Adapter (22100020)
- IBM 10/100 Mbps Ethernet PCI Adapter (23100020)
- FDDI MCA Adapter
- ISA Token Ring Adapter
- Token−Ring High−Performance Adapter (8fa2)
- Token−Ring High−Performance Adapter (8fc8)
- IBM PCI Token−Ring Adapter (14101800)
- IBM PCI Token−Ring Adapter (14103e00)

Drivers that do not support remote dump include the drivers for these network devices:

- ISA Ethernet Adapter
- Ethernet High−Performance LAN Adapter (8ef5)
- Ethernet High−Performance LAN Adapter (8f95)
- Gigabit Ethernet−SX PCI Adapter (14100401)

Flags

−c Specifies that dumps will not be compressed. The **−c** flag applies to only AIX Version 4.3.2 and later versions.

−C Specifies that all future dumps will be compressed before they are written to the dump device. The **−C** flag applies to only AIX Version 4.3.2 and later versions.

−d*Directory* Specifies the *Directory* the dump is copied to at system boot. If the copy fails at boot time, the **−d** flag ignores the system dump.

−D*Directory* Specifies the *Directory* the dump is copied to at system boot. If the copy fails at boot time, using the **−D** flag allows you to copy the dump to an external media.

Note: When using the **−d***Directory* or **−D***Directory* flags, the following error conditions are detected:

- *Directory* does not exist.
- *Directory* is not in the local journaled file system.
- *Directory* is not in the **rootvg** volume group.

−e Estimates the size of the dump (in bytes) for the current running system. If the dump will be compressed, then the size shown is the estimate of the size after compression.

−k Requires the key mode switch to be in the service position before a dump can be forced with the reset button or the dump key sequences. This is the default setting.

−K The reset button or the dump key sequences will force a dump with the key in the normal position, or on a machine without a key mode switch.

Note: On a machine without a key mode switch, a dump can not be forced with the reset button nor the key switch without this value set.

−l Lists the current value of the primary and secondary dump devices, copy directory, and **forcecopy** attribute.

−L Displays statistical information about the most recent system dump. This includes date and time of last dump, number of bytes written, and completion status. If the dump was compressed, then this flag shows both the original uncompressed size and the compressed size of the dump. The compressed size is the size of what was actually written to the dump device.

−P Makes permanent the dump device specified by **−p** or **−s** flags. The **−P** flag can only be used with the **−p** or **−s** flags.

−p *Device* Temporarily changes the primary dump device to the specified device. The device can be a logical volume or a tape device. For a network dump, the device can be a host name and a path name.

−q Suppresses all messages to standard output. If this flag is used with the **−l**, **−r**, **−z** or **−L** flag, the **−q** command will be ignored.

−r*Host***:***Path* Frees space used by the remote dump file on server *Host*. The location of the dump file is specified by the *Path*.

−s *Device* Temporarily changes the secondary dump device to the specified device. The device can be a logical volume or a tape device. For a network dump, the device can be a host name and a path name.

−z Determines if a new system dump is present. If one is present, a string containing the size of the dump in bytes and the name of the dump device will be written to standard output. If a new system dump does not exist, nothing is returned. After the **sysdumpdev −z** command is run on an existing system dump, the dump will no longer be considered recent.

If no flags are used with the **sysdumpdev** command, the default dump devices are used.

Security

Access Control: Only the root user can run this command.

Examples

1. To display current dump device settings, enter:

   ```
   sysdumpdev -l
   ```

2. To designate logical volume `hd7` as the primary dump device, enter:

   ```
   sysdumpdev -p /dev/hd7
   ```

3. To designate tape device `rmt0` as the secondary dump device, enter:

   ```
   sysdumpdev -s /dev/rmt0
   ```

4. To display information from the previous dump invocation, enter:

   ```
   sysdumpdev -L
   ```

5. To permanently change the database object for the primary dump device to `/dev/newdisk1`, enter:

   ```
   sysdumpdev -P -p /dev/newdisk1
   ```

6. To determine if a new system dump exists, enter:

   ```
   sysdumpdev -z
   ```

   If a system dump has occurred recently, output similar to the following will appear:

   ```
   4537344 /dev/hd7
   ```

7. To designate remote dump file `/var/adm/ras/systemdump` on host `mercury` for a primary dump device, enter:

   ```
   sysdumpdev -p mercury:/var/adm/ras/systemdump
   ```

   A : (colon) must be inserted between the host name and the file name.

8. To specify the directory that a dump is copied to after a system crash, if the dump device is `/dev/hd6`, enter:

   ```
   sysdumpdev -d /tmp/dump
   ```

   This attempts to copy the dump from `/dev/hd6` to `/tmp/dump` after a system crash. If there is an error during the copy, the system continues to boot and the dump is lost.

9. To specify the directory that a dump is copied to after a system crash, if the dump device is `/dev/hd6`, enter:

   ```
   sysdumpdev -D /tmp/dump
   ```

This attempts to copy the dump from `/dev/hd6` to the `/tmp/dump` directory after a crash. If the copy fails, you are prompted with a menu that allows you to copy the dump manually to some external media.

Related Information

The **crash** command, **mount** command, **ping** command, **savecore** command, **sysdumpstart** command.

System Dump Facility in *AIX Problem Solving Guide and Reference*.

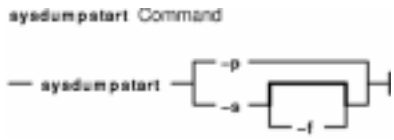
Setting up and running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

sysdumpstart Command

Purpose

Provides a command line interface to start a kernel dump to the primary or secondary dump device.

Syntax



sysdumpstart { **−p** | **−s** [**−f**] }

Description

The **sysdumpstart** command provides a command line interface to start a kernel dump to the primary or secondary dump device. When the dump completes, the system halts. Use the **crash** command to examine a kernel dump. Use the **sysdumpdev** command to reassign the dump device.

During a kernel dump, the following values can be displayed on the three−digit terminal display as follows:

0c0 Indicates that the dump completed successfully.

0c1 Indicates that an I/O occurred during the dump. This value only applies to AIX Version 4.2.1 or later.

0c2 Indicates that the dump is in progress.

0c4 Indicates that the dump is too smal l.

0c5 Indicates a dump internal error .

0c6 Prompts you to make the secondary dump device ready. This value does not apply for AIX Version 4.2.1 or later.

0c7 Indicates that the dump process is waiting for a response from the remote host.

0c8 Indicates that the dump was disabled. In this case, no dump device was designated in the system configuration object for dump devices. The **sysdumpstart** command halts, and the system continues running.

0c9 Indicates that a dump is in progress.

0cc Indicates that the system switched to the secondary dump device after attempting a dump to the primary device. This value only applies to AIX Version 4.2.1 or later.

You can use the Web−based System Manager Devices application (**wsm devices** fast path) to run this command. You could also use the System Management Interface Tool (SMIT) **smit sysdumpstart** fast path to run this command.

Flags

−f Suppresses the prompt to make the secondary dump device ready. This flag does not apply to AIX Version 4.2.1 or later.

−p Initiates a system dump and writes the results to the primary dump device.

−s Initiates a system dump and writes the results to the secondary dump device.

Security

Access Control: Only the root user can run this command.

Examples

1. To start a kernel dump to the primary dump device, enter:

   ```
   sysdumpstart −p
   ```

2. To start a kernel dump to the secondary dump device, enter:

   ```
   sysdumpstart −s
   ```

Related Information

System Dump Facility in *AIX Version 4.3 Problem Solving Guide and Reference*.

The **crash** command, **sysdumpdev** command.

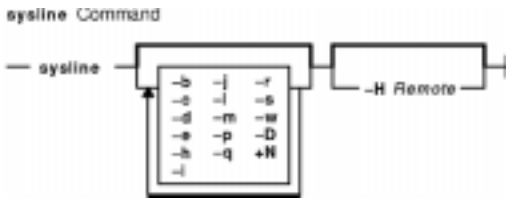
Setting up and running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

sysline Command

Purpose

Displays system status on the status line of a terminal.

Syntax



/usr/bin/sysline [**−b**] [**−c**] [**−d**] [**−e**] [**−h**] [**−i**] [**−j**] [**−l**] [**−m**] [**−p**] [**−q**] [**−r**] [**−s**] [**−w**] [**−D**] [**−H** *Remote*] [**+N**]

Description

The **sysline** command runs in the background and periodically displays system status information on the status line of the terminal. Not all terminals contain a status line. If no flags are specified, the **sysline** command displays the following status items:

- Time of day
- Current number of processes which may be run
- Number of users (followed by a u)
- Number of executable processes (followed by an r)
- Number of suspended processes (followed by an s)
- Number of users who have logged on and off since the last status report

Finally, if new mail has arrived, a summary of it is printed. If there is unread mail in your mailbox, an asterisk appears after the display of the number of users. The display is normally in reverse video (if your terminal supports this in the status line) and is right−justified to reduce distraction. Every fifth display is done in normal video to give the screen a chance to rest.

If you have a file named **.who** in your home directory, then the contents of that file is printed first. One common use of this feature is to alias the **chdir**, **pushd**, and **popd** commands to place the current directory stack in **/.who** after it changes the new directory.

If you have a file named **.syslinelock** in your home directory, then the **sysline** command will not update its statistics and write on your screen, it will just go to sleep for a minute. This is useful if you want to momentarily disable **sysline**. Note that it may take a few seconds from the time the lock file is created until you are guaranteed that **sysline** will not write on the screen.

Flags

−b	Beeps once every half hour and twice every hour.
−c	Clears the status line for five seconds before each redisplay.
−D	Prints out the current day/date before the time.
−d	Prints status line data in human readable format, debug mode.

−e Prints out only the information. Suppresses the control commands necessary to put the information on the bottom line. This option is useful for putting the output of the **sysline** command onto the mode line of an **emacs** window.

−H *Remote* Prints the load average on the remote host *Remote*. If the host is down, or is not sending *rwhod* packets, then the down time is printed instead. If the prefix **ucb** is present, then it is removed.

−h Prints out the host machine's name after the time.

−i Prints out the process ID of the **sysline** command process onto standard output upon startup. With this information you can send the alarm signal to the **sysline** process to cause it to update immediately. The **sysline** command writes to the standard error, so you can redirect the standard output into a file to catch the process ID.

−j Left−justifies the **sysline** command output on terminals capable of cursor movement on the status line.

−l Suppresses the printing of names of people who log in and out.

−m Suppresses mail check.

+N Updates the status line every *N* seconds. The default is 60 seconds.

−p Suppresses the report of the number of processes that are executable and suspended.

−q Suppresses the printout diagnostic messages if something goes wrong when starting up.

−r Suppresses reverse video display.

−s Prints the short form of a line by left−justifying **iff** (if and only if) escapes are not allowed in the status line. Some terminals (the Televideos and Freedom 100 for example) do not allow cursor movements (or other "intelligent" operations) in the status line. For these terminals, the **sysline** command normally uses blanks to cause right−justification. This flag disables the adding of blanks.

−w Prints the status on the current line of the terminal, suitable for use inside a one line window (Window mode).

Examples

To display the day and date, the number of processes which may be run, the number of users, and to clear the screen five seconds before it updates, enter:

```
sysline −Dcr
```

Note: This will only work on screens which have status line capabilities.

Files

/etc/utmp	Contains the names of users who are logged in.
/dev/kmem	Contains the process table.
/var/spool/rwho/whod.*	Contains who/Uptime information for remote hosts.
${HOME}/.who	Specifies information to print on the bottom line.
${HOME}/.syslinelock	Specifies that when it exists, **sysline** does not print.

Related Information

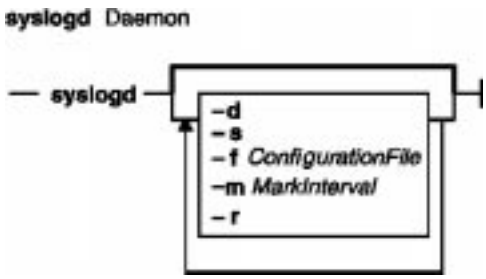
The **pstat** command, **vmstat** command.

syslogd Daemon

Purpose

Logs system messages.

Syntax



syslogd [**−d**] [**−s**] [**−f** *ConfigurationFile*] [**−m** *MarkInterval*] [**−r**]

Description

The **syslogd** daemon reads a datagram socket and sends each message line to a destination described by the **/etc/syslog.conf**configuration file. The **syslogd** daemon reads the configuration file when it is activated and when it receives a hangup signal.

The **syslogd** daemon creates the **/etc/syslog.pid** file, which contains a single line with the command process ID used to end or reconfigure the **syslogd** daemon.

A terminate signal sent to the **syslogd** daemon ends the daemon. The **syslogd** daemon logs the end−signal information and terminates immediately.

Each message is one line. A message can contain a priority code, marked by a digit enclosed in < > (angle braces) at the beginning of the line. Messages longer than 900 bytes may be truncated.

The **/usr/include/sys/syslog.h** include file defines the facility and priority codes used by the configuration file. Locally written applications use the definitions contained in the **syslog.h** file to log messages via the **syslogd** daemon.

Flags

−d	Turns on debugging.
−f *ConfigurationFile*	Specifies an alternate configuration file.
−m *MarkInterval*	Specifies the number of minutes between the **mark** command messages. If you do not use this flag, the **mark** command sends a message with **LOG_INFO** priority sent every 20 minutes. This facility is not enabled by a `selector` field containing an ***** (asterisk), which selects all other facilities.
−s	Specifies to forward a "shortened" message to another system (if it is configured to do so) for all the forwarding syslog messages generated on the local system.
−r	Suppresses logging of messages received from remote hosts.

Configuration File

The configuration file informs the **syslogd** daemon where to send a system message, depending on the message's priority level and the facility that generated it.

If you do not use the **−f** flag, the **syslogd** daemon reads the default configuration file, the **/etc/syslog.conf** file.

The **syslogd** daemon ignores blank lines and lines beginning with a # (pound sign).

Format

Lines in the configuration file for the **syslogd** daemon contain a `selector` field and an `action` field, separated by one or more tabs.

The `selector` field names a facility and a priority level. Separate facility names with a , (comma). Separate the facility and priority−level portions of the `selector` field with a . (period). Separate multiple entries in the same selector field with a ; (semicolon). To select all facilities, use an * (asterisk).

The `action` field identifies a destination (file, host, or user) to receive the messages. If routed to a remote host, the remote system will handle the message as indicated in its own configuration file. To display messages on a user's terminal, the destination field must contain the name of a valid, logged−in system user.

Facilities

Use the following system facility names in the `selector` field:

Facility	Description
kern	Kernel
user	User level
mail	Mail subsystem
daemon	System daemons
auth	Security or authorization
syslog	**syslogd** daemon
lpr	Line−printer subsystem
news	News subsystem
uucp	uucp subsystem
*****	All facilities

Priority Levels

Use the following message priority levels in the `selector` field. Messages of the specified priority level and all levels above it are sent as directed.

Priority Level	Description
emerg	Specifies emergency messages (**LOG_EMERG**). These messages are not distributed to all users. **LOG_EMERG** priority messages can be logged into a separate file for reviewing.
alert	Specifies important messages (**LOG_ALERT**), such as a serious hardware error. These messages are distributed to all users.
crit	Specifies critical messages not classified as errors (**LOG_CRIT**), such as improper login attempts. **LOG_CRIT** and higher−priority messages are sent to the system console.
err	Specifies messages that represent error conditions (**LOG_ERR**), such as an unsuccessful disk write.

warning Specifies messages for abnormal, but recoverable, conditions (**LOG_WARNING**).

notice Specifies important informational messages (**LOG_NOTICE**). Messages without a priority designation are mapped into this priority message.

info Specifies informational messages (**LOG_INFO**). These messages can be discarded, but are useful in analyzing the system.

debug Specifies debugging messages (**LOG_DEBUG**). These messages may be discarded.

none Excludes the selected facility. This priority level is useful only if preceded by an entry with an ***** (asterisk) in the same `selector` field.

Destinations

Use the following message destinations in the `action` field.

Destination	Description
File Name	Full path name of a file opened in append mode
@*Host*	Host name, preceded by **@** (at sign)
User[**,***User*][...]	User names
*****	All users

Examples

1. To log all mail facility messages at the debug level or above to the file **/tmp/mailsyslog**, enter:

   ```
   mail.debug /tmp/mailsyslog
   ```

2. To send all system messages except those from the mail facility to a host named `rigil`, enter:

   ```
   *.debug;mail.none @rigil
   ```

3. To send messages at the **emerg** priority level from all facilities, and messages at the **crit** priority level and above from the mail and daemon facilities, to users `nick` and `jam`, enter:

   ```
   *.emerg;mail,daemon.crit nick, jam
   ```

4. To send all mail facility messages to all users' terminal screens, enter:

   ```
   mail.debug *
   ```

Files

/etc/syslog.conf Controls the output of **syslogd**.

/etc/syslog.pid Contains the process ID.

Related Information

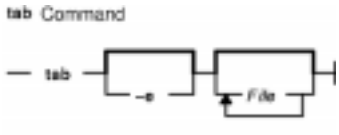
The **syslog** subroutine.

tab Command

Purpose

Changes spaces into tabs.

Syntax



tab [**−e**] [*File* ...]

Description

The **tab** command reads the file specified by the *File* parameter or standard input, and replaces spaces in the input with tab characters wherever the **tab** command can eliminate one or more spaces. If you specify a file with the *File* parameter, the **tab** command writes the resulting file back to the original file. If the input is standard input, the **tab** command writes to standard output. The **tab** command assumes that tab stops are set every eight columns, starting with column nine. The file name specified for the *File* parameter cannot exceed **PATH_MAX**−9 bytes in length.

Flag

−e Replaces only those spaces at the beginning of a line up to the first non−space character.

Example

To replace space characters in the `File` file with tab characters, enter:

```
tab File
```

File

/usr/bin/tab Contains the **tab** command.

Related Information

Files Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

Input and Output Redirection Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

The **expand** command, **newform** command, **unexpand** command, **untab** command.

tabs Command

Purpose

Sets tab stops on terminals.

Syntax

tabs [*TabSpec* ...] [**+m** [*Number*]] [**−T***Terminal* ...]

Description

The **tabs** command specifies tab stops on terminals that support remotely settable hardware tab characters. Tab stops are set according to the *TabSpec* parameter, and previous settings are erased.

When you use the **tabs** command, always refer to the leftmost column number as 1, even if your workstation refers to it as 0.

If you do not specify the *TabSpec* parameter, the default value is **−8**.

The following preset formats can be specified for the *TabSpec* parameter:

−a	Sets the tabs to 1, 10, 16, 36, and 72 (IBM System/370 Assembler first format).
−a2	Sets the tabs to 1, 10, 16, 40, and 72 (IBM System/370 Assembler second format).
−c	Sets the tabs to 1, 8, 12, 16, 20, and 55 (COBOL normal format).
−c2	Sets the tabs to 1, 6, 10, 14, and 49 (COBOL compact format, columns 1−6 omitted). With this code, the first column position corresponds to card column 7. One space gets you to column 8, and a tab gets you to column 12. Files using this code should include a format specification of: `<:t-c2 m6 s66 d:>`
−c3	Sets the tabs to 1, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62, and 67 (COBOL compact format with more tabs than **−c2**). These tabs provide the recommended format for COBOL. Files using this code should include a format specification of: `<:t-c3 m6 s66 d:>`
−f	Sets the tabs to 1, 7, 11, 15, 19, and 23 (FORTRAN).
−p	Sets the tabs to 1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, and 61 (PL/I).
−s	Sets the tabs to 1, 10, and 55 (SNOBOL).

−u Sets the tabs to 1, 12, 20, and 44.

In addition to the preset formats, the *TabSpec*s parameter can include:

−*Number* Sets regularly repeating tabs at every *Number* column. (The standard operating system tab setting is **−8**. The **−8** setting is required when using the **nroff** command with the **−h** flag.) Another special case is the **−0** setting, which implies no tabs at all. If more than 20 tabs are set, you must run the **tabs** command twice to clear them.

Number [,*Number*]...
Sets tabs at the specified column numbers (a comma−separated list in ascending order). You can specify up to 40 numbers. If any number except the first has a plus−sign prefix, the prefixed number is added to the previous number for the next setting. Thus, the tab list specified by **1,10,20,30** provides the same tab settings as the tab list specified by **1,10,+10,+10**.

−*Filep* Reads the first line of the *Filep* file for a format specification. If the **tabs** command finds a format specification, the **tabs** command sets tabs as specified. If the **tabs** command does not find a format specification, it sets tabs to the system default (**−8**).

It is sometimes convenient to maintain text files with nonstandard tab stop settings (tab stops that are not set at every eighth column). Such files must be converted to a standard format. This is often done by replacing all tab characters with the appropriate number of space characters, before they can be processed by any commands. A format specification occurring in the first line of a text file specifies how tab characters are to be expanded in the remainder of the file.

A format specification consists of a sequence of parameters separated by blanks and surrounded by **<:** and **:>**. Each parameter consists of a letter key, possibly followed immediately by a value. The following parameters are recognized:

ttabs Specifies the tab stop settings for a file. The value of *tabs* must be one of the following:

- A list of column numbers separated by commas, indicating tab stops set at the specified columns.
- A − (dash) followed immediately by an integer *n*, indicating tab stops set at intervals of *n* columns, that is, at 1+*n*, 1+2*n*, and so on.
- A − (dash) followed by the name of a preset tab stop specification.

Up to 40 numbers are allowed in a comma−separated list of tab stop settings. If any number (except the first one) is preceded by a plus sign, it is taken as an increment to be added to the previous value. Therefore, the formats **t1, 10, 20, 30** and **t1, 10, +10, +10** are considered identical.

Standard tab stops are specified by **t−8**, or, equivalently, **t1, 9, 17, 25**. This is the tab stop setting that most system utilities assume, and is the most likely setting to find at a terminal. The specification **t−0** specifies no tab stops at all.

The preset tab stop specifications that are recognized are as follow:

a	1, 10, 16, 36, 72 Assembler, IBM System/370, first format
a2	1, 10, 16, 40, 72 Assembler, IBM System/370, second format
c	1, 8, 12, 16, 20, 55 COBOL, normal format

c2	1, 6, 10, 14, 49 COBOL compact format (columns 1−6 omitted). Using this code, the first typed character corresponds to card column 7; one space gets you to column 8; and a tab gets you to column 12. Files using this tab stop setup should include a format specification as follows: `<:t-c2 m6 s66 d:>`
c3	1, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62, 67 COBOL compact format (columns 1−6 omitted) with more tab stops than **c2**. This is the recommended format for COBOL. The appropriate format specification is: `<:t-c3 m6 s66 d:>`
f	1, 7, 11, 15, 19, 23 FORTRAN
p	1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61 PL/I
s	1, 10, 55 SNOBOL
u	1, 12, 20, 44 UNIVAC 1100 Assembler

ssize Specifies a maximum line size. The value of *size* must be an integer. Size checking is performed after tab characters have been expanded, but before the margin is adjusted.

mmargin Specifies the number of space characters to be added to the beginning of each line. The value of *margin* must be an integer.

d Indicates that the line containing the format specification is to be deleted from the converted file. The *d* parameter takes no value.

e Indicates that the current format is valid only until another format specification is encountered in the file. The *e* parameter takes no value.

Default values, which are assumed for parameters not supplied, are **t−8** and **m0**. If the *s* parameter is not specified, no size checking is performed. If the first line of a file does not contain a format specification, the above defaults are assumed for the entire file. The following is an example of a line containing a format specification:

```
<:t5,10,15 s72:>
```

If a format specification can be disguised as a comment, it is not necessary to code the *d* parameter.

Flags

−T*Terminal* Identifies the terminal so the **tabs** command can set tabs and margins correctly. The *Terminal* variable is one of the terminals specified in the **greek** command. Supported values for the *Terminal* variable include:

ANSI Any ANSI terminal, such as a VT100 terminal.

hp Hewlett−Packard hardcopy terminals.

2621 Hewlett−Packard 2621.

2640 Hewlett−Packard 2640.

2645 Hewlett−Packard 2645.

Additional hardcopy terminals supported by the **tabs** command include:

- 1620
- 1620−12
- 1620−12−8
- 1700
- 1700−12
- 1700−12−8
- 300
- 300−12
- 300s
- 300s−12
- 40−2
- 4000a
- 4000a−12
- 43
- 450
- 450−12
- 450−12−8
- tn1200
- tn300
- oki

If you do not provide the **−T** flag, the value of the environment variable **TERM** is used. If the **−T** flag is provided with no value or if **−T** and **TERM** have invalid values, the error message `unknown terminal` is displayed and the command terminates.

+m *Number* Moves all tabs to the right the number of columns specified by the *Number* variable. This flag also sets the left margin to the column specified by the *Number* variable. If **m** is specified without a value, the default value for the Number variable is 10. The leftmost margin on most workstations is defined by **+m0**. The first column for tabs is defined as column 0 not column 1.

Note: If the same flag occurs more than once, only the last flag takes effect.

Exit Status

This command returns the following exit values:

0 Successful completion.

>0 An error occurred.

Examples

1. To set tabs every four spaces, enter:

   ```
   tabs −4
   ```

2. To set tabs every ten spaces on a VT100 terminal, enter:

   ```
   tabs −10 −TANSI
   ```

File

/usr/bin/tabs Contains the **tabs** command.

Related Information

The **greek** command, **nroff** command, **troff** command.

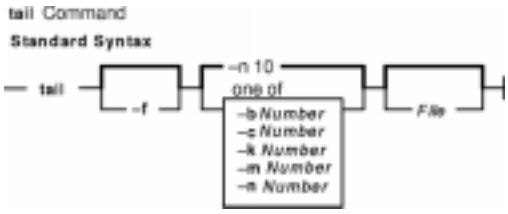
tail Command

Purpose

Writes a file to standard output, beginning at a specified point.

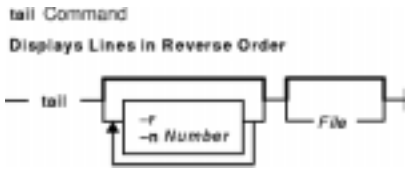
Syntax

Standard Syntax



tail [**−f**] [**−c** *Number* | **−n** *Number* | **−m***Number* | **−b** *Number* | **−k** *Number*] [*File*]

To Display Lines in Reverse Order



tail [**−r**] [**−n** *Number*] [*File*]

Description

The **tail** command writes the file specified by the *File* parameter to standard output beginning at a specified point. If no file is specified, standard input is used. The *Number* variable specifies how many units to write to standard output. The value for the *Number* variable can be a positive or negative integer. If the value is preceded by **+** (plus sign), the file is written to standard output starting at the specified number of units from the beginning of the file. If the value is preceded by **−** (minus sign), the file is written to standard output starting at the specified number of units from the end of the file. If the value is not preceded by **+** (plus sign) or **−** (minus sign), the file is read starting at the specified number of units from the end of the file.

The type of unit used by the *Number* variable to determine the starting point for the count is determined by the **−b**, **−c**, **−k**, **−m**, or **−n** flag. If one of these flags is not specified, the **tail** command reads the last ten lines of the specified file and writes them to standard output. This is the same as entering **−n 10** at the command line.

The **−m** flag provides consistent results in both single− and double−byte character environments. The **−c** flag should be used with caution when the input is a text file containing multibyte characters, because output can be produced that does not start on a character boundary.

Flags

−b*Number* Reads the specified file beginning at the 512−byte block location indicated by the *Number* variable.

−c *Number* Reads the specified file beginning at the byte location indicated by the *Number* variable.

−f If the input file is a regular file or if the *File* parameter specifies a FIFO (first−in−first−out), the **tail** command does not terminate after the last specified unit of the input file has been copied, but continues to read and copy additional units from the input file as they become available. If no *File* parameter is specified and standard input is a pipe, the **−f** flag is ignored. The **tail −f** command can be used to monitor the growth of a file being written by another process.

−k*Number* Reads the specified file beginning at the 1KB block location indicated by the *Number* variable.

−m *Number* Reads the specified file beginning at the multibyte character location indicated by the *Number* variable. Using this flag provides consistent results in both single− and double−byte character−code−set environments.

−n *Number* Reads the specified file beginning at the line location indicated by the *Number* variable.

−r Displays the output from the end of the file in reverse order. The default for the **−r** flag prints the entire file in reverse order. If the file is larger than 20,480 bytes, the **−r** flag displays only the last 20,480 bytes.

The **−r** flag is valid only with the **−n** flag. Otherwise, it is ignored.

Exit Status

This command returns the following exit values:

0 Successful completion.

>**0** An error occurred.

Examples

1. To display the last 10 lines of the `notes` file, enter:

   ```
   tail notes
   ```

2. To specify the number of lines to start reading from the end of the `notes` file, enter:

   ```
   tail −n 20 notes
   ```

3. To display the `notes` file a page at a time, beginning with the 200th byte, enter:

   ```
   tail −c +200 notes | pg
   ```

4. To follow the growth of a file, enter:

   ```
   tail −f accounts
   ```

This displays the last 10 lines of the `accounts` file. The **tail** command continues to display lines as they are added to the `accounts` file. The display continues until you press the Ctrl−C key sequence to stop it.

File

/usr/bin/tail Contains the **tail** command.

Related Information

Input and Output Redirection Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* describes how the operating system processes input and output.

Files Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* describes files, file types, and how to name files.

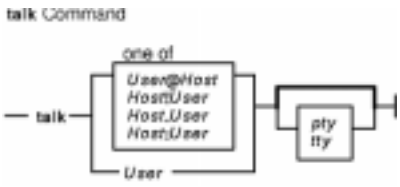
The **dd** command, **head** command, **more** command, **pg** command.

talk Command

Purpose

Converse with another user.

Syntax



talk {*User* | *User***@***Host* | *Host***!***User* | *Host***.***User* | *Host***:***User* } [*Tty*] [*Pty*]

Description

The **/usr/bin/talk** command allows two users on the same host or on different hosts to have an interactive conversation. The **talk** command opens both a send window and a receive window on each user's display. Each user is then able to type into the send window while the **talk** command displays what the other user is typing.

To initiate a conversation, a local user executes the **talk** command and specifies a remote user's login ID. The remote user's login ID can contain NLS characters. If the remote user is on a remote host, the name of the host must also be specified in one of the following ways:

```
User@Host
Host!User
Host.User
Host:User
```

When using full domain names, the only valid form for specifying the user and host is *User***@***Host*. For example, `michael@host17.dev.ibm.com` initiates a conversation with user `michael` at host `host17` in the `dev.ibm.com` domain.

When the local user initiates the conversation, a message is sent to the remote user, inviting a conversation. If the local user also specifies `tty`, the invitation message is sent only to the specified terminal. Otherwise, the invitation is sent to the remote user's login terminal. This usually is the console, but it may be another terminal. Once this invitation is received, the **talk** command displays two windows on the local user's terminal and displays progress messages until the remote user responds to the invitation.

> **Note:** If the remote user is running AIXwindows and has no other terminals open, the **talk** command cannot send an invitation.

To have the conversation, the remote user also has to execute the **talk** command from any terminal and specify the local user's account name and host name, if appropriate. When the remote user accepts the invitation, the **talk** command displays two windows on each user's terminal. One window displays what is typed by the local user; the other window displays what is typed by the remote user. To end the conversation, either user can press the Interrupt (Ctrl−C) key sequence and the connection is closed. The Interrupt key sequence can be displayed and modified using the **stty** command.

If the users involved in the conversation are using National Language Support (NLS) capabilities, their terminals must support the printing of NLS characters. The same is true for conversations using Kanji capabilities; the terminals being used must support the printing of Kanji characters.

The **talk** command requires a valid address to which to bind. The host name of the remote machine must be bound to a working network interface, which is usable by other network commands, such as the **ping** command. If a machine has no network interface, that is a standalone machine, it must bind its host name to the loopback address (127.0.0.1) in order for the **talk** command to work. For example, two users named `local` and `remote` on a standalone machine could initiate a conversation, using the **talk** command, by entering:

```
talk remote@loopback
```

To which user `remote` responds:

```
talk local@loopback
```

To disallow **talk** command invitations, the remote user can issue the **mesg** command.

> **Note:** The **talk** command uses the Talk 4.3 protocol, which is not compatible with 4.2 versions of the **talk** command.

Examples

1. To talk to a user logged in on a remote host, enter:

   ```
   talk dale@host2
   ```

   In this example, the local user wants to talk with user `dale` who is logged in on `host2`.

2. To talk to a user only if that user is logged in on the console of a remote host, enter:

   ```
   talk dale@host2 console
   ```

User `dale` receives this message only if logged in on the console at `host2`.

Related Information

Network Overview in *AIX Version 4.3 System User's Guide: Communications and Networks*.

The **mesg** command, **stty** command.

The **named** daemon, **talkd** daemon.

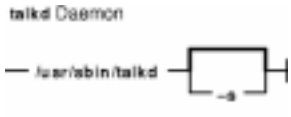
Conversing with a Remote User in *AIX Version 4.3 System User's Guide: Communications and Networks*.

talkd Daemon

Purpose

Provides the server function for the **talk** command.

Syntax



/usr/sbin/talkd [**−s**]

Description

> **Note:** The **talkd** daemon is normally started by the **inetd** daemon. It can also be controlled from the command line, using SRC commands.

The **/usr/sbin/talkd** daemon is the server that notifies a user (the recipient) that another user (the caller) wants to initiate a conversation. The daemon sets up the conversation if the recipient accepts the invitation. The caller initiates the conversation by executing the **talk** command specifying the recipient. The recipient accepts the invitation by executing the **talk** command specifying the caller.

The **talkd** daemon listens at the socket defined in the **/etc/services** file. When the **talkd** daemon receives a LOOK_UP request from a local or remote **talk** process, the **talkd** daemon scans its internal invitation table for an entry that pairs the client process (the local or remote **talk** process) with a caller.

If no entry exists in the invitation table, the **talkd** daemon assumes that the client process is the caller. The **talkd** daemon then receives the client process' ANNOUNCE request. The **talkd** daemon broadcasts an invitation on the remote computer where the recipient first logged in (unless the caller specifies a particular tty device). This terminal usually is the console, but it may be another terminal.

Otherwise, the invitation is sent to the terminal that the second user first logged in to. This usually is the console, but it may be another terminal.

If an entry does exist in the **talkd** daemon's internal invitation table, the **talkd** daemon assumes that the client is the recipient. The **talkd** daemon returns the appropriate rendezvous address to the **talk** process for the recipient. The recipient process then establishes a stream connection with the caller process.

> **Note:** The **talkd** daemon uses the Talk 4.3 protocol, which is not compatible with 4.2 versions of the **talk** process. The subserver name for the 4.3 protocol is **ntalk**.

Changes to the **talkd** daemon can be made using the System Management Interface Tool (SMIT) or System Resource Controller (SRC), by editing the **/etc/inetd.conf** or /**etc**/**services** file. Entering `talkd` at the command line is not recommended. The **talkd** daemon is started by default when it is uncommented in the **/etc/inetd.conf** file.

The **inetd** daemon get its information from the /**etc**/**inetd.conf** file and the /**etc**/**services** file.

After changing the **/etc/inetd.conf** or /**etc**/**services** file, run the **refresh−s inetd** or **kill −1** *InetdPID* command

to inform the **inetd** daemon of the changes to its configuration file.

Debugging messages are sent to the **syslogd** daemon.

> **Note:** The **talkd** daemon should be controlled using the System Management Interface Tool (SMIT) or by changing the **/etc/inetd.conf** file.

Manipulating the talkd Daemon with the System Resource Controller

The **talkd** daemon is a subserver of the **inetd** daemon, which is a subsystem of the System Resource Controller (SRC). The **talkd** daemon is a member of the **tcpip** SRC subsystem group. This daemon is enabled by default in the **/etc/inetd.conf** file and can be manipulated by the following SRC commands:

startsrc	Starts a subsystem, group of subsystems, or a subserver.
stopsrc	Stops a subsystem, group of subsystems, or a subserver.
lssrc	Gets the status or a subsystem, group or subsystems, or a subserver.

Flags

−s Turns on socket−level debugging.

Examples

1. To start the **talkd** daemon, enter the following:

   ```
   startsrc −t ntalk
   ```

   This command starts the **talkd** subserver.

2. To stop the **talkd** daemon normally, enter the following:

   ```
   stopsrc −t ntalk
   ```

   This command allows all pending connections to start and existing connections to complete but prevents new connections from starting.

3. To force stop the **talkd** daemon and all **talkd** connections, enter the following:

   ```
   stopsrc −t −f ntalk
   ```

   This command terminates all pending connections and existing connections immediately.

4. To display a short status report about the **talkd** daemon, enter the following:

   ```
   lssrc −t ntalk
   ```

This command returns the daemon's name, process ID, and state (active or inactive).

Files

/etc/utmp Contains data about users currently logged in.

Related Information

The **kill** command, **lssrc** command, **refresh** command, **startsrc** command, **stopsrc** command, **talk** command.

The **inetd** daemon, **syslogd** daemon.

The **/etc/inetd.conf** file format.

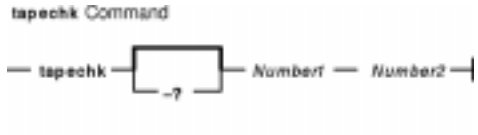
TCP/IP Daemons in *AIX Version 4.3 System Management Guide: Communications and Networks*.

tapechk Command

Purpose

Performs consistency checking on the streaming tape device.

Syntax



tapechk [**−?**] *Number1 Number2*

Description

The **tapechk** command performs rudimentary consistency checking on an attached streaming tape device. Some hardware malfunctions of a streaming tape drive can be detected by simply reading a tape. The **tapechk** command provides a way to perform tape reads at the file level.

Because the streaming tape drive cannot backspace over physical data blocks or files, the **tapechk** command rewinds the tape to its starting position prior to each check. This command either checks data for the next number of files specified by the *Number1* parameter or skips the next number of files specified by the *Number2* parameter. If you do not specify any parameters, the **tapechk** command rewinds the tape and checks only the first physical block.

The **tapechk** command uses the device in the **TAPE** environment variable if it is defined. Otherwise, the default tape device is **/dev/rmt0**.

> **Note:** The **backup** command allows you to archive files selectively or as an entire file system. It writes data as a continuous stream terminated by a file mark, regardless of the number of files specified. The **tapechk** command perceives each stream of data as a single file, which is important when you specify numeric parameters.

Although you can use the **tapechk** command on any streaming tape cartridge, it is primarily designed for checking tapes written by the **backup** command.

Flag

−? Explains the format of the **tapechk** command.

> **Note:** If you specify the **−?** flag, it must be specified before the *Number1* and *Number2* parameters.

Exit Status

This command returns the following exit values:

0 Successful completion.

>0 An error occurred.

Example

To check the first three files on a streaming tape device, enter:

```
tapechk 3
```

File

/usr/sbin/tapechk Contains the **tapechk** command.

Related Information

The **backup** command.

The **rmt** special file.

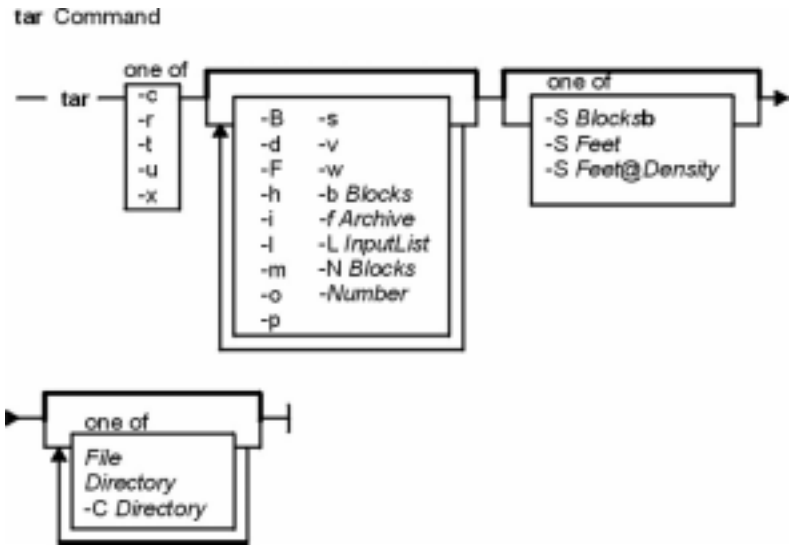
Tape Drives in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

tar Command

Purpose

Manipulates archives.

Syntax



tar { **−c** | **−r** | **−t** | **−u** | **−x** } [**−b***Blocks*] [**−B**] [**−d**] [**−F**] [**−h**] [**−i**] [**−L** *InputList*] [**−l**] [**−m**] [**−N** *Blocks*][**−o**] [**−p**] [**−s**] [**−v**] [**−w**] [**−***Number*] [**−f***Archive*] [**−S***Blocks***b** | **−S***Feet* | **−S** *Feet@Density*] [*File* | *Directory* | **−C***Directory*] ...

Description

> **Attention:** Because of limitations on header block space in the **tar** command, user numbers (UIDs), and group identification numbers (GIDs) larger than 65,535 will be corrupted when restored to certain systems. The size constraint affects only the ownership and permissions causing no damage to the data. Corruption of the ownership occurs on the following systems:

- Those that do not use `uname` and `gname` fields to check ownership.
- Those that do not have the same user and group IDs as the archiving system.

Note:

1. The **tar** command is not enabled for files greater than 2 Gig in size due to limitations imposed by XPG/4 and POSIX.2 standards.
2. **tar** does not preserve the sparse nature of any file that is sparsely allocated. Any file that was originally sparse before the restoration will have all space allocated within the filesystem for the size of the file.

The **tar** command manipulates archives by writing files to, or retrieving files from an archive storage medium. The files used by the **tar** command are represented by the *File* parameter. If the *File* parameter refers to a directory, then that directory and recursively all files and directories within it are referenced as well.

The **tar** command looks for archives on the default device (usually tape), unless you specify another device with the **−f***Archive* flag. When specifying path names that are greater than 100 characters for the United States Tape Archiver (USTAR) format, remember that the path name is composed of a prefix buffer, a /

(slash), and a name buffer.

The prefix buffer can be a maximum of 155 bytes and the name buffer can hold a maximum of 100 bytes. If the path name cannot be split into these two parts by a slash, it cannot be archived. This limitation is due to the structure of the **tar** archive headers, and must be maintained for compliance with standards and backwards compatibility. In additon, the length of a destination for a hard or symbolic link (the 'link name') cannot exceed 100 bytes.

When writing to an archive, the **tar** command uses a temporary file (the /**tmp**/**tar*** file) and maintains in memory a table of files with several links. You receive an error message if the **tar** command cannot create the temporary file, or if there is not enough memory available to hold the link tables.

Two groups of flags exist for the **tar** command: the required flags and the optional flags. The required flags control the actions of the **tar** command and include the **−c**, **−r**, **−t**, **−u**, and **−x** flags. At least one required flag must be selected for the **tar** command to function. Having selected a required flag, you can select an optional flag but none are necessary to control the **tar** command.

Notes:

1. When the storage device is an ordinary file or a block special file, the **−u** and **−r** flags backspace. However, raw magnetic tape devices do not support backspacing. So when the storage device is a raw magnetic tape, the **−u** and **−r** flags rewind the tape, open it, and then read it again.
2. Records are one block long on block magnetic tape, but they are typically less than half as dense on raw magnetic tape. As a result, although a blocked raw tape must be read twice, the total amount of tape motion is less than when reading one−block records from a block magnetic tape once.
3. The structure of a streaming tape device does not support the addition of information at the end of a tape. Consequently when the storage device is a streaming tape, the **−u** and **−r** flags are not valid options. An attempt to use these flags results in the following error message:

   ```
   tar: Update and Replace options not valid for a
   streaming tape drive.
   ```

4. No recovery exists from tape errors.
5. The performance of the **tar** command to the IBM 9348 Magnetic Tape Unit Model 12 can be improved by changing the default block size. To change the block size, enter the following at the command line:

   ```
   chdev −l <device_name> −a block_size=32k
   ```

For more information on using tape devices see the **rmt** special file.

Flags

Flags for the **tar** command are in two groups, the required and the optional. You must supply at least one required flag to control the **tar** command.

Required Flags

−c	Creates a new archive and writes the files specified by one or more *File* parameters to the beginning of the archive.
−r	Writes the files specified by one or more *File* parameters to the end of the archive. This flag is not valid for any tape devices because such devices do not support the addition of information at the end of a tape.
−t	Lists the files in the order in which they appear in the archive. Files can be listed more

than once.

−u Adds the files specified by one or more *File* parameters to the end of the archive only if the files are not in the archive already, or if they have been modified since being written to the archive. The **−u** flag is not valid for any tape devices because such devices do not support the addition of information at the end of a tape.

−x Extracts the files specified by one or more *File* parameters from the archive. If the *File* parameter refers to a directory, the **tar** command recursively extracts that directory from the archive. If you do not specify the *File* parameter, the **tar** command extracts all of the files from the archive. When an archive contains multiple copies of the same file, the last copy extracted overwrites all previously extracted copies. If the file being extracted does not already exist on the system, the file is created. If you have the proper permissions, the **tar** command restores all files and directories with the same owner and group IDs as they have on the tape. If you do not have the proper permissions, the files and directories are restored with your owner and group IDs. It is not possible to ask for any occurrence of a file other than the last.

Optional Flags

−B Forces input and output blocking to 20 blocks per record. With this option, the **tar** command can work across communications channels where blocking may not be maintained.

−b *Blocks* Specifies the number of 512 bytes blocks per record. Both the default and the maximum is 20, which is appropriate for tape records. Due to the size of interrecord gaps, tapes written with large blocking factors can hold much more data than tapes with only one block per record.

The block size is determined automatically when tapes are read (the **−x** or **−t** function flags). When archives are updated with the **−u** and **−r** functions, the existing record size is used. The **tar** command writes archives using the specified value of the *Blocks* parameter only when creating new archives with the **−c** flag.

For output to ordinary files with the **−f** flag, you can save disk space by using a blocking factor that matches the size of disk blocks (for example, the **−b4** flag for 2048−byte disk blocks).

−C *Directory* Causes the **tar** command to perform a **chdir** subroutine to the directory specified by the *Directory* variable. Using the **−C** flag allows multiple directories not related by a close common parent to be archived, using short relative path names. For example, to archive files from the **/usr/include** and **/etc** directories, you might use the following command:

```
tar c −C /usr/include File1 File2 −C /etc File3 File4
```

The **−C***Directory* flag must appear after all other flags and can appear in the list of file names given.

−d Makes separate entries for block files, special character files, and first−in−first−out (FIFO) piped processes. Normally, the **tar** command will not archive these special files. When writing to an archive with the **−d** flag, the **tar** command makes it possible to restore empty directories, special files, and first−in−first−out (FIFO) piped processes with the **−x** flag.

> **Note:** Although anyone can archive special files, only a user with root user authority can extract them from an archive.

−F Checks the file type before archiving. Source Code Control Systems (SCCS), Revision Control Systems (RCS), files named **core**, **errs**, **a.out**, and files ending in **.o** (dot o) are not archived.

−f *Archive* Uses the *Archive* variable as the archive to be read or written. When this flag is not specified, the **tar** command uses a system−dependent default file name of the form **/dev/rmt0**. If the *Archive* variable specified is − (minus sign), the **tar** command writes to standard output or reads from standard input. If you write to standard output, the **−c** flag

must be used.

−h Forces the **tar** command to follow symbolic links as if they were normal files or directories. Normally, the **tar** command does not follow symbolic links.

−i Ignores header checksum errors. The **tar** command writes a file header containing a checksum for each file in the archive. When this flag is not specified, the system verifies the contents of the header blocks by recomputing the checksum and stops with a directory checksum error when a mismatch occurs. When this flag is specified, the **tar** command logs the error and then scans forward until it finds a valid header block. This permits restoring files from later volumes of a multi−volume archive without reading earlier volumes.

−L *InputList* Writes the files and directories listed in the *InputList* variable to the archive. Directories from the *InputList* variable are not treated recursively. For directories contained in the *InputList* variable, the **tar** command writes only the directory to the archive, not the files and subdirectories rooted in the directory. If additional files and directories follow the *InputList* variable on the command line, the contents of the *InputList* variable are archived after these files and directories. These additional files or directories are archived with their default behavior, which is to treat them recursively.

−l Writes an error message to standard output for each file with a link count greater than 1 whose cooresponding links were not also archived. For example, if **file1** and **file2** are hard−linked together and only **file1** is placed on the archive, then the −l flag will issue an error message. Error messages are not displayed if the **−l** flag is not specified.

−m Uses the time of extraction as the modification time. The default is to preserve the modification time of the files.

−N *Blocks* Allows the **tar** command to use very large clusters of blocks when it deals with streaming tape archives. Note however, that on input, the **tar** command cannot automatically determine the block size of tapes with very long block sizes created with this flag. In the absence of a **−N** *Blocks* flag, the largest block size that the **tar** command can automatically determine is 20 blocks.

−o Provides backwards compatibility with older versions (non−AIX) of the **tar** command. When this flag is used for reading, it causes the extracted file to take on the User and Group ID (UID and GID) of the user running the program, rather than those on the archive. This is the default behavior for the ordinary user.

−p Says to restore fields to their original modes, ignoring the present umask. The **setuid**, **setgid**, and tacky bit permissions are also restored to the user with root user authority.

−s Tries to create a symbolic link If the **tar** command is unsuccessful in its attempt to link (regular link) two files with the **−s** flag.

−S*Blocks***b**, **−S***Feet*, **−S***Feet***@***Density* Specifies the number of 512KB blocks per volume (first format), independent of the tape blocking factor. You can also specify the size of the tape in feet by using the second form, in which case the **tar** command assumes a default *Density* variable. The third form allows you to specify both tape length and density. Feet are assumed to be 11 inches long to be conservative. This flag lets you deal more easily with multivolume tape archives, where the **tar** command must be able to determine how many blocks fit on each volume.

Note:

1. Tape drives vary in density capabilities. The *Density* variable calculates the amount of data a system can fit on a tape.
2. When using 1/4−inch tape devices, be sure to take into account the number of tracks on the tape device when specifying the value for the *Feet* variable. For example, a 4−track,1/4−inch tape drive with a 600−foot tape and a density of 8000 bpi can be specified using the **−S***Feet***@***Density* flag as follows:
```
−S 2400@8000
```

where 600 feet multiplied by 4 tracks equals 2400 feet.

−v	Lists the name of each file as it is processed. With the **−t** flag, **−v** gives more information about the tape entries, including file sizes, times of last modification, User Number (UID), Group Number (GID), and permissions.
−w	Displays the action to be taken, followed by the file name, and then waits for user confirmation. If the response is affirmative, the action is performed. If the response is not affirmative, the file is ignored.
−*Number*	Uses the **/dev/rmt***Number* file instead of the default. For example, the **−2** flag is the same as the **−f/dev/rmt2** file.

Exit Status

This command returns the following exit values:

0 Successful completion.

>0 An error occurred.

Examples

1. To write the `file1` and `file2` files to a new archive on the default tape drive, enter:
   ```
   tar −c file1 file2
   ```

2. To extract all files in the `/tmp` directory from the archive file on the `/dev/rmt2` tape device and use the time of extraction as the modification time, enter:

   ```
   tar −xm −f/dev/rmt2 /tmp
   ```

3. To create a new archive file that contains the `file1` file and pass the archive file to the **dd** command to be written to the `/dev/rmt1` device, enter:

   ```
   tar −cvf − file1 | dd of=/dev/rmt1 conv=sync
   ```

4. To display the names of the files in the `out.tar` disk archive file on the current directory, enter:

   ```
   tar −vtf out.tar
   ```

5. To expand the compressed **tar** archive file, `fil.tar.z`, pass the file to the **tar** command, and extract all files from the expanded **tar** archive file, enter:

   ```
   zcat fil.tar.Z | tar −xvf −
   ```

6. To archive the contents of `/usr/include` and `/usr/bin` files using short relative path names, enter:

   ```
   cd /usr
   tar −cvf/dev/rmt0 −C./include . −C ../bin .
   ```

   > **Note:** When specifying multiple instances of the **−C** flag with relative path names, the user must take the previous **−C** flag request into account.

7. To archive to an 8−mm device when using the **−S** flag, enter:

   ```
   tar −cvf /dev/rmt0 −S 4800000b /usr
   ```

   > **Note:** When archiving to an 8−mm device, the **−S***Feet* and **−S***Feet***@***Density* flags are not recommended, since the 8−mm device does not use the concept of density when writing to a tape.

Files

/dev/rmt0	Specifies the default tape device.
/bin/tar	Specifies the symbolic link to the **tar** command.
/usr/bin/tar	Contains the **tar** command.
/tmp/tar*	Specifies a temporary file.

Note: In version 3.2, the entire **/bin** directory is a symbolic link to **/usr/bin**.

Related Information

The **cat** command, **dd** command, **pax** command.

The **rmt** special file.

File Systems Overview for System Management in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains file system types, management, structure, and maintenance.

Directory Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* explains working with directories and path names.

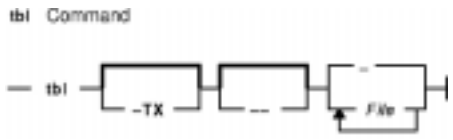
Files Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* provides information on working with files.

tbl Command

Purpose

Formats tables for the **nroff** and **troff** commands.

Syntax



tbl [**−TX**] [**−−**] [*File...* | **−**]

Description

The **tbl** command is a preprocessor that formats tables for the **nroff** and **troff** commands. It reads one or more files. If no *File* parameter or **−** (minus sign) is specified as the last parameter, the command reads standard input by default. It copies the input unchanged to standard output, except for text between lines containing **.TS** and **.TE**. The **tbl** command reformats such text, which describes tables, without altering the **.TS** and **.TE** lines.

Depending on the target output device, the output formatted by the **nroff** command may need to be post−processed by the **col** command to produce correct output.

> **Note:** To minimize the volume of data passed through pipelines, enter the **tbl** command first when using it with the **eqn** or **neqn** command.

Input Format

The **tbl** command processes text that is displayed within the following format:

```
[ {.DS .DF} ]
.TS
Options ;
Format .
Data
.TE
[.DE]
```

To include short tables in an **mm** macro document, enclose them within the **.DS** (or **.DF**) and **.DE** macro pair.

Options

Following are the available global options for the input format:

Option	Purpose
center or **CENTER**	Centers the table.

expand or **EXPAND**	Expands to PAGE length.
box or **BOX**	Encloses in a box.
allbox or **ALLBOX**	Boxes all tables.
doublebox or **DOUBLEBOX**	Encloses in double boxes.
tab(*Character***)** or **TAB(***Character***)**	Changes the tab character to the *Character* value.
linesize(*Number***)** or **LINESIZE(***Number***)**	Sets all lines to the thickness of the point size specified by the *Number* value.
delim(*XY***)** or **DELIM(***XY***)**	Recognizes the *X* and *Y* variables as **eqn** command delimiters.
;	Denotes end of options.

Format

The *Format* variable in the Input Format describes the format of text. Each format line (the last of which must end with a period) describes all remaining lines of the table. A single−key letter describes each column of each line of the table. Follow this key letter with specifiers that determine the font and point size of the corresponding item, indicate where vertical bars are to appear between columns, and determine such things as

column width and intercolumn spacing. The following are the available key letters:

l or **L**	Left−adjusts column.
r or **R**	Right−adjusts column.
c or **C**	Centers column.
n or **N**	Numerically aligns column. **Note:** Numerically aligned data, **n** or **N** format specification, are based upon the locale that is specific for *RADIXCHAR*, which is assumed to be a single character. The alignment can also be determined using the \& (backslash, ampersand) character sequence independent of the presence of any *RADIXCHAR* characters. If more than one *RADIXCHAR* character is displayed in a numerically aligned field, the last one is used for alignment. If no *RADIXCHAR* characters are displayed in a particular column, the alignment is based on the last ASCII arabic numeral. If there is no ASCII numeral and no *RADIXCHAR* character in a column, the data is centered.
a or **A**	Left−adjusts subcolumn.
s or **S**	Spans item horizontally.
t or **T**	Pushes vertical spans to top.
v or **V**	Adjusts vertical line spacing.
^	Spans item vertically.
u or **U**	Moves item half−line up.
z or **Z**	Indicates zero−width item.
−	Indicates horizontal line.
=	Indicates double horizontal line.
|	Indicates vertical line.
||	Indicates double vertical line.
b or **B**	Indicates boldface item.
i or **I**	Indicates italic item.
f*Character* or **F***Character*	Changes to the font specified by the *Character* variable.
p*Number* or **P***Number*	Changes to the size specified by the *Number* variable.
w(*Number***)** or **W(***Number***)**	Sets minimum column width equal to the *Number* variable value.
NumberNumber	Spaces between columns.
e or **E**	Makes equal−width columns.
.	Ends format.

Data

Handling data within the input format, especially for tables, uses the following line commands:

T{...T}	Indicates text block, as follows: *Data*<TAB>**T{** *Text Block* **T}**<TAB>*Data*
_	Writes short horizontal line.

\R*X*	Repeats the *X* parameter value across a column.
\^	Indicates that above item spans downward into this row.
.T&	Starts new format.
.TS H, **.TH**, and **.TE**	Allows multi−page tables with column headings repeated on each page. (This is a feature of the **mm** macros.)

Parameters

File Specifies the files that the **tbl** command will be processing.

Flags

−TX Uses only full vertical line motions, making the output suitable for line printers and other devices that do not have partial vertical line motions.

−− (double dash) Indicates the end of flags.

− Forces input to be read from standard input.

Examples

The following example shows coded input, and associated table output of the **tbl** command. The @ (at sign) is used in input to represent an input tab character.

Input

```
.TS
center box ;
cB s s
cI | cI s
^ | c c
l | n n .
Household Population
_
Town@Households
@Number@Size
=
Bedminster@789@3.26
Bernards Twp.@3087@3.74
Bernardsville@2018@3.30
Bound Brook@3425@3.04
Bridgewater@7897@3.81
Far Hills@240@3.19
.TE
```

Output

Household Population		
Town	*Households*	
	Number	Size
Bedminster	789	3.26
Bernards Twp.	3087	3.74
Bernardsville	2018	3.30
Bound Brook	3425	3.04
Bridgewater	7897	3.81
Far Hills	240	3.19

Related Information

The **col** command, **eqn** command, **mm** command, **mmt** command, **mvt** command, **neqn** command, **nroff** command, **soelim** command, **troff** command.

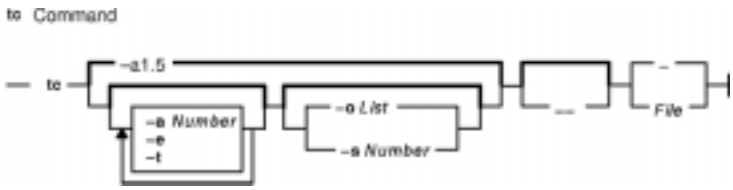
The **mm** macro package, **mv** macro package.

tc Command

Purpose

Interprets text into the **troff** command output for the Tektronix 4015 system.

Syntax



tc [**−t**] [**−e**] [**−a** *Number*] [**−o** *List* | **−s** *Number*] [**−−**] [*File* | **−**]

Description

The **tc** command interprets input as output from the **troff** command. The **tc** command reads one or more English−language files. If no file is specified or the **−** (minus sign) flag is specified as the last parameter, standard input is read by default. The standard output of the **tc** command is intended for a Tektronix 4015 (a 4014 terminal with ASCII and APL character sets). The various typesetter sizes are mapped into the 4014's four sizes. The entire **troff** command character set is drawn using the 4014 character generator, with overstruck combinations where necessary.

At the end of each page, the **tc** command waits for a new−line character from the keyboard before continuing to the next page. While it waits, the following commands are recognized:

!*Command*	Sends the value of the *Command* variable to the shell.
−e	Does not erase before each page.
−*Number*	Skips backward the specified number of pages.
−a*Number*	Sets the aspect ratio to the value of the *Number* variable.
?	Prints a list of available options.

Note: The **tc** command does not distinguish among fonts.

Parameters

File Specifies the English−language text files to be interpreted as output from the **troff** command.

Flags

−a*Number*	Sets the aspect ratio to the specified number. The default is 1.5.
−e	Does not erase before each page.
−o*List*	Prints only the pages enumerated in the *List* variable. The list consists of pages and page ranges (for example, 5−17) separated by commas. The range *Number*− goes from the *Number* variable value to end; the range −*Number* goes from the beginning to and including the page specified by the *Number* variable.
−s*Number*	Skips the first specified number of pages.

−t	Does not wait between pages when directing output into a file.
−	Reads from standard input.
−−	(double dash) Indicates the end of flags.

Example

To use the **tc** command in a pipeline with the **troff** command, enter:

```
troff [Flag...] [File...] | tc
```

Related Information

The **nroff** command, **troff** command.

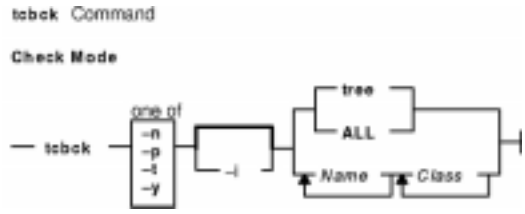
tcbck Command

Purpose

Audits the security state of the system.

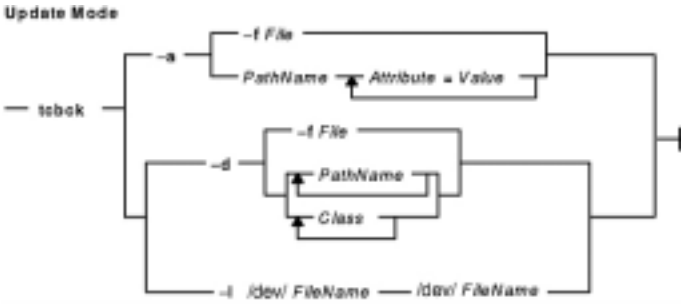
Syntax

Check Mode



tcbck { **−n** | **−p** | **−t**| **−y** } [**−i**] { **ALL** | **tree** | { *Name* ... *Class* ... } }

Update Mode



tcbck −a −f *File* | *PathName Attribute* **=** *Value ...*

OR

tcbck −d −f*File* | { *PathName* ... | *Class* ... }

OR

tcbck −l /dev/*filename* /dev/*filename*

Description

The **tcbck** command audits the security state of the system by checking the installation of the files defined in the **/etc/security/sysck.cfg** file (the **sysck** database). Each file definition in the **/etc/security/sysck.cfg** file can include one or more attributes that describe proper installation. When invoked with no flags and with no parameters, the **tcbck** command prints a synopsis of its syntax.

The **tcbck** database usually defines all the files and programs that are part of the trusted computing base, but the root user or a member of the security group can choose to define only those files considered to be security−relevant.

> **Note:** This command writes its messages to **stderr**.

Flags

−a Adds or updates file definitions in the **sysck** database.

−d Deletes file definitions from the **sysck** database.

−f *File* Specifies that file definitions be read from *File*.

−i Excludes filesystems under directories listed in the **treeck_nodir** attribute when the **tree** option is specified. This flag only applies to AIX Version 4.2 and above.

−l (Lowercase L) Adds entries to the **sysck.cfg** file for **/dev/** files that the administrator would like registered with the Trusted Computing Base.

−n Specifies the checking mode and indicates that errors are to be reported, but not fixed.

−p Specifies the checking mode and indicates that errors are to be fixed, but not reported.

−t Specifies the checking mode and indicates that errors are to be reported with a prompt asking whether the error should be fixed.

−y Specifies the checking mode and indicates that errors are to be fixed and reported.

Modes of Operation

The **tcbck** command has two modes of operation: check mode, and update mode. A description of each mode follows:

Check Mode

In Check mode, the **tcbck** command checks file definitions against the installed files. You can check all the file definitions in the **sysck** database (the **/etc/security/sysck.cfg** file) by specifying the **ALL** value, or all the files in the file system tree by specifying the **tree** value. If you prefer to check specific files, you can use the *Name* parameter to give the path names of individual files or the *Class* parameter to group several files into a logical group that is defined by a class name, such as audit. You must select one of the following: the **ALL** or **tree** values, or one or more files identified by the *Class* or *Name* parameter.

If the **tree** value is the selection criterion, all the files in the file system tree are checked to ensure that all the relevant files are defined in the **sysck** database. Files defined in the **tcbck** database are checked against their definitions. Files not in the **tcbck** database must *not*:

- Have the **trusted computing base** attribute set.
- Be **setuid** or **setgid** to an administrative ID.
- Be linked to a file in the **tcbck** database.
- Be a device special file.

If the **tcbck** command is running in check mode with both the **tree** value and the **−t** flag and an error occurs, the command provides an error message and prompts you for a decision on how or whether the error should be corrected. If you decide not to delete the file or turn off illegal permissions, you are prompted for a decision on updating the database. If you request an update, the system supplies missing information, such as the name of the file, the link, or the unregistered device name.

A flag (**−n**, **−p**, **−t**, **−y**) also must be included to specify check mode and identify the method of error handling. If there is a duplicate stanza in the **/etc/security/sysck.cfg** file, an error is reported, but not fixed.

Updating the Vital Product Database (VPD) involves defining the **type**, **checksum**, and **size** attributes of each file to the VPD manager. This information is used to verify a correct installation. If these attributes are not defined in **−f***File*, they are computed when the program is installed or updated. The **checksum** attribute is computed with a method specifically defined for the VPD manager. Refer to the "Fixing Errors" section for more information on file attributes.

The only file definitions modified during an update are the new definitions that indicate a file is part of the trusted computing base (TCB). The *File* parameter is the stanza file that contains the file definitions in **tcbck** format, and is defined in the **/etc/security/sysck.cfg** file. When the update is complete, the files are checked against their file definitions in the stanza file and errors are fixed and reported.

Programs that require **setuid** or **setgid** privilege must be in the **tcbck** database, or these privileges will be cleared when the **tcbck** command runs in Check mode.

Update Mode

In update mode, the **tcbck** command adds (**−a**), deletes (**−d**), or modifies file definitions in the **/etc/security/sysck.cfg** file for the file specified by the *File* parameter, the *PathName* parameter, or the *Class* parameter. The *Class* parameter permits you to group several files into a logical group that is defined by a class name, such as audit. The **tcbck** command also deletes the specified stanzas from the **/etc/security/sysck.cfg** file.

In update mode, the **tcbck** command (**−l**) adds or modifies **/dev/** entry definitions in the **/etc/security/sysck.cfg** file for the specified **/dev** entry. This flag should be run by the administrator to add newly created devices that are trusted to the **sysck.cfg** file. If new devices are not added to the **sysck.cfg** file the tree option produces warnings of unregistered devices.

The **−l** flag creates a stanza for each **/dev/** entry listed on the command line. The information for the stanza is taken from the current status of the **/dev** entry. The stanza includes:

Device name	**/dev/ entry name**
File type	Either FILE, DIRECTORY, FIFO, SYMLINK, BLK_DEV, CHAR_DEV, or MPX_DEV
Owner id	Owner name
Group id	Group name
Permissions	Read/write/execute permissions for owner, group and other. SUID, SGID, SVTX and TCB attribute bits
Target	If the file is a symbolic link, the target file will be listed.

File definitions to be added or modified with the **−a** flag can be specified on the command line or in a file as *Attribute*=*Value* statements. The following attributes can be used:

acl The access control list for the file. If the value is **blank**, the **acl** attribute is removed. If no value is specified, the command computes a value, according to the format described in Access Control Lists.

class The logical group of the file. A value must be specified, since it cannot be computed. If the value is **blank**, the **class** attribute is removed from the specified file stanza. The value is *ClassName* [*ClassName*].

checksum The checksum of the file. If the value is **blank**, the **checksum** attribute is removed. If no value is specified, the command computes a value, according to the format given in the **sum** command. The value is the output of the **sum −r** command, including spaces.

group The file group. If the value is **blank**, the **group** attribute is removed. If no value is specified, the command computes a value, which can be a group ID or a group name.

links The hard links to this file. If the value is **blank**, the **links** attribute is removed. A value must be specified, since it cannot be computed. The value must be an absolute path name, expressed as *Path* [,*Path* ...].

mode The File mode. If the value is **blank**, the **mode** attribute is removed. If no value is specified, the command computes a value, which can be an octal number or string (*rwx*), and have the **tcb**, **SUID**, **SGID**, and **SVTX** attributes.

owner The file owner. If the value is **blank**, the **owner** attribute is removed. If no value is specified, the

command computes a value, which can be a user ID or a user name.

program The associated checking program for the file. If the value is **blank**, the **program** attribute is removed. A value must be specified, since it cannot be computed. The value must be an absolute path name. If flags are specified, the value should be expressed as *Path* ,*Flag*.

symlinks The symbolic links to the file. If the value is **blank**, the **symlinks** attribute is removed. A value must be specified, since it cannot be computed. The value must be an absolute path name, expressed as *Path* [,*Path*..].

size The size of the file in bytes. If the value is **blank**, the **size** attribute is removed. If no value is specified, the command computes a value. The value is a decimal number.

source The source for the file. If the value is **blank**, the **source** attribute is removed. If no value is specified, an empty file of the appropriate type is created. The value must be an absolute path name.

type The type of file. This value cannot be **blank**. If no value is specified, the command computes a value, which can be the **FILE**, **DIRECTORY**, **FIFO**, **BLK_DEV**, **CHAR_DEV**, or **MPX_DEV** keywords.

You can add, delete, or modify the attributes of the **tcbck** command by creating or modifying a **sysck** stanza in the **/etc/security/sysck.cfg** file. The following attributes can be used:

checksum An alternate checksum command to compute the checksum value of files. The system appends the name of each file to the command. If the value is **blank**, this alternate **checksum** attribute is removed. The value is the command string to be run on each file. The default string is **/usr/bin/sum −r <**.

setgids An additional list of administrative groups to be checked for **setgid** programs that are not valid (groups with ID numbers above 200). If the value is **blank**, the **setgids** attribute is removed. The value is a comma separated list of group names.

setuids An additional list of administrative users to be checked for **setuid** programs that are not valid (users with ID numbers above 200). If the value is **blank**, the **setuids** attribute is removed. The value is a comma separated list of user names.

treeck_novfs A list of file systems to be excluded from verification by the **tcbck** command during a check of an installed file system tree. If the value is **blank**, the **treeck_novfs** attribute is removed. The value is a comma separated list of file systems.

treeck_nodir A list of directories to be excluded from verification by the **tcbck** command. If the value is blank, the **treeck_nodir** attribute is removed. The value is a comma separated list of directories. File systems that exist under directories contained in this attribute are NOT excluded. Use the −i flag to exclude these file systems.

Refer to the **/etc/security/sysck.cfg** file for more information about these attributes and to the "Examples" section for an example of a typical stanza.

If *Attributes* are included without values, the command tries to compute the value from the file to be changed. The **type** attribute is mandatory, but the others do not need to be specified.

Fixing Errors

To fix errors, the **tcbck** command usually resets the attribute to the defined value. For the following attributes, the command modifies its actions as described:

checksum Disables the file by clearing its access control list, but does not stop any further checks.

links Creates any missing hard links. If a link exists to another file, the link is deleted.

program Invokes the program, which must exist and have an absolute path name. A message is printed if an error occurs, but no additional action is taken.

size	Disables the file by clearing its access control list, but does not stop any further checks.
source	Copies the source file to the file identified by the *File* parameter. If the source is null, any existing file is deleted and a file of the correct **type** is created.
symlinks	Creates any missing symbolic links. If a link exists to another file, the link is deleted.
type	Disables the file by clearing its access control list, and stops any further checks.

If you used the **−t** flag with the **tcbck** command, you are prompted for a decision on fixing errors. If you answer `yes`, errors are fixed. If you give any other response, errors are not fixed.

Security

Access Control: This command should grant execute (x) access only to the root user and members of the security group. The command should be setuid to the root user and have the **trusted computing base** attribute.

Files Accessed:

Mode	File
r	**/etc/passwd**
r	**/etc/group**
r	**/etc/security/user**
rw	**/etc/security/sysck.cfg**
x	**/usr/bin/aclget**
x	**/usr/bin/aclput**
x	**/usr/bin/sum**

Auditing Events:

Event	Information
TCBCK_Check	file, error, status
TCBCK_Update	**file, function**

Examples

1. To add the `/bin/boo` file with `acl`, `checksum`, `class`, `group`, `owner`, and `program` attributes to the `tcbck` database, enter:

   ```
   tcbck −a /bin/boo acl checksum class=audit group owner\
   program=/bin/boock
   ```

   The resulting stanza will contain the attributes given above, with computed values inserted for those attributes you do not define. The database will contain a stanza like the following:

   ```
   /bin/boo:
          acl =
          checksum = 48235
          class = audit
          group = system
   ```

```
owner = root
program = /bin/boock
type = FILE
```

The attribute values are added to the installation definition but are not checked for correctness. The `program` attribute value comes from the command line, the `checksum` attribute value is computed with the **checksum** program, and all the others, except `acl`, are computed from the file i−node.

2. To indicate that the size of a file should be checked but not added to the database, since it can expand during installation, use the `VOLATILE` keyword, as in the following example for the `/etc/passwd` file:

```
/etc/passwd:
        type =  FILE
        owner = root
        group = system
        size  = 1234,VOLATILE
```

3. To delete the `/bin/boo` file definition from the `tcbck` database, enter:

```
tcbck −d /bin/boo
```

4. To delete all definitions with a `class` of `audit` from the `tcbck` database, enter:

```
tcbck −d audit
```

5. To check all the files in the `tcbck` database, and fix and report all errors, enter:

```
tcbck −y ALL
```

6. To exclude the `/calvin` and the `/hobbes` file systems from verification during a security audit of an installed file system tree, enter:

```
tcbck −a sysck treeck_novfs=/calvin,/hobbes
```

7. To exclude a directory from verification during a security audit, enter:

```
tcbck −a sysck treeck_nodir=/home/john
```

8. To add `jfh` and `jsl` as administrative users and `developers` as an administrative group to be verified during a security audit of an installed file, enter:

```
tcbck −a sysck setuids=jfh,jsl setgids=developers
```

9. To create/modify **sysck.cfg** stanza entries for the newly created /**dev** entries `foo` and `bar`, enter:

```
tcbck −l /dev/foo /dev/bar
```

> **Note:** By adding these entries you are registering them as part of the Trusted computing base.

Warning: Although the special characters "$" and "?" are allowed in this routine, using them in filenames may result in potential problems such as ambiguous files.

Files

/usr/bin/tcbck Specifies the path to the **tcbck** command.

/etc/security/sysck.cfg Specifies the path to the system configuration database.

Related Information

The **aclget** command, **grpck** command, **installp** command, **pwdck** command, **sum** command, **usrck** command.

The Software Vital Product Data (SWVPD) Overview in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

Access Control Lists in *AIX Version 4.3 System User's Guide: Operating System and Devices* discusses the format of an access control list and provides an example of one.

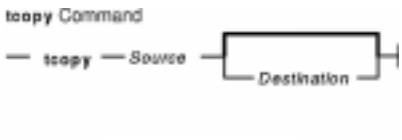
For more information about the identification and authentication of users, discretionary access control, the trusted computing base, and auditing, refer to Security Administration in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

tcopy Command

Purpose

Copies a magnetic tape.

Syntax



tcopy *Source* [*Destination*]

Description

The **tcopy** command copies magnetic tapes. Source and target file names are specified by the *Source* and *Destination* parameters. The **tcopy** command assumes that there are two tape marks at the end of the tape, and it ends when it finds the double file marks. With only a source tape specified, the **tcopy** command prints information about the size of records and tape files

Examples

To copy from one streaming tape to a 9−track tape, enter:

```
tcopy /dev/rmt0 /dev/rmt8
```

Files

/usr/bin/tcopy Contains the **tcopy** command.

Related Information

Backup Files and Storage Media Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

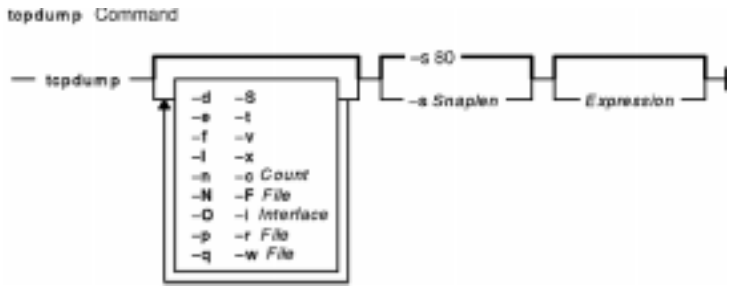
The **rmt** special file.

tcpdump Command

Purpose

Prints out packet headers.

Syntax



tcpdump [**−d**] [**−e**] [**−f**] [**−I**] [**−n**] [**−N**] [**−O**] [**−p**] [**−q**] [**−S**] [**−t**] [**−v**] [**−x**] [**−c** *Count*] [**−F** *File*] [**−i** *Interface*] [**−r** *File*] [**−s** *Snaplen*] [**−w** *File*] [*Expression*]

Description

The **tcpdump** command prints out the headers of packets captured on a network interface that matches the boolean *Expression* parameter. If no *Expression* parameter is given, all packets on the network will be dumped. Otherwise, only packets for which the *Expression* parameter is True will be dumped. Only Ethernet, Fiber Distributed Data Interface (FDDI), token−ring, and loopback interfaces are supported. Access is controlled by the permissions on */dev/bpfO*,*1*,*2*, and *3*.

The *Expression* parameter consists of one or more *primitives*. Primitives usually consist of an *id* (name or number) preceded by one or more qualifiers. There are three types of qualifier:

type Specifies what kind of device the *id* name or number refers to. Possible types are **host**, **net**, and **port**. Examples are `host foo`, `net 128.3`, `port 20`. If there is no type qualifier, **host** is assumed.

dir Specifies a particular transfer direction to or from *id*. Possible directions are **src**, **dst**, **src or dst**, and **srcand dst**. Some examples with *dir* qualifiers are: `src foo`, `dst net 128.3`, `src` or `dst port ftp−data`. If there is no *dir* qualifier, **src or dst** is assumed.

proto Restricts the match to a particular protocol. Possible *proto* qualifiers are: **ether**, **ip**, **arp**, **rarp**, **tcp**, and **udp**. Examples are: `ether src foo`, `arp net 128.3`, `tcp port 21`. If there is no *proto* qualifier, all protocols consistent with the type are assumed. For example, `src foo` means `ip` or `arp`, `net bar` means `ip` or `arp` or `rarp net bar`, and `port 53` means `tcp` or `udp port 53`.

In addition to the above, there are some special primitive keywords that do not follow the pattern: **broadcast**, **multicast**, **less**, **greater**, and arithmetic expressions. All of these keywords are described below.

Allowable Primitives

Primitives allowed are the following:

dst host*Host* True if the value of the IP (Internet Protocol) destination field of the packet is the same as the value of the *Host* variable, which may be either an address or a name.

src host*Host* True if the value of the IP source field of the packet is the same as the value of the *Host* variable.

host*Host* True if the value of either the IP source or destination of the packet is the same as the value of the *Host* variable. Any of the above **host** expressions can be prepended with the keywords **ip**, **arp**, or **rarp** as in:

ip host*Host*

If the *Host* variable is a name with multiple IP addresses, each address will be checked for a match.

dst net*Net* True if the value of the IP destination address of the packet has a network number of *Net*.

src net*Net* True if the value of the IP source address of the packet has a network number of *Net*.

net*Net* True if the value of either the IP source or destination address of the packet has a network number of *Net*.

dst port*Port* True if the packet is TCP/IP (Transmission Control Protocol/Internet Protocol) or IP/UDP (Internet Protocol/User Datagram Protocol) and has a destination port value of *Port*. The port can be a number or a name used in */etc/services*. If a name is used, both the port number and protocol are checked. If a number or ambiguous name is used, only the port number is checked (`dst port 513` will print both TCP/login traffic and UDP/who traffic, and `port domain` will print both TCP/domain and UDP/domain traffic).

src port*Port* True if the value of the *Port* variable is the same as the value of the source port.

port*Port* True if the value of either the source or the destination port of the packet is *Port*. Any of the above **port** expressions can be prepended with the keywords **tcp** or **udp**, as in:

tcp src port*port*

which matches only TCP packets.

less*Length* True if the packet has a length less than or equal to *Length*. This is equivalent to:

len < = *Length*.

greater*Length* True if the packet has a length greater than or equal to the *Length* variable. This is equivalent to:

len > = *Length*

ip proto*Protocol* True if the packet is an IP packet of protocol type *Protocol*. *Protocol* can be a number or one of the names **icmp**, **udp**, or **tcp**.

> **Note:** The identifiers **tcp**, **udp**, and **icmp** are also keywords and must be escaped via \ (backslash), which is \\ (double backslash) in the korn−shell.

ip broadcast True if the packet is an IP broadcast packet. It checks for the all−zeroes and all−ones broadcast conventions, and looks up the local subnet mask.

ip multicast True if the packet is an IP multicast packet.

proto*Protocol* True if the packet is of type *Protocol*. *Protocol* can be a number or a name like **ip**, **arp**, or **rarp**.

> **Note:** These identifiers are also keywords and must be escaped via \ (backslash).

ip, **arp**, **rarp** Abbreviations for:

proto*p*

where *p* is one of the above protocols.

tcp, **udp**, **icmp** Abbreviations for:

ip proto*p*

where *p* is one of the above protocols.

Relational Operators of the Expression Parameter

The simple relation:

expr replop expr

Holds true where *relop* is one of > (greater than), < (less than), >= (greater than or equal to), <= (less than or equal to), = (equal), != (exclamation point and equal sign) and *expr* is an arithmetic expression composed of integer constants (Expressed in standard C syntax), the normal binary operators + (plus sign), − (minus sign), * (asterisk), / (slash), & (ampersand), | (pipe), a length operator, and special packet data accessors. To access data inside the packet, use the following syntax:

*proto***[***expr***:***size* **]**

Proto is one of the keywords **ip**, **arp**, **rarp**, **tcp** or **icmp**, and indicates the protocol layer for the index operation. The byte offset relative to the indicated protocol layer is given by *expr*. The indicator *size* is optional and indicates the number of bytes in the field of interest; it can be either one, two, or four, and defaults to one byte. The length operator, indicated by the keyword **len**, gives the length of the packet.

For example, expression `ip[0] & 0xf != 5` catches only unfragmented datagrams and frag `0` of fragmented datagrams. This check is implicitly implied to the **tcp** and **udp** index operations. For instance, **tcp[0]** always means the first byte of the TCP header, and never means the first byte of an intervening fragment.

Combining Primitives

More complex filter expressions are built up by using the words **and**, **or**, and **not** to combine primitives. For example, `host foo and not port ftp and not port ftp-data`. To save typing, identical qualifier lists can be omitted. For example, `tcp dst port ftp or ftp-data or domain` is exactly the same as `tcp dst port ftp or tcp dst port ftp-data or tcp dst port domain`.

Primitives may be combined using a parenthesized group of primitives and operators (parentheses are special to the Shell and must be escaped).

- A
- Negation (`!' or `**not**').
- Concatenation (`**and**').
- Alternation (`**or**').

Negation has highest precedence. Alternation and concatenation have equal precedence and associate left to right.

If an identifier is given without a keyword, the most recent keyword is assumed. For example,

```
not host gil and devo
```

is short for

```
not host gil and host devo
```

which should not be confused with

```
not \(host gil or devo\)
```

Expression arguments can be passed to the **tcpdump** command as either a single argument or as multiple arguments, whichever is more convenient. Generally, if the expression contains Shell metacharacters, it is easier to pass it as a single, quoted argument. Multiple arguments are concatenated with spaces before being parsed.

Protocol Output Formats

The output of the **tcpdump** command is protocol−dependent. The following are brief descriptions and examples of most output formats.

TCP Packets

The general format of a TCP protocol line is:

```
src  >  dst: flags data−seqno ack win urg options
```

In the following list of fields, `src,dst` and `flags` are always present. The other fields depend on the contents of the packet's TCP protocol header and are output only if appropriate.

`src`	Indicates the source (host) address and port. The `src` field is always specified.
`dst`	Indicates the destination address and port. The `dst` field is always specified.
`flags`	Specifies some combination of the flags S (SYN), F (FIN), P (PUSH) or R (RST) or a single `.` (period) to indicate no flags. The `flags` field is always specified.
`data−seqno`	Describes the portion of sequence space covered by the data in this packet (see example below).
`ack`	Specifies (by acknowledgement) the sequence number of the next data expected from the other direction on this connection.
`win`	Specifies the number of bytes of receive buffer space available from the other direction on this connection.
`urg`	Indicates there is urgent data in the packet.
`options`	Specifies TCP options enclosed in angle brackets (for example, `<mss 1024>`).

Here is the opening portion of the **rlogin** command from host `gil` to host `devo`:

```
gil.1023 > devo.login:S 768512:768512(0) win 4096 <mss 1024>
devo.login > gil.1023:S 947648:947648(0) ack 768513 win 4096 <mss
1024>
gil.1023 > devo.login: . ack 1 win 4096
gil.1023 > devo.login: P 1:2(1) ack 1 win 4096
devo.login > gil.1023: ack 2 win 4096
gil.1023 > devo.login: P 2:21(19) ack 1 win 4096
devo.login > gil.1023: P 1:2(1) ack 21 win 4077
devo.login > gil.1023: P 2:3(1) ack 21 win 4077 urg 1
devo.login > gil.1023: P 3:4(1) ack 21 win 4077 urg 1
```

The first line says that TCP port 1023 on host `gil` sent a packet to the `login` port on host `devo`. The `S` indicates that the SYN flag was set. The packet sequence number was `768512` and it contained no data. (The notion is `first:last(nbytes)' which means `sequence numbers *first* up to but not including *last* which is

nbytes bytes of user data'.) There was no piggy−backed `ack` field, the available receive field `win` was `4096` bytes and there was a max−segment−size(mss) option requesting an `mss` of `1024` bytes.

Host `Devo` replies with a similar packet except it includes a piggy−backed `ack` field for host `gil`'s SYN. Host `gil` then acknowledges host `devo`'s SYN. The `.` (period) means no flags were set. The packet contains no data so there is no data sequence number.

> **Note:** The `ack` field sequence number is a small integer (1).

The first time a **tcpdump** sees a TCP conversation, it prints the sequence number from the packet. On subsequent packets of conversation, the difference between the current packet's sequence number and this initial sequence number is printed. This means that sequence numbers after the first can be interpreted as relative byte positions in the conversation's data stream (with the first data byte each direction being 1). The **−S** flag overrides this feature, causing the original sequence numbers to be output.

On the sixth line, host `gil` sends host devo `19` bytes of data (bytes 2 through 20 in the `gil−devo` side of the conversation). The PUSH flag is set in the packet. On the seventh line, host `devo` says it received data sent by host `gil` up to but not including byte `21`. Most of this data is apparently sitting in the socket buffer since host `devo`'s receive window has gotten 19 bytes smaller. Host `devo` also sends one byte of data to host `gil` in its packet. On the eighth and ninth lines, host `devo` sends two bytes of `urgentPUSH` data to host `gil`.

UDP Packets

UDP format is illustrated by this **rwho** command packet:

```
devo.who > bevo.who: udp 84
```

This command sequence says that port `who` on host `devo` sent a `udp` datagram to port `who` on host `bevo`. The packet contained `84` bytes of user data.

Some UDP services are recognized (from the source or destination port number) and the higher level protocol information is printed. In particular, Domain Name service requests (RFC−1034/1035) and Sun RPC calls (RFC−1050) to NFS.

UDP Name Server Requests

Name server requests are formatted as:

```
src > dst: id op? flags qtype qclass name (len)
```

In addition to those fields previously explained, UDP name server requests have the following:

`id` Specifies the identification number of the query.
`op` Specifies the type of operation. The default is the query operation.

`qclass`

`name`

`(len)`

An example of a name server rquest is:

```
tnegev.1538 > tnubia.domain: 3+ A? austin.ibm.com. (37)
```

Host `tnegev` asked the domain server on `tnubia` for an address record (`qtype`=A) associated with the name `austin.ibm.com`. The query id was `3`. The `+` (plus sign) indicates the recursion desired flag was set. The query length was `37` bytes, not including the UDP and IP protocol headers. The query operation was the normal one, Query, so the op field was omitted. If the op had been anything else, it would have been printed between the `3` and the `+` . Similarly, the qclass was the normal one (*C_IN*), and it was omitted. Any other qclass would have been printed immediately after the `A`.

A few anomalies are checked and may result in extra fields enclosed in square brackets. If a query contains an answer, name server, or authority section, then *ancount*, *nscount*, or *arcount* are printed as [*n*a], [*n*n] or [*n*au] where *n* is the appropriate count. If any of the response bits are set (AA, RA, or rcode) or any of the `must be zero' bits are set in bytes two and three [b2&3=*x*] is printed, where *x* is the hex value of header bytes two and three.

UDP Name Server Responses

Name server responses are formatted as:

```
src > dst: id op rcode flags a/n/au type class data (len)
```

In addition to those fields previously explained, UDP name server responses have the following:

`rcode`

`data`

An example of a name server response is:

```
tnubia.domain > tnegev.1538: 3 3/3/7 A 129.100.3
tnubia.domain > tnegev.1537: 2 NXDomain* 0/1/0 (97)
```

In the first example, `tnubia` responds to query `3` from `tnegev` with `3` answer records, `3` name server records, and `7` authority records. The first answer record is type `A` (address) and its data is internet address `129.100.100.3`. The total size of the response was 273 bytes, excluding UDP and IP headers. The op (Query) and response code (NoError) were omitted, as was the class (C_IN) of the `A` record.

In the second example, `tnubia` responds to query `2` with a response code of non−existent domain (`NXDomain`) and with `0` answer records, `1` name server record, and `0` authority records. The `*` (asterisk) indicates that the authoritative answer bit was set. Since there were no answers, no type, class, or data were printed.

Other flag characters that might appear are − (recursion available, RA, *not* set) and | (truncated message, TC, set).

> **Note:** Name server requests and responses tend to be large and the default snaplen of 80 bytes may not capture enough of the packet to print. Use the **−s** flag to increase the snaplen if you need to investigate large quantities of name server traffic.

NFS Requests

Sun NFS (Network FIle System) requests and replies are formatted as:

```
src.xid > dst.nfs: len op args
src.nfs > dst.xid: reply stat len
```

In addition to fields previously explained, NFS requests and responses include these fields:

`args`	Specifies the directory file$file handle$.
`reply stat`	Indicates the response status of the operation.

An example of an NFS request and response is:

```
L1.e2766 > L2.nfs: 136 readdir fh 6.5197 8192 bytes @ 0
L2.nfs > L1.e2766: reply ok 384
L1.e2767 > L2.nfs: 136 lookup fh 6.5197 `RCS'
```

In the first line, host `L1` sends a transaction with id `e2766` to `L2` (note that the number following the `src` host is a transaction id, not the source port). The request was `136` bytes, excluding the UDP and IP headers. The operation was a `readir` (read directory) on file handle (`fh`) `6.5197`. Starting at offset `0`, `8192` bytes are read. `L2` replies `ok` with `384` bytes of data.

In the third line, `L1` asks `L2` to lookup the name `` `RCS' `` in directory file `6.5197`. Note that the data printed depends on the operation type.

> **Note:** NFS requests are very large and the above won't be printed unless *snaplen* is increased. Use the flag **−s 192** to watch NFS traffic.

ARP/RARP Packets

Address Resolution Protocol/Reverse Address Resolution Protocol (ARP/RARP) output shows the type of request and its arguments. The following example shows the start of the **rlogin** command from host `devo` to host `bevo`:

```
arp who−has bevo tell devo
arp reply bevo is−at 1d:2d:3d:4d:5d:6d
```

The first line says that `devo` sent an ARP packet asking for the Ethernet address of Internet host `bevo`. In the second line `bevo` replies with its Ethernet address.

IP Fragmentation

Fragmented Internet datagrams are printed as:

```
(frag id:size@offset+)
(frag id:size@offset)
```

The first form indicates that there are more fragments. The second indicates this is the last fragment. IP fragments have the following fields:

`id`	Identifies the fragment.
`size`	Specifies the fragment size (in bytes) including the IP header.
`offset`	Specifies the fragment's offset (in bytes) in the original datagram.

The fragment information is output for each fragment. The first fragment contains the higher level protocol header and the frag info is printed after the protocol info. Fragments after the first contain no higher level protocol header and the frag info is printed after the source and destination addresses. For example here is a ping echo/reply sequence:

```
gil > devo: icmp: echo request (
```

```
frag 34111: 1480@0+)
gil > devo: (frag 34111!28@1480)
devo > gil: icmp: echo reply (frag
                   15314:148@0+)
```

A packet with the IP *don't fragment* flag is marked with a trailing **(DF)**.

Timestamps

By default, all output lines are preceded by a timestamp. The timestamp is the current clock time in the form

```
hh:mm:ss.frac
```

and is as accurate as the kernel's clock. The timestamp reflects the time the kernel first saw the packet. No attempt is made to account for the time lag between when the ethernet interface removed the packet from the wire and when the kernel serviced the new packet interrupt.

Flags

−c	Exits after receiving *Count* packets.
−d	Dumps the compiled packet−matching code to standard output, then stops.
−e	Prints the link−level header on each dump line. If the **−e** flag is specified, the link level header is printed out. On Ethernet and token−ring, the source and destination addresses, protocol, and packet length are printed.
−f	Prints foreign internet addresses numerically rather than symbolically.
−F	Uses *File* as input for the filter expression. The **−F** flag ignores any additional expression given on the command line.
−i	Listens on *Interface*. If unspecified, the **tcpdump** command searches the system interface list for the lowest numbered and configured interface that is up. This search excludes loopback interfaces.
−I	(Capital i) Specifies immediate packet capture mode. The **−l** flag does not wait for the buffer to fill up.

−l	(Lowercase L) Buffers the standard out (stdout) line. This flag is useful if you want to see the data while capturing it. For example:
	`tcpdump −l : tee dat`
	or
	`tcpdump −l > dat & tail −f dat`

−n	Omits conversion of addresses to names.
−N	Omits printing domain name qualification of host names. For example, the **−N** flag prints `gil` instead of `gil.austin.ibm.com`.
−O	Omits running the packet−matching code optimizer. This is useful only if you suspect a bug in the optimizer.
−p	Specifies that the interface not run in promiscuous mode. **Note:** The interface might be in promiscuous for some other reason; hence, **−p** cannot be used as an abbreviation for `ether host {localhost}' or `broadcast'.
−q	Quiets output. The **−q** flag prints less protocol information so output lines are shorter.
−r	Reads packets from *File* (which was created with the **−w** option).

	Standard input is used if *File* is "−".
−s	Captures *Snaplen* bytes of data from each packet rather than the default of 80. Eighty bytes is adequate for IP, ICMP, TCP, and UDP but may truncate protocol information from name server and NFS packets (see below). Packets truncated because of a limited snapshot are indicated in the output with "[|*proto*]", where *proto* is the name of the protocol level at which the truncation has occurred. **Note:** Taking larger snapshots increases the amount of time it takes to process packets thereby decreasing the amount of packet buffering. This may cause packets to be lost. You should limit *Snaplen* to the smallest number of bytes that capture the protocol information you are interested in.
−S	Prints absolute rather than relative TCP sequence numbers.
−t	Omits the printing of a timestamp on each dump line.
−tt	Prints an unformatted timestamp on each dump line.
−v	Specifies slightly more verbose output. For example, the time to live and the type of service information in an IP packet is printed.
−w	Writes the raw packets to *File* rather than parsing and printing them out. They can later be printed with the **−r** flag. Standard output is used if *File* is "−".
−x	Prints each packet (minus its link level header) in hex. The smaller of the entire packet or *Snaplen* bytes will be printed.

Examples

1. To print all packets arriving at or departing from `devo`:

   ```
   tcpdump host devo
   ```

2. To print traffic between `gil` and either `devo` or `bevo`:

   ```
   tcpdump ip host gil and \(devo bevo\)
   ```

3. To print all IP packets between `bevo` and any host except `gil`:

   ```
   tcpdump ip host bevo and bevo gil
   ```

4. To print all traffic between local hosts and hosts on network `192.100.192`:

   ```
   tcpdump net 192.100.192
   ```

5. To print traffic neither sourced from nor destined for local hosts:

   ```
   tcpdump ip and not net localnet
   ```

6. To print the start and end packets (the SYN and FIN packets) of each TCP conversation that involves a non−local host:

   ```
   tcpdump \(tcp[13] \& 3 !=0 and not src and dst net localnet\)
   ```

7. To print all ICMP packets that are not echo requests or replies (not ping packets):

   ```
   tcpdump \(icmp[0] !=8 and icmp[0] !=0\)
   ```

8. To immediately print packet information, enter

   ```
   tcpdump −I
   ```

9. To specify the token−ring interface to listen on, enter:

   ```
   tcpdump −i tr0
   ```

10. To print packet information to the file *TraceInfo*, enter:

    ```
    tcpdump −wTraceInfo
    ```

Related Information

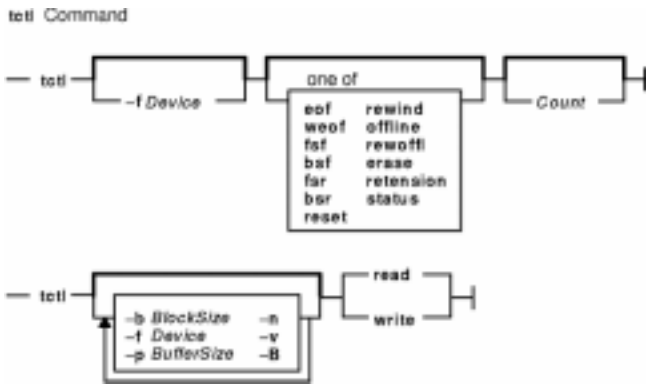
None

tctl Command

Purpose

Gives subcommands to a streaming tape device.

Syntax



tctl [**−f** *Device*] [**eof** | **weof** | **fsf** | **bsf** | **fsr** | **bsr** | **rewind** | **offline** | **rewoffl** | **erase** | **retension** | **reset** | **status**] [*Count*]

tctl [**−b** *BlockSize*] [**−f** *Device*] [**−p** *BufferSize*] [**−v**] [**−n**] [**−B**] { **read** | **write** }

Description

The **tctl** command gives subcommands to a streaming tape device. If you do not specify the *Device* variable with the **−f** flag, the **TAPE** environment variable is used. If the environment variable does not exist, the **tctl** command uses the **/dev/rmt0.1** device. (When the **tctl** command gives the **status** subcommand, the default device is **/dev/rmt0**.) The *Device* variable must specify a raw (not block) tape device. The *Count* parameter specifies the number of end−of−file markers, number of file marks, or number of records. If the *Count* parameter is not specified, the default count is 1.

Subcommands

eof or **weof**
: Writes the number of end−of−file markers specified by the *Count* parameter at the current position on the tape. On an 8 mm tape drive, an end−of−file marker can be written in three places:
- Before blank tape
- Before an extended file mark
- At the beginning−of−tape mark

On a 9−track tape drive, the end−of−tape marker can be written at any location on the tape. However, this subcommand does not support overwriting single blocks of data.

fsf
: Moves the tape forward the number of file marks specified by the *Count* parameter and positions it on the end−of−tape (EOT) side of the file mark.

bsf
: Moves the tape backward the number of file marks specified by the *Count* parameter and positions it on the beginning−of−tape (BOT) side of the file mark.

If the **bsf** subcommand moves the tape past the beginning, the tape rewinds, and the **tctl** command returns **EIO**.

fsr	Moves the tape forward the number of records specified by the *Count* parameter.
bsr	Moves the tape backwards the number of records specified by the *Count* parameter.
rewind	Rewinds the tape. The *Count* parameter is ignored.
offline or **rewoffl**	Rewinds the tape and takes the tape drive offline. This will unload the tape when appropriate. The tape must be re−inserted before the device can be used again.
erase	Erases all contents on the tape and rewinds it.
read	Reads from the specified tape device (using the specified block size) until the internal buffer is full, and then writes the data to standard output, continuing to read and write this way until an end−of−file (EOF) mark is reached.
reset	Sends a bus device reset (BDR) to the tape device. The BDR will only be sent if the device cannot be opened and is not busy.
retension	Moves the tape to the beginning, then to the end, and then back to the beginning of the tape. If you have excessive read errors during a restore operation, you should run the **retension** subcommand. If the tape has been exposed to environmental extremes, you should run the **retension** subcommand before writing to tape.
status	Prints status information about the specified tape device.
write	Opens the tape device, reads from standard input, and writes the data to the tape device.

Notes:

1. When you specify the **read** or **write** subcommand, the **tctl** command opens the tape device and sets up the tape block size as specified by the **−b** or **−n** flag. If neither flag is specified, the **tctl** command uses a default block size of 512 bytes.
2. The **−b**, **−n**, **−p**, and **−v** flags apply only when using the **read** and **write** subcommands.
3. The **−B** flag applies only when using the **read** subcommand.

Flags

−b *BlockSize*	Specifies, in bytes, the size of buffer used to read and write to the tape device, and also specifies, in the absence of the **−n** flag, the tape block size. If the block size is 0, variable−length blocks are used and the size of the tape buffer is 32,768. If the **−b** flag is not specified, the default block size and the size of the tape buffer is 512 bytes.
−B	Writes the contents of the buffer each time the tape is read. Set this flag when reading variable−length records that are not of a regular and consistent size.
−f *Device*	Specifies the tape device.
−p *BufferSize*	Specifies the size of the buffer to be used on standard input and standard output. The default buffer size is 32,768 bytes. The *BufferSize* value must be a multiple of the tape block size.
−v	Verbose. Prints the sizes of each read and write to standard error.
−n	Specifies variable−length records when reading or writing to tape with the **read** or **write** subcommand.

Exit Status

This command returns the following exit values:

0 Successful completion.

>0 An error occurred.

Examples

1. To rewind the rmt1 tape device, enter:

   ```
   tctl −f /dev/rmt1 rewind
   ```

2. To move forward two file marks on the default tape device, enter:

   ```
   tctl fsf 2
   ```

3. To write two end−of−file markers on the tape in /dev/rmt0.6, enter:

   ```
   tctl −f /dev/rmt0.6 weof 2
   ```

4. To read a tape device formatted in 80−byte blocks and put the result in a file, enter:

   ```
   tctl −b 80 read > file
   ```

5. To read variable−length records from a tape device formatted in 80−byte blocks and put the result in a file, enter:

   ```
   tctl −b 80 −n read > file
   ```

6. To write variable−length records to a tape device using a buffer size of 1024 byes, enter:

   ```
   cat file | tctl −b 1024 −n −f/dev/rmt1 write
   ```

7. To write to a tape device in 512−byte blocks and use a 5120−byte buffer for standard input, enter:

   ```
   cat file | tctl −v −f /dev/rmt1 −p 5120 −b 512 write
   ```

   Note: The only valid block sizes for quarter−inch (QIC) tape drives are 0 and 512.

8. To write over one of several backups on an 8 mm tape, position the tape at the start of the backup file and issue these commands:

   ```
   tctl bsf 1
   ```

   ```
   tctl eof 1
   ```

The first command moves the tape to the beginning−of−tape side of the file mark. The second command rewrites the file mark, because writing is allowed before extended file marks. The erase head of the drive erases data before the write head reaches it, so the **write** subroutines can write over data already in the tape. However, all old data following is lost because its file markers are meaningless.

> **Note:** The **write** subroutines cannot write over a short file mark unless blank tape follows the short file mark. To write over existing data, as in the case of this example, the tape must be written with extended file marks (as specified through the SMIT interface).

Files

/dev/rmt*n* Specifies the raw streaming tape interface.
/usr/bin/tctl Contains the **tctl** command.

Related Information

The **dd** command, **mt** command.

The **environment** file, **rmt** special file.

The **ioctl** subroutine.

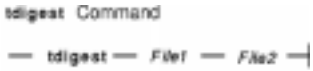
Backup Files and Storage Media Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

tdigest Command

Purpose

Converts the **terms** files.

Syntax



tdigest*File1File2*

Description

The **tdigest** command converts a structured file into a binary file. The conversion makes the entries readable by the INed editor.

The file designated by the *File1* parameter must be a structured file that contains terminal descriptions for the editor. The **/usr/lib/INed/def.trm** file is an example of this type of file. It contains a list of the terminal types supported by the INed editor. It also contains information about reading input from and writing output to various terminals. If two or more terminals have different terminal types, but use the same terminal descriptions, both of the terminal types are listed in the `Terminal Type` field.

To see the settings for the input and output sequences for a terminal description, go to the **/usr/lib/INed/def.trm** file, move the cursor to `Output` or `Input,` and press the F11 key (the **Zoom In** function). For information about keyboard layouts, press the F1 key (the **Help** function) within the INed editor. The `Input` list contains all of the INed functions and the corresponding key escape sequences. The `Output` list contains the values used to perform display functions such as cursor movement and clear screen.

If you want to use the editor on a terminal not listed in the **def.trm** file, you can add it to the **def.trm** file and run the **tdigest** command. The editor looks for the terminal description file in the **/usr/lib/INed/terms.bin** file.

Files

/usr/lib/INed/def.trm	Contains default terminal descriptions for the INed **termcap** editor. The **def.trm** file is a structured file.
/usr/lib/INed/terms.bin	Contains converted terminal descriptions. The **terms.bin** file is a binary file.

Related Information

The **e** command.

The **lft.h** file.

The **INed** files.

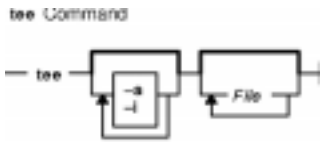
INed Editor Overview in *AIX Version 4.3 INed Editor User's Guide*.

tee Command

Purpose

Displays the output of a program and copies it into a file.

Syntax



tee [**−a**] [**−i**] [*File* ...]

Description

The **tee** command reads standard input, then writes the output of a program to standard output and simultaneously copies it into the specified file or files.

Flags

−a Adds the output to the end of *File* instead of writing over it.

−i Ignores interrupts.

Exit Status

This command returns the following exit values:

0 The standard input was successfully copied to all output files.

>0 An error occurred.

> **Note:** If a write to any successfully opened *File* operand is not successful, writes to other successfully opened *File* operands and standard output will continue, but the exit value will be **>0**.

Examples

1. To view and save the output from a command at the same time:

   ```
   lint program.c | tee program.lint
   ```

   This displays the standard output of the command **lint program.c** at the workstation, and at the same time saves a copy of it in the file `program.lint`. If a file named `program.lint` already exists, it is deleted and replaced.

2. To view and save the output from a command to an existing file:

   ```
   lint program.c | tee −a program.lint
   ```

This displays the standard output of the **lint program.c** command at the workstation and at the same time appends a copy of it to the end of the `program.lint` file. If the `program.lint` file does not exist, it is

created.

Files

/usr/bin/tee Contains the **tee** command.

Related Information

The **script** command.

Input and Output Redirection Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* describes how the operating system processes input and output and how to use the redirect and pipe symbols.

telinit or init Command

Purpose

Initializes and controls processes.

Syntax

{ **telinit** | **init** } { **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **a** | **b** | **c** | **Q** | **q** | **S** | **s** | **M** | **m** | **N** }

Description

The **init** command initializes and controls processes. Its primary role is to start processes based on records read from the **/etc/inittab** file. The **/etc/inittab** file usually requests that the **init** command run the **getty** command for each line on which a user can log in. The **init** command controls autonomous processes required by the system.

The process that constitutes the majority of the **init** command's process dispatching activities is **/usr/sbin/getty**. The **/usr/sbin/getty** process initiates individual terminal lines. Other processes typically dispatched by the **init** command are daemons and the shell.

The **telinit** command, which is linked to the **init** command, directs the actions of the **init** command. The **telinit** command takes a one−character argument and signals the **init** command by way of the **kill** subroutine to perform the appropriate action.

The **telinit** command sets the system at a specific run level. A run level is a software configuration that allows only a selected group of processes to exist. The system can be at one of the following run levels:

0−9 Tells the **init** command to place the system in one of the run levels **0−9**. When the **init** command requests a change to run levels **0−9**, it kills all processes at the current run levels and then restarts any processes associated with the new run levels.

0−1 Reserved for the future use of the operating system.

2 Contains all of the terminal processes and daemons that are run in the multiuser environment. In the multiuser environment, the **/etc/inittab** file is set up so that the **init** command creates a process for each terminal on the system. The console device driver is also set to run at all run levels so the system can be operated with only the console active.

3−9 Can be defined according to the user's preferences.

S,s,M,m Tells the **init** command to enter the maintenance mode. When the system enters maintenance mode from another run level, only the system console is used as the terminal.

The following arguments also serve as directives to the **init** command:

a,b,c Tells the **init** command to process only those records in the **/etc/inittab** file with **a**, **b**, or **c** in the run level field. These three arguements, **a**, **b**, and **c**, are not true run levels. They differ from run levels in that the **init** command cannot request the entire system to enter run levels **a, b,** or **c**.

When the **init** command finds a record in the **/etc/inittab** file with a value of **a**, **b**, or **c** in the run level field, it starts the process. However, it does not kill any processes at the current run level; processes with a value of **a**, **b**, or **c** in the run level field are started in addition to the processes already running at the current system run level. Another difference between true run levels and **a**, **b**, or **c** is that processes started with **a**, **b**, or **c** are not stopped when the **init** command changes run levels. Three ways stop **a**, **b**, or **c** processes:

- Type **off** in the *Action* field.
- Delete the objects entirely.
- Use the **init** command to enter maintenance state.

Q,q Tells the **init** command to re−examine the **/etc/inittab** file.

N Sends a signal that stops processes from being respawned.

During system startup, after the root file system has been mounted in the pre−initialization process, the following sequence of events occurs:

1. The **init** command is run as the last step of the startup process.
2. The **init** command attempts to read the **/etc/inittab** file.
3. If the **/etc/inittab** file exists, the **init** command attempts to locate an `initdefault` entry in the **/etc/inittab** file.
    a. If the `initdefault` entry exists, the **init** command uses the specified run level as the initial system run level.
    b. If the `initdefault` entry does not exist, the **init** command requests that the user enter a run level from the system console (**/dev/console**).
    c. If the user enters an **S**, **s**, **M** or **m** run level, the **init** command enters maintenance run level. These are the only run levels that do not require a properly formatted **/etc/inittab** file.
4. If the **/etc/inittab** file does not exist, the **init** command places the system in the maintenance run level by default.
5. The **init** command rereads the **/etc/inittab** file every 60 seconds. If the **/etc/inittab** file has changed since the last time the **init** command read it, the new commands in the **/etc/inittab** file are executed during system startup.

When you request the **init** command to change the run level, the **init** command reads the **/etc/inittab** file to identify what processes should exist at the new run level. Then, the **init** command cancels all processes that should not be running at the new level and starts any processes that should be running at the new level.

The processes run by the **init** command for each of these run levels are defined in the **/etc/inittab** file. The run level is changed by having a root user run the **telinit** command, which is linked to the **init** command. This user−run **init** command sends appropriate signals to the original **init** command initiated by the system during startup. The default run level can be changed by modifying the run level for the `initdefault` entry in the **/etc/inittab** file.

In the maintenance run level, the **/dev/console** console terminal is opened for reading and writing. The password for root is prompted. When the root password is entered successfully, the **su** command is invoked. Two ways exist to exit from the maintenance run level:

- If the shell is terminated, the **init** command requests a new run level.

  OR

- The **init** (or **telinit**) command can signal the **init** command and force it to change the run level of the system.

During a system startup attempt, apparent failure of the **init** command to prompt for a new run level (when **initdefault** is maintenance) may be due to the fact that the terminal console device (**/dev/console**) has been

switched to a device other than the physical console. If this occurs and you wish to work at the physical console rather than the **/dev/console**, you can force the **init** command to switch to the physical console by pressing the DEL (delete) key at the physical console device.

When the **init** command prompts for a new run level, enter one of the digits **0** through **9** or any of the letters **S**, **s**, **M**, or **m**. If you enter **S**, **s**, **M**, or **m**, the **init** command operates in maintenance mode with the additional result that if control had previously been forced to switch to the physical console, the **/dev/console** file is switched to this device as well. The **init** command generates a message to this effect on the device to which the **/dev/console** file was previously connected.

If you enter a **0** through **9** run level, the **init** command enters the corresponding run level. The **init** command rejects any other input and re−prompts you for the correct input. If this is the first time the **init** command enters any run level other than maintenance, it searches the **/etc/inittab** file for entries with the **boot** or **bootwait** keywords. If the **init** command finds these keywords, it performs the corresponding task, provided the run level entered matches that of the entry. For example, if the **init** command finds the **boot** keyword, it boots the machine. Any special initialization of the system, such as checking and mounting file systems, takes place before any users are allowed on the system. The **init** command then scans the **/etc/inittab** file to find all entries that are processes for that level. It then resumes normal processing of the **/etc/inittab** file.

Run level **2** is defined by default to contain all of the terminal processes and daemons that are run in the multiuser environment. In the multiuser environment, the **/etc/inittab** file is set up so that the **init** command creates a process for each terminal on the system.

For terminal processes, the shell terminates either as a result of an end of file character (EOF) typed explicitly or as the result of disconnection. When the **init** command receives a signal telling it that a process has terminated, it records the fact and the reason it stopped in **/etc/utmp** file and /**var**/**adm**/**wtmp** file. The /**var**/**adm**/**wtmp** file keeps a history of the processes started.

To start each process in the **/etc/inittab** file, the **init** command waits for one of its descendant processes to stop, for a power fail signal **SIGPWR**, or until the **init** command is signaled by the **init** or **telinit** commands to change the system's run level. When one of the above three conditions occurs, the **init** command re−examines the **/etc/inittab** file. Even if new entries have been added to the **/etc/inittab** file, the **init** command still waits for one of the three conditions to occur. To provide for instantaneous response, re−examine the **/etc/inittab** file by running the **telinit −q** command.

If the **init** command finds that it is continuously running an entry in the **/etc/inittab** file (more than five times in 225 seconds), it assumes that an error in the entry command string exists. It then prints an error message to the console and logs an error in the system error log. After the message is sent, the entry does not run for 60 seconds. If the error continues to occur, the command will respawn the entry only five times every 240 seconds. The **init** command continues to assume an error occurred until the command does not respond five times in the interval, or until it receives a signal from a user. The **init** command logs an error for only the first occurrence of the error.

When the **init** command is requested to change run levels by the **telinit** command, the **init** command sends a **SIGTERM** signal to all processes that are undefined in the current run level. The **init** command waits 20 seconds before stopping these processes with the **SIGKILL** signal.

If the **init** command receives a **SIGPWR** signal and is not in maintenance mode, it scans the **/etc/inittab** file for special power fail entries. The **init** command invokes the tasks associated with these entries (if the run levels permit) before any further processing takes place. In this way, the **init** command can perform cleanup and recording functions whenever the system experiences a power failure. It is important to note that these power fail entries should not use devices that need to be initialized first.

Environments

Because the **init** command is the ultimate ancestor of every process on the system, every other process on the system inherits the **init** command's environment variables. As part of its initialization sequence, the **init** command reads the **/etc/environment** file and copies any assignments found in that file into the environment passed to all of its subprocesses. Because **init** subprocesses do not run from within a login session, they do not inherit a umask setting from **init**. These processes may set the umask to whatever value they require. A command that is executed by **init** from the **/etc/inittab** file uses **init**'s ulimit values and not the default values as given in **/etc/security/limits**. The result is that a command that is successfully executed from the command line may not execute correctly when invoked by **init**. Any command that has specific **ulimit** requirements should include specific actions to set the **ulimit** values as required.

Examples

1. To request the **init** command to reexamine the **/etc/inittab** file, enter:

   ```
   telinit q
   ```

2. To request the **init** command to enter maintenance mode, enter:

   ```
   telinit s
   ```

Files

/etc/inittab	Specifies the **init** command control file.
/etc/utmp	Specifies the record of logged−in users.
/var/adm/wtmp	Specifies the permanent login accounting file.
/sbin/rc.boot	Specifies the pre−initialization command file.
/etc/rc	Specifies the initialization command file.
/etc/environment	Specifies system environment variables.
/dev/console	Specifies the console device driver.

Related Information

The **chitab** command, **lsitab** command, **mkitab** command, **rmitab** command, **getty** command, **rc** command.

The **inittab** file, the **rc.boot** file.

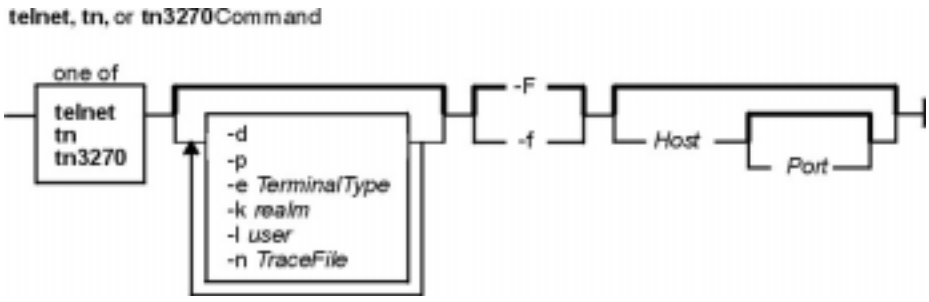
The **reboot** subroutine, **umask** subroutine, **ulimit** subroutine, **kill** subroutine.

telnet, tn,or tn3270 Command

Purpose

Connects the local host with a remote host, using the Telnet interface.

Syntax



{ **telnet** | **tn** | **tn3270** } [**−d**] [**−p**] [**−n***TraceFile*] [**−e***TerminalType*] [**−f** | **−F**] [**−k***realm*] [**−l** user] [*Host* [*Port*]]

Description

The **telnet** command, which is also referred to as the **tn** or **tn3270** command, operates in two different modes: command mode and input mode.

Command Mode

When the **telnet** command is issued without arguments, it enters command mode, as indicated by the `telnet>`, `tn>`, or the `tn3270>` prompt. A user can also enter command mode from input mode by pressing Ctrl−] for the **telnet** command, Ctrl−T for the **tn** command, or Ctrl−C for the **tn3270** command. In command mode, subcommands can be entered to manage the remote system. Some of these subcommands return you to the remote session upon completion. For those subcommands that do not, pressing the Enter key returns you to the remote session.

> **Note:** The default escape sequence for this command is Ctrl−] for the **telnet** command, Ctrl−T for the **tn** command, or Ctrl−C for the **tn3270** command. This default can be overridden by changing the **TNESC** environment variable.

To enter **telnet** command mode while connected to a remote host, type the Telnet escape key sequence. When in command mode, the standard operating system editing conventions, such as backspace, are available.

Input Mode

When the **telnet** command is issued with arguments, it performs an **open** subcommand with those arguments and then enters input mode. The type of input mode is either character−at−a−time or line−by−line, depending on what the remote system supports. In character−at−a−time mode, most text that is typed is immediately sent to the remote host for processing. In line−by−line mode, all text is echoed locally and completed lines are sent to the remote host.

In either input mode, if the **toggle localchars** subcommand has a value of True, the user's QUIT, INTR, and FLUSH characters are trapped locally and sent as Telnet Protocol sequences to the remote host. The **toggle**

autoflush and **toggle autosynch** subcommands cause this action to flush subsequent output to the terminal until the remote host acknowledges the Telnet sequence and to flush previous terminal input (in the case of QUIT and INTR characters).

Arabic/Hebrew Support

The **telnet**, **tn**, and **tn3270** command supports the Arabic and Hebrew texts, allowing the user to type Arabic or Hebrew characters while in an emulation session. The **Ar_AA** locale displays the Arabic characters in their correct shapes. The following functions support the bidirectional Arabic and Hebrew texts:

Language Selection

This function allows you to toggle the language layer. Activate the Arabic/Hebrew language selection with the following key combinations:

Alt+N	From an AIX terminal
Esc+N	From an ASCII terminal
Alt+N or **Esc+N**	From a Latin AIX terminal

Activate the Latin language layer with the following key combinations:

Alt+L	From an Arabic or Hebrew AIX terminal
Esc+L	From an ASCII terminal
Alt+L or **Esc+L**	From an AIX terminal

Screen Reverse

This function reverses the screen image and invokes the default language of the new screen orientation. Thus, if the screen is reversed to right−to−left, the language is changed to Arabic/Hebrew. If the screen is reversed to left−to−right, the language is changed to Latin.

If symmetric character swapping is enabled, reversing the screen causes bidirectional characters to be replaced by their counterparts. For example, if numeric character swapping is enabled, reversing the screen causes Hindi numerals to be replaced by their Arabic counterparts and the Arabic numerals to be replaced by their Hindi counterparts.

Activate screen reverse with the following key combinations:

Alt+S	From an Arabic or Hebrew AIX terminal
Esc+S	From an ASCII terminal
Alt+S or **Esc+S**	From a Latin AIX terminal

Push/End Push

The Push function allows you to edit text whose direction is opposite the screen orientation. When you activate this function, the cursor orientation is reversed, the language layer is changed accordingly, and a Push segment is created.

The Push function has two secondary modes:

Boundary Mode	This mode is activated upon entering the Push mode. In this mode, the cursor remains in its position while you type additional characters. The text is pushed in the opposite direction of the screen orientation.
Edit Mode	This mode is activated when the cursor is moved from its boundary position into the Push

segment area. In this mode, you can edit the text within the Push segment, while typing in the field's natural direction.

Activate this function with the following key combinations:

Alt+P	From an Arabic or Hebrew AIX terminal
Esc+P	From an ASCII terminal
Alt+P or **Esc+P**	From a Latin AIX terminal

The End Push function terminates the Push function. The cursor jumps to the end of the Push segment and its direction changes to the original direction. You can activate End Push by pressing any field exit keys such as cursor up, cursor down, or any attention identifier (AID) key such as the Enter key. You can also activate this function with the following key combinations:

Alt+E	From an Arabic or Hebrew AIX terminal
Esc+E	From an ASCII terminal
Alt+E or **Esc+E**	From a Latin AIX terminal

Field Reverse

This function toggles the field orientation to either the opposite of or the same as the screen orientation. This function does not invert the text in the field. The cursor orientation is set to the new field orientation and the language layer is selected accordingly.

For example, if the cursor is in the first logical position of a field or line when you activate the field reverse function, the cursor skips to the opposite side of that field or line. This position is now the first logical position. If the cursor is not in the first position of the field or line when you activate field reverse function, the cursor remains in its position and allows natural and correct editing of the existing text. Activate this function with the following key combinations:

Alt+R	From an Arabic or Hebrew AIX terminal
Esc+R	From an ASCII terminal
Alt+R or **Esc+R**	From a Latin AIX terminal

Autopush

This function assists you in typing mixed left−to−right and right−to−left text. When enabled, reversed segments are automatically initiated and terminated according to the typed characters or the selected language layer. Thus, this mode automatically invokes the Push mode and relieves you of invoking the Push function.

When you type a digit or Latin character in a right−to−left field, the Autopush function automatically initiates the Push function without changing the language. If you type additional digits or Latin character, the Push function continues; otherwise, the Push function automatically terminates. Thus, you can type Arabic/Hebrew text with embedded digits or Latin characters without invoking the Push/End Push functions.

When you type an Arabic/Hebrew character in a left−to−right field, the Autopush function automatically initiates the Push function without a language change. If you then type a digit or Latin character, the Autopush function automatically terminates. Thus, you can type Latin text with embedded Arabic/Hebrew text using the Language Selection function rather than the Push/End Push functions.

Activate this function with the following key combinations:

Alt+A	From an Arabic or Hebrew AIX terminal

Esc+A	From an ASCII terminal
Alt+A or **Esc+A**	From a Latin AIX terminal

Field Shape

This function shapes the Arabic characters in the current field or line. Activate this function with the following key combinations:

Alt+H	From an Arabic AIX terminal
Esc+H	From an ASCII terminal
Alt+H or **Esc+H**	From a Latin AIX terminal

Field Deshape

This function deshapes Arabic text in the current field or line. Activate this function with the following key combinations:

Alt+B	From an Arabic AIX terminal
Esc+B	From an ASCII terminal
Alt+B or **Esc+B**	From a Latin AIX terminal

Contextual Shape Determination

This function determines the shape of an Arabic character based on the surrounding text. Use the Contextual Shape Determination function only when typing or editing right−to−left text. This function is terminated when any of the specific shape selection keys is pressed. This is the default function. Activate this function with the following key combinations:

Alt+C	From an Arabic AIX terminal
Esc+C	From an ASCII terminal
Alt+C or **Esc+C**	From a Latin AIX terminal

Initial Shape Determination

This function shapes Arabic characters in their initial shapes. Activate this function with the following key combinations:

Alt+I	From an Arabic AIX terminal
Esc+I	From an ASCII terminal
Alt+I or **Esc+I**	From a Latin AIX terminal

Middle Shape Determination

This function shapes Arabic characters in their middle shapes. Activate this function with the following key combinations:

Alt+M	From an Arabic AIX terminal
Esc+M	From an ASCII terminal
Alt+M or **Esc+M**	From a Latin AIX terminal

Isolated Shape Determination

This function shapes Arabic characters in their isolated shapes. Activate this function with the following key combinations:

Alt+O	From an Arabic AIX terminal
Esc+O	From an ASCII terminal
Alt+O or **Esc+O**	From a Latin AIX terminal

Final Shape Determination

This function shapes Arabic characters in their final shapes. Activate this function with the following key combinations:

Alt+Y	From an Arabic AIX terminal
Esc+Y	From an ASCII terminal
Alt+Y or **Esc+Y**	From a AIX terminal

Miscellaneous Functions

To activate numeric swapping, type the following line at the command line:

```
export ARB_NUM_SWAP=1
```

To activate symmetric swapping, that is, to swap bidirectional characters such as braces, brackets, and so on, type the following line at the command line:

```
export ARB_SYM_SWAP=1
```

To specify the code page that the host uses, type the following line at the command line:

```
export RM_HOST_LANG=IBM-420
```

Terminal Type Negotiation

The **telnet** command negotiates the terminal type, using the Telnet protocol, and it sets the **TERM** environment variable according to what has been negotiated.

To override the terminal negotiation from the console, use the **EMULATE** environment variable or the **-e** flag; or invoke the tn3270 command if you require 3270 emulation. To determine whether terminal-type negotiation is performed, the following list describes the order of the **telnet** command processing:

1. The **-e** command-line flag. (No negotiation.)
2. The **EMULATE** environment variable. (No negotiation.)
3. The **tn3270** command. (No negotiation.)
4. If steps 1, 2, and 3 are not present, terminal-type negotiation occurs automatically.

If the client and the server negotiate to use a 3270 data stream, the keyboard mapping is determined by the following precedence:

$HOME/.3270keys	Specifies the user's 3270 keyboard mapping when the **tn** or **telnet** command is invoked. If you are using a color display, you can also change this file to customize the colors for 3270 displays.

/etc/map3270	Specifies the user's 3270 keyboard mapping when the **tn3270** command is invoked. The **/etc/map3270** file defines keyboard mapping and colors for the **tn3270** command.
/etc/3270.keys	Specifies the base 3270 keyboard mapping for use with limited function terminals.

Secure Attention Key (SAK) Option

In addition to terminal negotiation, the **telnet** command allows negotiation for the Secure Attention Key (SAK) option. This option, when supported, provides the local user with a secure communication path to the remote host for tasks such as changing user IDs or passwords. If the remote host supports the **SAK** function, a trusted shell is opened on the remote host when the **telnetsend sak** subcommand is issued. The **SAK** function can also be assigned to a single key available in **telnet** input mode, using the **set sak** subcommand.

End−of−Line Convention

The Telnet protocol defines the carriage−return line−feed (CR−LF) sequence to mean "end−of−line." For terminal input, this corresponds to a command−completion or end−of−line key being pressed on a user terminal. On an ASCII terminal, this is the CR key, but it may also be labeled "Return" or "Enter."

When a Telnet server receives the Telnet end−of−line sequence, CR−LF, as input from a remote terminal, the effect is the same as if the user had pressed the end−of−line key on a local terminal.

On ASCII servers, receiving the Telnet sequence CR−LF causes the same effect as a local user pressing the CR key on a local terminal. CR−LF and CR−NUL have the same effect on an ASCII server when received as input over a Telnet connection.

> **Note:** A Telnet user must be able to send CR−LF, CR−NULL, or LF. An ASCII user must be able to send CR−LF or CR−NULL.

A Telnet user on an ASCII host should have a user−controllable mode to send either CR−LF or CR−NULL when the user presses the end−of−line key. The CR−LF should be the default. The Telnet end−of−line sequence, CR−LF, must be used to send Telnet data that is not terminal−to−computer. This occurs, for example, when a Telnet server sends output or when the Telnet protocol incorporates another application protocol.

The **telnet** command "execs" (using the **exec** command) the **/usr/sbin/login** command to validate a user. This 1) allows all user and device attributes to take effect on telnet connections and 2) causes telnet connections to count against the maximum number of login sessions allowable at a time (determined by the **maxlogins** attribute). Attributes are defined in the **/etc/security/user** and **/etc/security/login.cfg** files.

Restrictions

- Earlier versions of the **telnet** command are not compatible with AIX Version 4 of the **telnet** command in sending escapes that emulate a high function terminal (HFT). AIX Version 4 of the **telnet** command sends only one escape when the escape key is hit, while prior versions send two escape characters. Therefore, when the **telnet** command is used to connect with an AIX Version 4 machine from a version 3.1 or earlier machine or earlier, the escape key produces two escape characters on the AIX Version 4 machine. This is fixed in a version 3.1 update.
- The **telnet** command must allow transmission of 8−bit characters that are not in binary mode to implement ISO 8859 Latin code page. This is necessary for internationalization of the TCP/IP commands.
- In order to support new character sets, the following was added to the `hft−m`, `ibm5081`, `hft`, `hft−nam`, `hft−c`, `aixterm−m`, and `aixterm` entries in the **terminfo** file:

```
box1=\154\161\153\170\152\155\167\165\166\164\156,     batt1=f1,
box2=\154\161\153\170\152\155\167\165\166\164\156,     batt2=f1md,
```

```
font0=\E(B,     font1=\E(0,
```

The `font0` and `font1` entries are *not* understood by version 3.1 terminals. If you work on a version 3.1 machine, use either `hft-m-old`, `hft-old`, `hft-nam-old`, `hft-c-old`, `aixterm-m-old`, or `aixterm-old` terminfo entries.

A common problem with the new **terminfo** file entries is that version 3.1 machines display a `B` character before each typed character when using the **telnet** command to connect to a VM system. If you see this, disconnect the telnet session and set the **TERM** environment variable to its corresponding `-old` terminal type and reconnect. The `B` character comes from the `font0=\E(B` entry. The escape sequence is not understood by version 3.1 and therefore the `B` is displayed on the screen.

- The **rlogind** and **telnetd** daemons use POSIX line discipline to change the line discipline on the local tty. If POSIX line discipline is not used on the local tty, echoing other line disciplines may result in improper behavior. AIX TCP/IP must have POSIX line discipline to function properly.
- The mouse cannot be used as an input device with the **telnet** command.
- The **telnet** command does not support the APL data stream.

Environment Variables

The following environment variables can be used with the **telnet** command:

EMULATE Overrides terminal−type negotiation in the same way as the **−e** flag. If the value of the **EMULATE** environment variable is defined as **vt100** or **3270**, the **telnet** command emulates a DEC VT100 terminal or 3270 terminal, respectively. If the **EMULATE** variable is not defined or has a value of **none**, the **telnet** command operates normally. If the **EMULATE** variable is set to **vt100** or **3270**, the **TERM** environment variable in the remote login connection should be set to the same value. You can check this by using the **env** command after the connection is open.

TNESC Specifies an alternate TELNET escape character, other than the default, Ctrl−] for the **telnet** command, Ctrl−T for the **tn** command, or Ctrl−C for the **tn3270** command. To change the **telnet** escape sequence, set **TNESC** to the octal value of the character you want to use. Then export **TNESC**. For example, set **TNESC** to 35 to change the TELNET escape sequence to Ctrl−].

MAP3270 Specifies an alternate file that contains the user's 3270 keyboard mapping. The **MAP3270** variable must contain the full path name to the alternate file. Create the alternate file using the same format as the default **/etc/map3270** file.

RM_HOST_LANG Specifies the EBCDIC code page being used on the remote 3270 host. Set the **RM_HOST_LANG** environment variable to the correct code page before you telnet (using the **telnet** command) to a non−English−speaking 3270 host. The default is English. Refer to the List of Converters in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs* for possible code pages to use. Format the **RM_HOST_LANG** environment variable by specifying the desired code page.

The **telnet** command converts characters by using the **iconv** command. Users can change the default conversion tables by using the **genxlt** command.

Flags

−d Turns debugging mode on.

−e *TerminalType* Overrides terminal−type negotiation. Possible values are **vt100**, **3270**, or **none**.

−n *TraceFile* Records network trace information in the file specified by the *TraceFile* variable.

−p Preserves current tty attributes.

−f	Causes the credentials to be forwarded. This flag will be ignored if Kerberos 5 is not the current authentication method. Authentication will fail if the current DCE credentials are not marked forwardable.
−F	Causes the credentials to be forwarded. In addition, the credentials on the remote system will be marked forwardable (allowing them to be passed to another remote system). This flag will be ignored if Kerberos 5 is not the current authentication method. Authentication will fail if the current DCE credentials are not marked forwardable.
−k*realm*	Allows the user to specify the realm of the remote station if it is different from the local systems realm. For these purposes, a *realm* is synonymous with a DCE cell. This flag will be ignored if Kerberos 5 is not the current authentication method.
−l*user*	Specifies the remote user the telnet wants to login as. This option is ignored if Kerberos 5 is not the current authentication method.

Subcommands

Before entering each subcommand, press the escape key sequence. The escape sequence tells the program that non−text information follows. Otherwise, the program interprets subcommands as text.

For each of the subcommands in the following list, you only need to type enough letters to uniquely identify the subcommand. (For example, **q** is sufficient for the **quit** subcommand.) This is also true for the arguments to the **display**, **emulate**, **mode**, **set**, and **toggle** subcommands.

The **telnet** subcommands are:

? [*Subcommand*]	Requests help on **telnet** subcommands. Without arguments, the **?** subcommand prints a help summary. If a *Subcommand* variable is specified, help information is displayed for that subcommand.
close	Closes the TELNET connection and returns to **telnet** command mode when the **open** subcommand is used to establish the connection. When the **telnet** command is invoked and a host is specified, the **close** subcommand closes the TELNET connection and exits the **telnet** program (identical to the **quit** subcommand).
display [*Argument*]	Displays all of the **set** and **toggle** values if no *Argument* variable is specified; otherwise, lists only those values that match the *Argument* variable.
emulate *TerminalType*	Overrides terminal−type negotiation with the specified terminal type. Possible choices are: **?** Prints help information. **3270** Emulates a 3270 terminal. **none** Specifies no emulation. **vt100** Emulates a DEC VT100 terminal.

All output received from the remote host is processed by the specified emulator. The initial terminal type to emulate can be specified through the **EMULATE** environment variable or the **−e** flag to the **telnet** command.

Note: Only standard ASCII characters are allowed in emulation mode.

mode *Type*	Specifies the current input mode. When the *Type* variable has a value of **line**, the mode is line−by−line. When the *Type* variable has a value of **character**, the mode is character−at−a−time. Permission is requested from the remote host before entering the requested mode, and if the remote host supports it, the new mode is entered.
open *Host* [*Port*]	Opens a connection to the specified host. The *Host* specification can be either a host name or an Internet address in dotted−decimal form. If no *Port* variable is specified, the **telnet** subcommand attempts to contact a TELNET server at the default port.

quit Closes a TELNET connection and exits the **telnet** program. A Ctrl−D in command mode also closes the connection and exits.

send *Arguments* Sends one or more arguments (special character sequences) to the remote host. Multiple arguments are separated by spaces. The following arguments can be used:

- **?** Prints help information for the **send** subcommand.
- **ao** Sends the TELNET AO (Abort Output) sequence, which causes the remote host to flush all output from the remote system to the local terminal.
- **ayt** Sends the TELNET AYT (Are You There) sequence, to which the remote system can respond.
- **brk** Sends the TELNET BRK (Break) sequence, which causes the remote system to perform a kill operation.
- **ec** Sends the TELNET EC (Erase Character) sequence, which causes the remote host to erase the last character entered.
- **el** Sends the TELNET EL (Erase Line) sequence, which causes the remote system to erase the line currently being entered.
- **escape** Sends the current **telnet** escape character. The default escape sequence is Ctrl−] for the **telnet** command, Ctrl−T for the **tn** command, or Ctrl−C for the **tn3270** command.
- **ga** Sends the TELNET GA (Go Ahead) sequence, which provides the remote system with a mechanism to signal the local system to return control to the user.
- **ip** Sends the TELNET IP (Interrupt Process) sequence, which causes the remote system to cancel the currently running process.
- **nop** Sends the TELNET NOP (No Operation) sequence.
- **sak** Sends the TELNET SAK (Secure Attention Key) sequence, which causes the remote system to invoke the trusted shell. If the SAK is not supported, then an error message is displayed that reads:
  ```
  Remote side does not support SAK.
  ```
- **synch** Sends the TELNET SYNC sequence, which causes the remote system to discard all previously typed input that has not yet been read. This sequence is sent as TCP/IP urgent data.

set *VariableValue* Sets the specified TELNET variable to the specified value. The special value **off** turns off the function associated with the variable entered. The **display** subcommand can be used to query the current setting of each variable. The variables that can be specified are:

- **echo** Toggles between local echo of entered characters and suppressing local echo. Local echo is used for normal processing, while suppressing the echo is convenient for entering text that should not be displayed on the screen, such as passwords. This variable can only be used in line−by−line mode.
- **eof** Defines the character for the **telnet** command. When the **telnet** command is in line−by−line mode, entering the eof character as the first character on a line sends the character to the remote host. The initial value for the eof character is the local terminal End−Of−File character.
- **erase** Defines the erase character for the **telnet** command. When the **telnet** command is in character−at−a−time mode and **localchars** has a value of **true**, typing the erase character sends the TELNET EC sequence to the remote host. The initial value for the erase character is the local terminal ERASE character.
- **escape** Specifies the **telnetescape** character, which puts the **telnet** command into command mode when connected to a remote host. This character can also be specified in octal in the **TNESC** environment variable.
- **flushoutput** Defines the flush character for the **telnet** command. When **localchars** has a value of **true**, typing the flushoutput character sends the TELNET AO

sequence to the remote host. The initial value for the flush character is Ctrl−O. If the remote host is running AIX, the **flushoutput** variable, unlike the other special characters defined by the **set** subcommand, only works in **localchars** mode since it has no **termio** equivalent.

interrupt Defines the interrupt character for the **telnet** command. When **localchars** has a value of **true**, typing the interrupt character sends the TELNET IP sequence to the remote host. The initial value for the interrupt character is the local terminal interrupt (INTR) character.

kill Defines the kill character for the **telnet** command. When the **telnet** command is in character−at−a−time mode and **localchars** has a value of **true**, typing the kill character sends the TELNET EL sequence to the remote host. The initial value for the kill character is the local terminal KILL character.

quit Defines the quit character for the **telnet** command. When **localchars** has a value of **true**, typing the quit character sends the TELNET BRK sequence to the remote host. The initial value for the quit character is the local terminal QUIT character.

sak Defines the Secure Attention Key (SAK) for the **telnet** command. When the sak character is entered, the remote system is asked to create a trusted shell. If the remote host does not support the SAK, this sequence has no effect.

status Shows the status of the **telnet** command, including the current mode and the currently connected remote host.

toggle *Arguments* Toggles one or more arguments that control how the **telnet** command responds to events. Possible values are **true** and **false**. Multiple arguments are separated by spaces. The **display** subcommand can be used to query the current setting of each argument. The following arguments can be used:

? Displays valid arguments to **toggle**.

autoflush If **autoflush** and **localchars** both have a value of **true** and the AO, INTR, and QUIT characters are recognized and transformed into TELNET sequences, the **telnet** command does not display any data on the user's terminal until the remote system acknowledges (with a TELNET **timing mark** option) that it has processed those TELNET sequences. The initial value of **autoflush** is **true** if the terminal has not done an **stty noflsh**, and **false** if it has.

autosynch If **autosynch** and **localchars** are both **true**, then typing the INTR or QUIT character sends that character's TELNET sequence, followed by the TELNET SYNC sequence. This procedure causes the remote host to discard all previously typed input until both of the TELNET sequences have been read and acted upon. The initial value of this toggle is **false**.

crmod Toggles carriage return mode. When set to **true**, most carriage return characters received from the remote host are mapped into a carriage return followed by a line feed. This mode does not affect the characters typed by the user, only those received from the remote host. This mode is useful when the remote host sends only a carriage return and not a line feed. The initial value of this toggle is **false**.

debug Toggles debugging at the socket level. The initial value of this toggle is **false**.

localchars Determines the handling of TELNET special characters. When this value is **true**, the ERASE, FLUSH, INTERRUPT, KILL, and QUIT characters are recognized locally and transformed into the appropriate TELNET control sequences (EC, AO, IP, BRK, and EL, respectively). When this value is **false**, these special characters are sent to the remote host as literal characters. The initial value of **localchars** is **true** in line−by−line mode and **false** in character−at−a−time mode.

netdata Toggles the display of all network data (in hexadecimal format). The data is written to standard output unless a *TraceFile* value is specified with the **−n** flag on the **telnet** command line. The initial value of this toggle is **false**.

options Toggles the display of internal TELNET Protocol processing options, such as terminal negotiation and local or remote echo of characters. The initial value of this toggle is **false**, indicating that the current options should not be displayed.

lineterm Toggles the default end−of−line terminator to CR−LF (ASCII carriage−return line−feed). A telnet client running on an ASCII host should have the user configurable option to send either the CR−NUL or CR−LF terminator when the user presses the end−of−line key. The initial value of this toggle is **false**.

z Suspends the TELNET process. To return to the TELNET process, use the **fg** built−in command of the **csh** or **ksh** command.

Note: The **z** subcommand has the same effect as a Ctrl−Z key sequence for any other process. It suspends Telnet execution and returns you to your original login shell.

Authentication

If the system is configured for Kerberos 5 authentication, the telnet client will attempt authentication negotiation. The authentication negotiation used by telnet and the definitions of the options and suboptions for this are defined in rfc 1416.

If the client and server agree on an authentication type, they will exchange authentication information including the account the client wants to access. This will be the local user unless the **−l** flag is set.

If they cannot agree on the authentication information or if it fails, the telnet connection will continue with the standard connection (provided Standard AIX is configured).

The remote host allows access only if all of the following conditions are satisfied:

- The local user has current DCE credentials.
- The remote system accepts the DCE credentials as sufficient for access to the remote account. See the **kvalid_user** function for additional information.

Examples

In the following examples, if you enter the **tn** command instead of the **telnet** command, the command mode prompt is displayed as `tn>`.

1. To log in to the remote host `host1` and perform terminal negotiation, enter:

   ```
   telnet host1
   ```

2. To log in to `host1` as a **vt100** terminal (no terminal type negotiation), choose one of the following methods:
   a. Use the following commands to set the **EMULATE** environment variable for this login session, then enter the **telnet** command:

```
EMULATE=vt100; export EMULATE
telnet host1
```

3. Use the **−e** flag to set the terminal type for this **telnet** session only:

```
telnet −e vt100 host1
```

- To log in to a remote host and then check the status of the **telnet** program, enter:

```
telnet host3
```

When the login prompt appears, enter your login ID and password. Press the Ctrl−T key sequence to receive the `telnet>` prompt. Enter the following at the `telnet>` prompt:

```
status
```

Information similar to the following is displayed on your screen:

```
Connected to host3.
Operating in character−at−a−time mode.
Escape character is '^]'.
```

Upon completion of the **status** subcommand, press the Enter key to return to the remote prompt.

Once you have completed your login, you can issue commands. To log out of the system and close the connection, press the Ctrl−D key sequence, or exit.

- To log in to a remote host using the **tn3270** command, enter:

```
tn3270 hostname
```

The host login screen should be displayed. You can now enter your login ID and password. Once you have completed your login, you can issue commands. To log out of the system and close the connection, press Ctrl−D or exit.

Files

/etc/3270.keys Defines base 3270−keyboard mapping for use with limited function terminals.

Related Information

The **env** command, **ftp** command, **login** command, **rcp** command, **rexec** command, **rlogin** command, **rsh** command.

The **telnetd** daemon.

The **kvalid_user** function.

The **map3270** file format, **.3270keys** file format.

Network Overview in *AIX Version 4.3 System User's Guide: Communications and Networks*.

Connecting a Local Host to a Remote Host in *AIX Version 4.3 System User's Guide: Communications and Networks*.

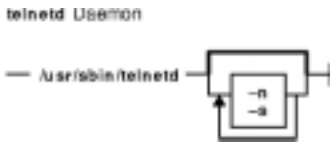
Secure Rcmds in *AIX Version 4.3 System User's Guide: Communications and Networks*.

telnetd Daemon

Purpose

Provides the server function for the TELNET protocol.

Syntax



/usr/sbin/telnetd [**−n**] [**−s**]

Description

> **Note:** The **telnetd** daemon is normally started by the **inetd** daemon. It can also be controlled from the command line, using SRC commands.

The **/usr/sbin/telnetd** daemon is a server that supports the Defense Advanced Research Product Agency (DARPA) standard Telnet Protocol (TELNET). Changes to the **telnetd** daemon should be made using the System Management Interface Tool (SMIT).

Changes to the **telnetd** daemon can be made using the System Management Interface Tool (SMIT) or System Resource Controller (SRC), by editing the **/etc/inetd.conf** or /**etc**/**services** file. Entering `telnetd` at the command line is not recommended. The **telnetd** daemon is started by default when it is uncommented in the **/etc/inetd.conf** file.

The **inetd** daemon get its information from the **/etc/inetd.conf** file and the **/etc/services** file.

After changing the **/etc/inetd.conf** or **/etc/services** file, run the **refresh−s inetd** or **kill −1** *InetdPID* command to inform the **inetd** daemon of the changes to its configuration file.

When a **telnet** session is started, the **telnetd** daemon sends TELNET options to the client (remote) host to indicate an ability to perform options.

Terminal Negotiation

The **telnetd** daemon requests the terminal type from the client host. On receipt, the **telnetd** daemon checks whether the indicated type is supported on the local system. If not, the daemon requests a terminal type again.

This terminal type negotiation continues until the remote client sends an acceptable terminal type or until the client sends the same type twice in a row, indicating that it has no other types available. When necessary, the **telnetd** daemon refers to the **/etc/telnet.conf** file to translate a client's terminal−type strings into **terminfo** file entries.

> **Note:** Since the **telnetd** daemon allows the sending and receiving of 8−bit ASCII, NLS is supported.

If the remote client sends the TELNET **SAK** command, the **telnetd** daemon passes the local SAK characters through the PTY to invoke the trusted shell.

The **telnetd** daemon supports the following TELNET options:

- Binary
- Echo/no echo
- Support SAK
- Suppress go ahead
- Timing mark
- Negotiate About Window Size (NAWS)
- Authentication
- Binary
- Echo/no echo
- Support SAK
- Suppress go ahead
- Timing mark
- Negotiate About Window Size (NAWS)

The **telnetd** daemon also recognizes the following options for the remote client:

- Binary
- Suppress go ahead
- Echo/no echo
- Terminal type

The **telnetd** daemon should be controlled using the System Management Interface Tool (SMIT) or by changing the **/etc/inetd.conf** file. Entering `telnetd` at the command line is not recommended.

Authenication Negotiation

If the system has Kerberos 5 authentication configured, **telnetd** will accept authentication option negotiation. If both agree on Kerberos 5 authentication, the client will pass over the DCE principal and **telnetd** will use the **kvalid_user** routine to determine if the DCE principal should have access to the account. If it passes, no password will be requested.

Manipulating the telnetd Daemon with the System Resource Controller

The **telnetd** daemon is a subserver of the **inetd** daemon, which is a subsystem of the System Resource Controller (SRC). The **telnetd** daemon is a member of the **tcpip** SRC subsystem group. This daemon is enabled by default in the **/etc/inetd.conf** file and can be manipulated by the following SRC commands:

startsrc Starts a subsystem, group of subsystems, or a subserver.

stopsrc Stops a subsystem, group of subsystems, or a subserver.

lssrc Gets the status or a subsystem, group or subsystems, or a subserver.

Flags

−n Disables transport−level keep−alive messages. Messages are enabled by default.

−s Turns on socket−level debugging.

Examples

Note: The arguments for the **telnetd** daemon can be specified by using SMIT or by editing the **/etc/inetd.conf** file.

1. To start the **telnetd** daemon, enter the following:

   ```
   startsrc -t telnet
   ```

   This command starts the **telnetd** subserver.

2. To stop the **telnetd** daemon normally, enter the following:

   ```
   stopsrc -t telnet
   ```

   This command allows all pending connections to start and existing connections to complete but prevents new connections from starting.

3. To force stop the **telnetd** daemon and all **telnetd** connections, enter the following:

   ```
   stopsrc -t -f telnet
   ```

   This command terminates all pending connections and existing connections immediately.

4. To display a short status report about the **telnetd** daemon, enter the following:

   ```
   lssrc -t telnet
   ```

This command returns the daemon's name, process ID, and state (active or inactive).

File

terminfo Describes terminal by capability.

Related Information

The **ftp** command, **kill** command, **lssrc** command, **rcp** command, **refresh** command, **rlogin** command, **rsh** command, **startsrc** command, **stopsrc** command, **telnet** command.

The **kill** command, **lssrc** command, **refresh** command, **startsrc** command, **stopsrc** command, **telnet** command.

The **/etc/inetd.conf** file format, **/etc/telnet.conf** file format.

The **pty** special file.

The **kvalid_user** subroutine.

Telnet Protocol (TELNET) in *AIX Version 4.3 System Management Guide: Communications and Networks*.

TCP/IP Daemons in *AIX Version 4.3 System Management Guide: Communications and Networks*.

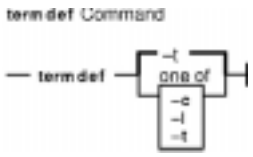
Secure Rcmds in *AIX Version 4.3 System User's Guide: Communications and Networks*.

termdef Command

Purpose

Queries terminal characteristics.

Syntax



termdef [**−c** | **−l** | **−t**]

Description

The **termdef** command identifies the current display type, the active lines setting, or the current columns setting. This simplifies resetting the lines and columns when you switch fonts as well as resetting the **TERM** environment variable when you switch displays. The terminfo database defines the default number of lines and columns for each display, but the lines and columns can change depending upon which font is currently active. Also, the **TERM** environment variable does not automatically reflect the currently active display.

The flags for the **termdef** command are mutually exclusive. If you use more than one flag with the command, the **termdef** command recognizes and returns the current value for the first flag only. Any other flags are ignored. For example, the **termdef −lc** command returns only the active lines setting for the current display.

Flags

−c Returns the current column value.

−l Returns the current line value.

−t Returns the name of the current display (the default action).

Example

To determine the current value of the **TERM** environment variable, enter:

```
termdef −c
```

File

/usr/bin/termdef Contains the **termdef** command.

Related Information

None

test Command

Purpose

Evaluates conditional expressions.

Syntax

test*Expression*

OR

[*Expression* **]**

Description

The **test** command evaluates the *Expression* parameter, and if the expression value is True, returns a zero (True) exit value. Otherwise, the **test** command returns a nonzero (False) exit value. The **test** command also returns a nonzero exit value if there are no parameters.

Notes:

1. In the second form of the command, the **[ ]** (brackets) must be surrounded by blank spaces.
2. You must test explicitly for file names in the C shell. File−name substitution (globbing) causes the shell script to exit.

Functions and operators are treated as separate parameters by the **test** command. The *Expression* parameter refers to a statement that is checked for a true or false condition. The following functions are used to construct this parameter:

−b *FileName* Returns a True exit value if the specified *FileName* exists and is a block special file.
−c *FileName* Returns a True exit value if the specified *FileName* exists and is a character special file.
−d *FileName* Returns a True exit value if the specified *FileName* exists and is a directory.
−e*FileName* Returns a True exit value if the specified *FileName* exists.
−f *FileName* Returns a True exit value if the specified *FileName* exists and is a regular file.
−g *FileName* Returns a True exit value if the specified *FileName* exists and its Set Group ID bit is set.

−h *FileName* Returns a True exit value if the specified *FileName* exists and is a symbolic link.
−k *FileName* Returns a True exit value if the specified *FileName* exists and its sticky bit is set.

−L *FileName* Returns a True exit value if the specified *FileName* exists and is a symbolic link.
−n *String1* Returns a True exit value if the length of the *String1* variable is nonzero.
−p *FileName* Returns a True exit value if the specified *FileName* exists and is a named pipe (FIFO).

−r *FileName*	Returns a True exit value if the specified *FileName* exists and is readable by the current process.
−s *FileName*	Returns a True exit value if the specified *FileName* exists and has a size greater than 0.
−t *FileDescriptor*	Returns a True exit value if the file with a file descriptor number of *FileDescriptor* is open and associated with a terminal.
−u *FileName*	Returns a True exit value if the specified *FileName* exists and its Set User ID bit is set.
−w *FileName*	Returns a True exit value if the specified *FileName* exists and the write flag is on. However, the *FileName* will not be writable on a read−only file system even if **test** indicates true.
−x *FileName*	Returns a True exit value if the specified *FileName* exists and the execute flag is on. If the specified file exists and is a directory, the True exit value indicates that the current process has permission to search in the directory.
−z *String1*	Returns a True exit value if the length of the *String1* variable is 0 (zero).
*String1***=** *String2*	Returns a True exit value if the *String1* and *String2* variables are identical.
*String1***!=***String2*	Returns a True exit value if the *String1* and *String2* variables are not identical.
String1	Returns a True exit value if the *String1* variable is not a null string.
Integer1 **−eq** *Integer2*	Returns a True exit value if the *Integer1* and *Integer2* variables are algebraically equal. Any of the comparisons **−ne**, **−gt**, **−ge**, **−lt**, and **−le** can be used in place of **−eq**.

These functions can be combined with the following operators:

!	Unary negation operator
−a	Binary AND operator
−o	Binary OR operator (that is, the **−a** operator has higher precedence than the **−o** operator)
\(*Expression***\)**	Parentheses for grouping

Exit Status

This command returns the following exit values:

0 The *Expression* parameter is true.

1 The *Expression* parameter is false or missing.

>1 An error occurred.

Examples

1. To test whether a file exists and is not empty, enter:

   ```
   if test ! −s "$1"
   then
      echo $1 does not exist or is empty.
   fi
   ```

   If the file specified by the first positional parameter to the shell procedure, `$1`, does not exist, the **test** command displays an error message. If `$1` exists and has a size greater than 0, the **test** command displays nothing.

   Note: There must be a space between the **−s** function and the file name.

   The quotation marks around `$1` ensure that the test works properly even if the value of `$1` is a null string. If the quotation marks are omitted and `$1` is the empty string, the **test** command displays the

error message `test: argument expected.`

2. To do a complex comparison, enter:

```
if [ $# -lt 2  -o  ! -e "$1" ]
then
   exit
fi
```

If the shell procedure is given fewer than two positional parameters or the file specified by `$1` does not exist, then the shell procedure exits. The special shell variable `$#` represents the number of positional parameters entered on the command line that starts this shell procedure.

The "Shells Overview" in *AIX Version 4.3 System User's Guide: Operating System and Devices* describes shells in general, defines terms that are helpful in understanding shells, and describes the more useful shell functions.

File

/usr/bin/test Contains the **test** command.

Related Information

The **bsh** command, **csh** command, **find** command, **ksh** command, **sh** command.

Shells Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

tftp or utftp Command

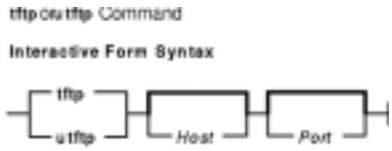
Purpose

Transfers files between hosts using the Trivial File Transfer Protocol (TFTP).

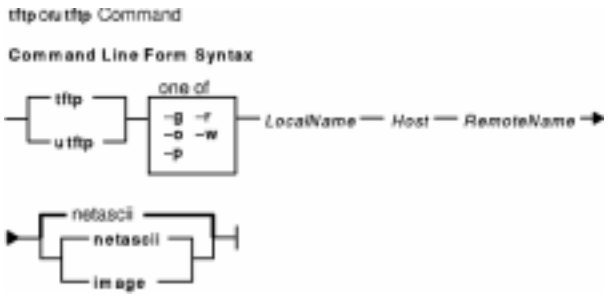
Syntax

{**tftp** | **utftp**} {**−g** | **−o** | **−p** | **−r** | **−w** } *LocalName HostPort RemoteName* [**netascii** | **image**]

Interactive Form Syntax



Command Line Form Syntax



Description

The **/usr/bin/tftp** and **utftp** commands transfer files between hosts using the Trivial File Transfer Protocol (TFTP). Since TFTP is a minimal file transfer protocol, the **tftp** and **utftp** commands do not provide all of the features of the **ftp** command. For example, the **tftp** and **utftp** commands do not provide the ability to list remote files or change directories at the remote host, and only limited file access privileges are given to the remote TFTP server. The **utftp** command is a form of the **tftp** command for use in a pipe.

The remote host must have a **tftpd** daemon started by its **inetd** daemon and have an account defined that limits the access of the **tftpd** daemon. Use the procedure defined by the **tftpd** command to setup the TFTP environment and the nobody account.

> **Note:** The **tftp** and **utftp** commands should not be available when your host is operating in secure mode.

The **tftp** command ignores duplicate acknowledgments for any block sent and sends an error packet and exit if a block with an inappropriate (future) block number arrives. It also ignores duplicate data blocks if they have already been received and sends an error packet and exits.

Maximum Time−out Value

The user can pick the maximum time−out value, but the initial time−out value for the first block is hardcoded. The user cannot pick the maximum time−out value for the server; the server times out after six retries with a maximum time−out value of 64 seconds.

Access Control

The **/etc/tftpaccess.ctl** file is searched for lines that start with `allow:` or `deny:`. Other lines are ignored. If the file doesn't exist, access is allowed. The allowed directories and files can be accessed and the denied directories cannot be accessed. For example, the **/usr** directory might be allowed and the **/usr/ucb** directory might be denied. This means that any directory or file in the **/usr** directory, except the **/usr/ucb** directory, can be accessed. The entries in the **/etc/tftpaccess.ctl** file must be absolute path names.

The **/etc/tftpaccess.ctl** file should be write−only by the root user and readable by all `groups` and `others` (that is, owned by `root` with permissions of 644). The user `nobody` must be able to read the **/etc/tftpaccess.ctl** file. Otherwise, the **tftpd** daemon is not able to recognize the existence of the file and allows access to the entire system. For more information, refer to the sample **tftpaccess.ctl** file, which resides in the **/usr/samples/tcpip** directory.

The search algorithm assumes that the local path name used in the **tftp** command is an absolute path name. It searches the **/etc/tftpaccess.ctl** file looking for `allow:/`. It repeatedly searches for allowed path names with each partial path name constructed by adding the next component from the file path name. The longest path name matched is the one allowed. It then does the same with denied names, starting with the longest allowed path name matched.

For example, if the file path name were **/a/b/c** and the **/etc/tftpaccess.ctl** file contained `allow:/a/b` and `deny:/a`, one allowed match would be made (`/a/b`) and no denied match starting with `/a/b` would be made, and access would be allowed.

If the **/etc/tftpaccess.ctl** file contained `allow:/a` and `deny:/a/b`, one allowed match would be made (`/a`) and one denied match starting with `/a` (`/a/b`) would be made, and access would be denied. If the **/etc/tftpaccess.ctl** file contained `allow:/a/b` and also contained `deny:/a/b`, access would be denied because allowed names are searched first.

> **Note:** Further information and example configurations for Xstations, Diskless clients, and restricted entry can be found in the **/usr/samples/tcpip/tftpaccess.ctl** file.

The **tftp** and **utftp** commands have two forms: interactive form and command−line form.

Interactive Form

In the interactive form, the **tftp** and **utftp** commands are issued alone or with a *Host* parameter that specifies the default host to use for file transfers during this session. If you choose, you can also specify with the *Port* parameter which port the **tftp** or **utftp** connection should use, such as the one specified for **mail** in the **/etc/services** file. When you enter the interactive form of either of these commands, the `tftp>` prompt is displayed.

When transferring data to a remote host, the transferred data is placed in the directory specified by the *RemoteName* parameter. The remote name must be a fully specified file name, and the remote file must both exist and have write permission set for others. The **tftp** command attempts to write the data to the specified file. However, if the remote TFTP server does not have the appropriate privileges to write the remote file or if the file does not already exist, the transfer is unsuccessful. This can be overridden using the **tftpd** daemon.

Command−Line Form

The command−line forms of the **tftp** and **utftp** commands are equivalent, except that the **utftp** command does not overwrite a local file. The **tftp** command can overwrite a file, but prompts the user before doing so. Because it is not interactive, the command line form of the **utftp** command can be more useful than the **tftp** command in a pipe. In the command line form, all of the arguments to either command are specified on the command line, and no prompt is displayed.

Subcommands

The **tftp** and **utftp** subcommands can be entered in either their interactive form or in their command−line form.

Subcommands Used in the Interactive Form

Once the `tftp>` prompt is displayed, the following subcommands can be issued:

? [*Subcommand*] Displays help information. If a *Subcommand* parameter is specified, only information about that subcommand is displayed.

ascii Synonym for the **mode ascii** subcommand.

binary Synonym for the **mode binary** subcommand. This subcommand is used in the interactive mode. The **image** subcommand accomplishes the same thing as the **mode binary** subcommand, but is used on the command line.

connect *Host* [*Port*] Sets the remote host, and optionally the port, for file transfers. Since the TFTP protocol does not maintain connections between transfers, the **connect** subcommand does not create a connection to the specified host, but stores it for transfer operations. Because the remote host can be specified as part of the **get** or **put** subcommand, which overrides any host previously specified, the **connect** subcommand is not required.

get *RemoteFile* [*LocalFile*]

get*RemoteFileRemoteFileRemoteFile* [*RemoteFile* . . .]

Gets a file or set of files from the remote host to the local host. Each of the *RemoteFile* parameters can be specified in one of the following two ways:

- As a file (*File*) that exists on the remote host if a default host has already been specified.
- As a host file (*Host*:*File*), where *Host* is the remote host and *File* is the name of the file to copy to the local system. If this form of the parameter is used, the last host specified becomes the default host for later transfers in this **tftp** session.

mode *Type* Sets the type (*Type*) of transfer mode to either **ascii** or **binary**. A transfer mode of **ascii** is the default.

put *LocalFile* [*RemoteFile*]

put *LocalFile LocalFile LocalFile* [*LocalFile* . . .] *RemoteDirectory*

Puts a file or set of files from the local host onto the remote host. The *RemoteDirectory* and *RemoteFile* parameters can be specified in one of the following two ways:

- As a file or directory that exists on the remote host if a default host has already been specified.
- With *Host*:*RemoteFile* parameter, where *Host* is the remote host and *RemoteFile* is the name of the file or directory on the remote system. If this form of the parameter is used, the last host specified becomes the default host for later transfers in this **tftp** session.

In either case, the remote file or directory name must be a fully specified path name, even if the local and remote directories have the same name. If a remote directory is specified, the remote host is assumed to be a UNIX machine. The default value of the

	put subcommand is write−replace, but you can add an option in the **tftpd** daemon to allow write−create.
quit	Exits the **tftp** session. An End−Of−File key sequence also exits the program.
status	Shows the current status of the **tftp** program, including, for example, the current transfer mode (**ascii** or **binary**), connection status, and time−out value.
timeout *Value*	Sets the total transmission time out to the number of seconds specified by the *Value* parameter.
trace	Turns packet tracing on or off.
verbose	Turns verbose mode, which displays additional information during file transfer, on or off.

Subcommands Used in the Command Line Form

In this form, if the *Action* flag is:

−w or **−p** Writes (or puts) local data, specified by the *LocalName* parameter, to the file specified by the *RemoteName* parameter on the remote host specified by the *Host* parameter. If the *LocalName* parameter is a file name, the **tftp** command transfers the specified local file. If the *LocalName* parameter is specified as a − (dash), the **tftp** command transfers data from local standard input to the remote host. When the *LocalName* parameter is standard input, the **tftp** command allows 25 seconds for all input to be entered before it times out.

−r or **−g** or **−o** Reads (or gets) remote data from the file specified by the *RemoteName* parameter at the remote host specified by the *Host* parameter and writes it to the file specified by the *LocalName* parameter. If the *LocalName* parameter is a file name, the **tftp** command writes the data to the specified local file. For the **−r** and **−g** actions, the **tftp** command prompts for verification before overwriting an existing local file. For the **−o** action, the **tftp** command overwrites an existing local file without prompting. If the *LocalName* parameter is specified as a − (dash), the **tftp** command writes the data to local standard output.

> **Note:** Since the **tftp −g** and **tftp −r** commands prompt before overwriting an existing local file, it may be impractical to use the **tftp** command in a pipe. The **utftp** command performs the same **−r** and **−g** actions as the **tftp** command, but simply stops before overwriting a local file. Thus, the **utftp** command may be more appropriate for use in a pipe.

For both of the following modes of file transfer, the *RemoteName* parameter is the name of a file that has write permission set for others. Note that the *RemoteName* parameter must be in double quotes (" ") if it contains shell special characters.

The mode of transfer is one of the following:

netascii Transfers the data as 7−bit ASCII characters in 8−bit transfer bytes. This is the default.

image Transfers the data as 8−bit binary data bytes in 8−bit transfer bytes, with no conversion. **image** transfer can be more efficient than **netascii** transfer when transferring between two hosts. It is recommended that **netascii** be used when transferring ASCII files from a workstation to a different type of host.

Examples

The following examples distinguish the differences between the interactive form and the command line form of the **tftp** command:

Using the Interactive Form of the tftp Command

To enter the **tftp** command, check the current status, connect to a remote host, and transfer a file from a remote host to your local host, enter:

```
tftp
```

The `tftp>` prompt is displayed. Enter the **status** subcommand following this prompt:

```
status
```

A message similar to the following is displayed on your screen:

```
Not connected.
Mode: netascii   Verbose: off   Tracing: off
Max-timeout: 25 seconds
tftp> _
```

After the `tftp>` prompt, enter the **connect** subcommand and the name of the remote system to which you want to connect:

```
tftp> connect host1
```

The `tftp>` prompt is displayed as an indication that you are connected to `host1`. Following the `tftp>` prompt, enter the **get** subcommand to transfer the file `update` from the remote host to your local host.

```
get /home/alice/update update
```

The `/home/alice` directory on the remote host must have read permission set for others. The `/home/alice/update` file from `host1` was transferred to the `update` file on your local system. In this example, the user is connected to `host1` and the `update` file is transferred from `host1` to the local host.

Using the Command Line Form of the tftp Command

1. To copy a text file from a remote host and write it to a local file, enter:

   ```
   tftp -g newsched host1 /home/john/schedule
   $ _
   ```

   In this example, the `/home/john/schedule` file was copied from the remote host `host1` and written to the local file `newsched`.

2. To copy a file from a remote host and redirect the output to standard output of the local host, enter:

   ```
   tftp -g - host3 /etc/hosts
   ```

   If the copy is successful, information similar to the following is displayed on your screen:

   ```
   192.100.13.3 nameserver
   192.100.13.3 host2
   192.100.13.5 host1
   192.100.13.7 host3
   192.100.13.3 timeserver
   Received 128 bytes in 0.4 seconds
   $ _
   ```

In this example, the `/etc/hosts` file from remote host `host3` was copied and the output redirected to standard output of the local host.

3. To copy a file from a remote host, pipe it to the **grep** command, and write it to a local file, enter:

```
utftp –g – host1 /home/john/schedule | grep Jones > jones.todo
$ _
```

In this example, the `/home/john/schedule` file was copied from the remote host `host1`. This file was then piped to the **grep** command and written into the local file `jones.todo`.

4. To copy a file to another system, enter:

```
tftp –p /home/jeanne/test host2 /tmp/test
```

If the copy is successful, information similar to the following is displayed on your screen:

```
Sent 94146 bytes in 6.7 seconds
```

In this example, the `/home/jeanne/test` file was sent to the `/tmp` directory on the remote host `host2`.

5. To copy a binary file to another system, enter:

```
tftp –p core host3 /tmp/core image
```

If the copy is successful, information similar to the following is displayed on your screen:

```
Sent 309295 bytes in 15 seconds
```

In this example, the binary file `core` from the current directory was sent to the `/tmp` directory on remote host `host3`.

Files

/etc/tftpaccess.ctl Allows or denies access to files and directories.

Related Information

The **ftp** command, **grep** command, **rcp** command.

The **ftpd** daemon, **inetd** daemon, **tftpd** daemon, **syslogd** daemon.

The **hosts** file format, **services** file format.

Copying Files Using the tftp Command in *AIX Version 4.3 System User's Guide: Communications and Networks*.

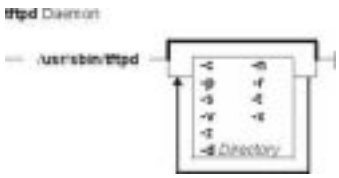
Network Overview in *AIX Version 4.3 System User's Guide: Communications and Networks*.

tftpd Daemon

Purpose

Provides the server function for the Trivial File Transfer Protocol.

Syntax



/usr/sbin/tftpd [**−c**] [**−n**] [**−p**] [**−r**] [**−v**] [**−t**] [**−s**] [**−x**] [**−z**] [**−d***Directory*]

Description

> **Note:** The **tftpd** daemon is normally started by the **inetd** daemon. It can also be controlled from the command line, using SRC commands.

The **/usr/sbin/tftpd** daemon runs the Trivial File Transfer Protocol (TFTP) server. Files sent using TFTP can be found in the directory specified by the full path name given on the **tftp** or **utftp** command line.

> **Note:** The **tftp** command, **utftp** command, and **tftpd** server are not available when the auditing system is in use. For more information, see Understanding Security for TCP/IP, the Auditing Overview, and the **audit** command.

Changes to the **tftpd** daemon can be made using the System Management Interface Tool (SMIT) or System Resource Controller (SRC), by editing the **/etc/inetd.conf** or **/etc/services** file. The **tftpd** daemon is started by default when it is uncommented in the **/etc/inetd.conf** file.

The **inetd** daemon get its information from the **/etc/inetd.conf** file and the **/etc/services** file.

After changing the **/etc/inetd.conf** or **/etc/services** file, run the **refresh−s inetd** or **kill −1** *InetdPID* command to inform the **inetd** daemon of the changes to its configuration file.

The **tftpd** server should have a user ID with the least privileges possible. The **nobody** ID allows the least permissions, and is the default user ID.

The **tftpd** daemon should be controlled using the System Management Interface Tool (SMIT) or by changing the **/etc/inetd.conf** file. Entering `tftpd` at the command line is not recommended.

With Release 4.3.x, the **tftpd** server has become a multithreaded application. Another major change is the server's ability to handle the new TFTP Blocksize Option (RFC1783). This new capability allows a client to negotiate a larger blocksize which improves tftp file transfer performance significantly. As a result, the boot time performance of diskless nodes using TFTP also improves significantly. The tftp client must be able to do blocksize negotiation to take advantage of this performance improvement. The blocksize option has not been implemented in the current tftp command.

tftpaccess.ctl File

The **/etc/tftpaccess.ctl** file is searched for lines that start with `allow:` or `deny:`. Other lines are ignored. If the file doesn't exist, access is allowed. The allowed directories and files minus the denied directories and files can be accessed. For example, the **/usr** directory might be allowed and the **/usr/ucb** directory might be denied. This means that any directory or file in the **/usr** directory, except the **/usr/ucb** directory, can be accessed. The entries in the **/etc/tftpaccess.ctl** file must be absolute path names.

The **/etc/tftpaccess.ctl** file should be write−only by the root user and readable by all `groups` and `others` (that is, owned by `root` with permissions of 644). The user `nobody` must be able to read the **/etc/tftpaccess.ctl** file. Otherwise, the **tftpd** daemon is not able to recognize the existence of the file and allows access to the entire system. For more information, refer to the sample **tftpaccess.ctl** file, which resides in the **/usr/samples/tcpip** directory.

The search algorithm assumes that the local path name used in the **tftp** command is an absolute path name. It searches the **/etc/tftpaccess.ctl** file looking for `allow:`/. It repeatedly searches for allowed path names with each partial path name constructed by adding the next component from the file path name. The longest path name matched is the one allowed. It then does the same with denied names, starting with the longest allowed path name matched.

For example, if the file path name were **/a/b/c** and the **/etc/tftpaccess.ctl** file contained `allow:/a/b` and `deny:/a`, one allowed match would be made (`/a/b`) and no denied match starting with `/a/b` would be made, and access would be allowed.

If the **/etc/tftpaccess.ctl** file contained `allow:/a` and `deny:/a/b`, one allowed match would be made (`/a`) and one denied match starting with `/a` (`/a/b`) would be made, and access would be denied. If the **/etc/tftpaccess.ctl** file contained `allow:/a/b` and also contained `deny:/a/b`, access would be denied because allowed names are searched first.

Manipulating the tftpd Daemon with the System Resource Controller

The **tftpd** daemon is a subserver of the **inetd** daemon, which is a subsystem of the System Resource Controller (SRC). The **tftpd** daemon is a member of the **tcpip** SRC subsystem group. This daemon is enabled when it is uncommented in the **/etc/inetd.conf** file and can be manipulated by the following SRC commands:

startsrc	Starts a subsystem, group of subsystems, or a subserver.
stopsrc	Stops a subsystem, group of subsystems, or a subserver.
lssrc	Gets the status of a subsystem, group of subsystems, or a subserver.

Flags

−c	Specfies the maximum number of concurrent threads per process, excluding the initial thread.
−d*Directory*	Specfies default destination directory. The *Direcrtory* specified will be used as the home directory for storing files only. This default directory will be used only if a full pathname is not specified. The default directory for retrieving files is still **/tftpboot**.
−i	Logs the IP address of the calling machine with error messages.
−n	Allows the remote user to create files on your machine. Remote users are only allowed to read files with read permission for other if this flag is not specified.
−p	Specfies the port number for the incoming request.
−r	Attempts to convert the IP address to the appropriate host name before it logs messages. This flag must be used with the **−i** flag or the **−v** flag.
−s	Turns on socket−level debugging.

−t	Specfies the timeout value for datagrams.
−v	Logs information messages when any file is successfully transferred by the **tftpd** daemon. This logging keeps track of who is remotely transferring files to and from the system with the **tftpd** daemon.
−x	Specifies the maximum of timeouts waiting for a datagram.
−z	Specfies the maximum allowed segment size for transfers.

Examples

Note: The arguments for the **tftpd** daemon can be specified by using SMIT or by editing the **/etc/inetd.conf** file.

1. To start the **tftpd** daemon, enter the following:

   ```
   startsrc −t tftp
   ```

   This command starts the **tftpd** subserver.

2. To stop the **tftpd** daemon normally, enter the following:

   ```
   stopsrc −t tftp
   ```

   This command allows all pending connections to start and existing connections to complete but prevents new connections from starting.

3. To force stop the **tftpd** daemon and all **tftpd** connections, enter the following:

   ```
   stopsrc −t −f tftp
   ```

   This command terminates all pending connections and existing connections immediately.

4. To display a short status report about the **tftpd** daemon, enter the following:

   ```
   lssrc −t tftp
   ```

This command returns the daemon's name, process ID, and state (active or inactive).

Related Information

The **kill** command, **lssrc** command, **refresh** command, **startsrc** command, **stopsrc** command, **tftp** command.

The **inetd** daemon.

The **/etc/inetd.conf** file format.

Trivial File Transfer Protocol (TFTP) in *AIX Version 4.3 System Management Guide: Communications and Networks*.

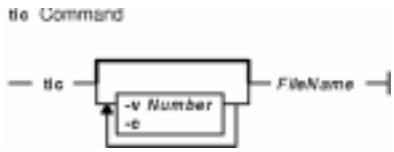
TCP/IP Daemons in *AIX Version 4.3 System Management Guide: Communications and Networks*.

tic Command

Purpose

Translates the terminfo description files from source to compiled format.

Syntax



tic [**−v** [*Number*]] [**−c**] *FileName*

Description

The **tic** command translates the terminfo files from the source format into the compiled format. The **tic** command places the results in the **/usr/share/lib/terminfo** directory. If the **TERMINFO** environment variable is set, the results are placed there instead of in the **/usr/share/lib/terminfo** directory.

The **tic** command compiles all terminfo descriptions in *FileName*. When the **tic** command finds a `use=` entry−*name* field, it searches the current file first, If unable to find the entry −*name*, it obtains the entry from the binary file in **/usr/share/lib/terminfo**. If **TERMINFO** is set, the terminfo directory is searched before **/usr/share/lib/terminfo**.

The total compiled entries cannot exceed 4096 bytes, and the name field cannot exceed 128 bytes.

Flags

−v[*Number*]	Writes trace information on the progress of the **tic** command. *Number* is an integer from 1 to 10 inclusive that increases the level of the verbosity. If *Number* is omitted, the default level is 1. The amount of information output increases as *Number* increases.
−c	Only checks *FileName* for errors. Errors in *use=entry−name* are not detected.

Files

/usr/share/lib/termi

nfo/?/* Contains the compiled terminal capability database.

Related Information

The **terminfo** file format.

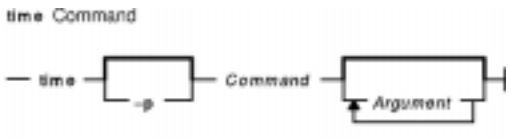
The Curses Overview for Programming in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

time Command

Purpose

Prints the time of the execution of a command.

Syntax



time [**−p**] *Command* [*Argument* ...]

Description

The **time** command prints the elapsed time during the execution of a command, time in the system, and execution time of the **time** command in seconds to standard error.

> **Note:** Sleep time is not charged to either system or user time.

The **time** command is also built into the C shell (**csh**) and Korn shell (**ksh**) with a different format. To run the **time** command while in the **csh** and **ksh** shells, enter:

```
/usr/bin/time
```

Flags

−p Writes the timing output to standard error. Seconds are expressed as a floating−point number with at least one digit following the radix character.

The standard format for this flag is as follows:

```
"real %f\nuser %f\nsys %f\n", <real seconds>, <user seconds>, <system seconds>
```

Exit Status

If you use the *Command* parameter, the exit status of the **time** command is the exit status of the specified command. Otherwise, the **time** command exits with one of the following values:

1−125 Indicates an error occurred in the **time** command.
126 Indicates the command specified by the *Command* parameter was found but could not be invoked.
127 Indicates the command specified by the *Command* parameter could not be found.

Examples

1. To measure the time required to run a program, enter:

```
/usr/bin/time -p a.out
```

This command runs the program **a.out** and writes the amount of real, user, and system time to standard error, in the format specified by the **-p** flag; for example:

```
real      10.5
user      0.3
sys       3.6
```

2. To save a record of the **time** command information in a file, enter:

```
/usr/bin/time a.out 2> a.time
```

Files

/usr/bin/time Specifies the path of the **time** command.

Related Information

The **timex** command.

Setting Up an Accounting System in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

Accounting Commands in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

Accounting Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

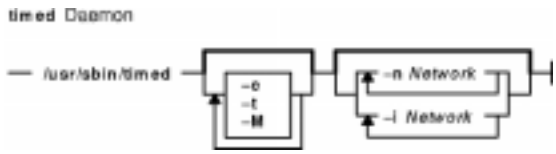
Using the time Command to Measure CPU Use in *AIX Versions 3.2 and 4 Performance Tuning Guide*.

timed Daemon

Purpose

Invokes the time server daemon.

Syntax



/usr/sbin/timed [**−c**] [**−M**] [**−t**] [[**−n** *Network*] ... | [**−i** *Network*] ...]

> **Note:** Use the **rc.tcpip** file to start the daemon with each initial program load. You can specify the **timed** daemon at the command line. You can also use SRC commands to control the **timed** daemon from the command line.

Description

The **timed** daemon synchronizes one machine's clock with those of other machines on the local area network that are also running the **timed** daemon. The **timed** daemon slows the clocks of some machines and speeds up the clocks on other machines to create an average network time.

When the **timed** daemon is started without the **−M** flag, the machine locates the nearest master time server and asks for the network time. Then the machine uses the **date** command to set the machine's clock to the network time. The machine accepts synchronization messages periodically sent by the master time server and calls the **adjtime** subroutine to perform the needed corrections on the machine's clock.

When the **timed** daemon is started with the **−M** flag, the machine polls each of its local area networks to determine which networks have master time servers. The machine becomes a master time server on the networks that do not have a master time server. The machine becomes a submaster time server on the networks that already have a master time server. The **timed** daemon creates the **/var/adm/timed.masterlog** file when the **timed** daemon is started with the **−M** flag. The **/var/adm/timed.masterlog** file contains a log of the deltas between the local machine's clock and the clocks of the other machines on the networks for which the local machine is the master time server. The **/var/adm/timed.masterlog** file is updated approximately every 4 minutes and is never cleared. You may need to clear this file to conserve disk space. If the machine is only a submaster time server on its networks, the **/var/adm/timed.masterlog** file remains empty. To clear the **/var/adm/timed.masterlog** file, enter:

```
cat /dev/null > /var/adm/timed.masterlog
```

If the master time server ceases to function on a network, a new master time server is elected from the submaster time servers on that network. The **timedc** command enables you to select which submaster time server becomes the master time server.

The **timed** daemon can be controlled using the System Resource Controller (SRC), the System Management Interface Tool (SMIT), or the command line. The **timed** daemon is not started by default. Use the **rc.tcpip** file to start the **timed** daemon with each initial program load.

Manipulating the timed Daemon with the System Resource Controller

The **timed** daemon is a subsystem controlled by the SRC. The **timed** daemon is a member of the SRC **tcpip** system group. Use the following SRC commands to manipulate the **timed** daemon:

startsrc	Starts a subsystem, group of subsystems, or a subserver.
stopsrc	Stops a subsystem, group of subsystems, or a subserver.
lssrc	Gets the short status of a subsystem, group of subsystems, or a subserver. The long status option usually found in **lssrc** is not supported for the timed daemon.

Flags

−c	Specifies that the master−timed daemon should ignore the time values it gets from the other slave−timed daemons when for calculating the average network time. This flag changes the network time to be the same as the system clock on the master−timed daemon.
−i *Network*	Specifies a network to be excluded from clock synchronization. The *Network* variable can be either a network address or a network name. If a network name is specified for the *Network* variable, the network name must be defined in the **/etc/networks** file. Specify one network address or network name with each **−i** flag. Do not use this flag with the **−n** flag.
−M	Specifies the machine is a master or submaster time server on its local area networks. If a master time server is not currently available on a network, the machine becomes the master time server for that network. If a master time server already exists on a network, the machine becomes a submaster time server on that network. However, the machine can become the master time server if the current master time server becomes inoperative. The **timed** daemon creates the **/var/adm/timed.masterlog** file when the **timed** daemon is started with the **−M** flag.
−n*Network*	Specifies a network to include in clock synchronization. The *Network* variable can be either a network address or a network name. If a network name is specified for the *Network* variable, the network name must be defined in the **/etc/networks** file. Specify one network address or network name with each **−n** flag. Do not use this flag with the **−i** flag.
−t	Allows the **timed** daemon to trace the messages it receives and store them in the **/var/adm/timed.log** file. You can also use the **timedc** command to activate tracing.

Examples

1. To start the **timed** daemon with SRC control, enter:

   ```
   startsrc −s timed
   ```

   This command starts the daemon. You can use this command in the **rc.tcpip** file or on the command line. The **−s** flag specifies that the subsystem that follows is to be started.

2. To stop the **timed** daemon normally with SRC control, enter:

   ```
   stopsrc −s timed
   ```

   This command stops the daemon. The **−s** flag specifies that the subsystem that follows is to be stopped.

3. To get a short status report from the **timed** daemon, enter:

   ```
   lssrc −s timed
   ```

   This command returns the name of the daemon, the process ID of the daemon, and the state of the daemon (active or inactive).

4. To start the **timed** daemon with SRC control as the master or submaster time server and to exclude networks `net1` and `net2` from clock synchronization, enter:

   ```
   startsrc −s timed −a "−M −i net1 −i net2"
   ```

   This command starts the daemon. The machine becomes the master or submaster time server for its networks. Networks `net1` and `net2` are excluded from clock synchronization. The **−s** flag specifies that the subsystem that follows is to be started. The **−a** flag specifies that the **timed** daemon should be started with the flags that follow. The flags must be enclosed in quotes.

5. To start the **timed** daemon, activate tracing, and include `net1` and `net2` in clock synchronization, enter:

   ```
   timed −t −n net1 −n net2
   ```

This command starts the daemon. Tracing is activated and both `net1` and `net2` are included in clock synchronization.

Files

/var/adm/timed.log	Contains the messages traced for the **timed** daemon. This file is created when the **timed** daemon is started with the **−t** flag or when tracing is enabled with the **timedc** command.
/etc/rc.tcpip	Contains the SRC commands to be executed at system startup.
/var/adm/timed.masterlog	Contains a log of the deltas between the master time server clock and the clocks of the other machines on the networks. This file is created when the **timed** daemon is started with the **−M** flag. However, this file only contains information for those networks on which the machine is the master time server.

Related Information

The **date** command, **timedc** command.

The **adjtime** subroutine, **gettimeofday** subroutine.

The **networks** file format.

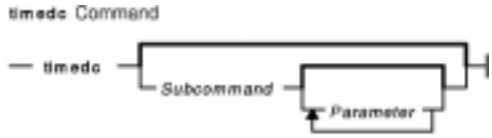
TCP/IP Daemons in *AIX Version 4.3 System Management Guide: Communications and Networks*.

timedc Command

Purpose

Returns information about the **timed** daemon.

Syntax



timedc [*Subcommand* [*Parameter* ...]]

Description

The **timedc** command controls the operation of the **timed** daemon. The **timedc** command does the following:

- Measures the difference between clocks on various machines on a network.
- Finds the location of the master time server.
- Enables or disables tracing of messages received by the **timed** daemon.
- Debugs.

Without any variables, the **timedc** command assumes an interactive mode and prompts for subcommands from standard input. If variables are supplied, the **timedc** command interprets the first variable as a subcommand and the remaining variables as parameters to the subcommand. You can redirect standard input so the **timedc** command reads subcommands from a file.

Variables

The **timedc** command recognizes the following subcommands:

? [*Parameter* ...]	Displays a short description of each variable specified in the parameter list. The **?** subcommand only works in interactive mode. If you give no variables, the **?** subcommand shows a list of subcommands recognized by the **timedc** command.
clockdiff *Host* ...	Computes the differences between the clock of the host machine and the clocks of the machines given as variables.
election *Host* ...	Requests that the **timed** daemon on the specified host (s) reset its election timers and ensure that a **timed** master server is available. Up to 4 hosts can be specified. If a master **timed** server is no longer available, then the **timed** daemon on the specified host (s) will request to become the new **timed** master server. The specified host(s) must be running the **timed** daemon in submaster mode with the **−M** flag.
help [*Parameter* ...]	Displays a short description of each subcommand specified in the parameter list. If you give no variables, the **help** subcommand shows a list of subcommands recognized by the **timedc** command.
msite	Finds the location of the master site.
quit	Exits the **timedc** command.

trace { **on** | **off** }	Enables or disables tracing of incoming messages to the **timed** daemon. The messages are held in the **/var/adm/timed.log** file.

You can use other commands for testing and debugging the **timed** daemon. Use the **help** command to find these commands.

These error messages may occur with the **timedc** command:

`Ambiguous command`	Abbreviation matches more than one command.
`Invalid command`	No match found.
`Privileged command`	Command can be executed only by the root user.

Examples

1. To display the time difference between the local host `sahara` and the remote host `sandy`, enter:

   ```
   timedc clockdiff sandy
   ```

   The output would be:

   ```
   time on sandy.austin.century.com is 37904247 ms ahead of time on
   sahara.austin.century.com
   ```

2. To display the client location of the **timed** daemon, enter:

   ```
   timedc msite
   ```

   The output would be:

   ```
   client timed daemon runs on bupu.austin.century.com
   ```

Related Information

The **date** command.

The **timed** daemon.

The **adjtime** subroutine.

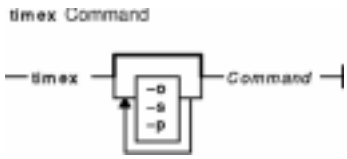
TCP/IP Daemons in *AIX Version 4.3 System Management Guide: Communications and Networks*.

timex Command

Purpose

Reports, in seconds, the elapsed time, user time, and system execution time for a command.

Syntax



timex [**−o**] [**−p**] [**−s**] *Command*

Description

The **timex** command reports, in seconds, the elapsed time, user time, and system execution time for a command. With specified flags, the **timex** command lists or summarizes process accounting data for a command and all of its children. *Command* is the name of any executable file on the system. It also reports total system activity during the execution interval. Output is written to standard error. The system uses the **/var/adm/pacct** file to select process records associated with the command and includes background processes with the same user ID, workstation ID, and execution time window.

Flags

−o Reports the total number of blocks read or written and total characters transferred by a command and all its children.

−p Lists process accounting records for a command and all its children. The number of blocks read or written and the number of characters transferred are reported. The **−p** flag takes the **f**, **h**, **k**, **m**, **r**, and **t** arguments defined in the **acctcom** command to modify other data items.

−s Reports total system activity during the execution of the command. All the data items listed in the **sar** command are reported.

> **Note:** Accounting must be turned on to use the **−o** or **−p** flags.

Examples

1. To report the total number of blocks read and total characters transferred by the **ls** command, enter:

   ```
   timex −o ls
   ```

2. To list the process accounting records for the **ps** command, enter:

   ```
   timex −p ps −fe
   ```

3. To report total system activity for the execution of the **ls** command, enter:

   ```
   timex −s ls
   ```

Files

/var/adm/pacct Used to select record associated with the command.

Related Information

The **acctcom** command, **sar** command, **time** command.

Setting Up an Accounting System in *AIX Version 4.3 System Management Guide: Operating System and Devices* describes the steps you must take to establish an accounting system.

See the Accounting Commands in *AIX Version 4.3 System Management Concepts: Operating System and Devices* for a list of accounting commands that can be run automatically or entered from the keyboard.

For more information about the Accounting System, the preparation of daily and monthly reports, and the accounting files, see the Accounting Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

AIX Performance Monitoring and Tuning Commands in *AIX Versions 3.2 and 4 Performance Tuning Guide*.

tip Command

Purpose

Connects to a remote system.

Syntax

tip [**−v**] [*−BaudRate*] { *SystemName* | *PhoneNumber* }

Description

The **tip** command connects to a remote system and allows you to work on the remote system as if logged in directly.

Either the *SystemName* parameter or the *PhoneNumber* parameter is required. The *SystemName* parameter specifies the name of a remote system to be contacted. The remote system must be defined in the **/etc/remote** file, or in the file specified by the **REMOTE** environment variable. The *PhoneNumber* parameter specifies the number to dial over a modem connection.

When the **tip** command is invoked with the *SystemName* parameter, it searches the **remote** file for an entry beginning with that system name. When the command is invoked with the *PhoneNumber* parameter, it searches the **remote** file for an entry of the form **tip***BaudRate*, where *BaudRate* is the baud rate for the connection. If the *−BaudRate* flag is not used, the **tip** command looks for a `tip1200` entry, since 1200 is the default baud rate.

The actions of the **tip** command can be controlled using flags, escape signals and variables. The **tip** command reads the **/etc/remote** file to find out how to contact a remote system and discover the escape−send sequence to use when communicating with that system. In addition, the command may check the **/etc/phones** file to find out a phone number for the remote system.

A **tip** user can create an individual remote file in the format of the **/usr/lib/remote−file** file, and then specify the file to use with the **REMOTE** environment variable. A user can also create an individual phones file in the format of the **/usr/lib/phones−file** file, and then specify the file to use with the **PHONES** environment variable. The **tip** command does not read the **/usr/lib/remote−file** file or **/usr/lib/phones−file** file by default, however. The default files that the **tip** command uses are the **/etc/remote** file and **/etc/phones** file.

A **tip** user can create a **$HOME/.tiprc** file to specify initial settings for the **tip**variables. In addition, settings made in the remote file, the phones file, and the **.tiprc** file can be overridden by using escape signals while **tip** is running. Escape signals can also be used, for instance, to start and stop file transfers or interrupt a connection to remote system.

The **tip** command uses lock files in the **/etc/locks** directory to lock devices against multiple access and to prevent multiple users from logging in on the same system.

When the **tip** command prompts for a response, edit the line as you type using the standard keys. Entering ~. (tilde, period) in response to a prompt, or pressing the Interrupt key, will abort the **tip** dialog and return you

to the remote system.

You can use the **tip** command to transfer files to and from the remote system. You can use **tip** command escape signals to start and stop the file transfers. Several **tip** command variables work together to control file transfers.

File transfers normally use tandem mode to control the flow of data. If the remote system does not support tandem mode, set the **echocheck** variable to `on` to cause the **tip** command to synchronize with the remote system after transmitting each character. When transferring files with the ~< and ~> escape signals, use the **eofread** and **eofwrite** variables to specify the end of a file when writing, and recognize the end of a file when reading.

If the **verbose** variable is set `on`, the **tip** command:

- Writes a running count of the number of lines transferred during a file transfer.
- Writes messages indicating its actions as it dials a phone number.

You can use scripting to record the conversations you have with the **tip** command. Use the **script** variable to start scripting.

Notes:

1. Only a user with root user authority can change the **dialtimeout** variable.
2. Although any user can specify a host at the command line, only the root user can change the **host** variable setting after the **tip** command has been started. However, this does not change the system to which the **tip** command is currently connected.

Flags

−v	Displays the settings of variables as they are read from the **.tiprc** file.
−*BaudRate*	Overrides the default baud rate, which is 1200 baud.

Escape Signals

Using escape signals, you can instruct the **tip** command to terminate, log off from the remote system, and transfer files. The escape character at the beginning of a line indicates an escape signal. The default escape character is a ~ (tilde). The character can be changed using the **escape** variable. All other typed characters are transmitted directly to the remote system. The **tip** command recognizes the following escape signals:

~^D~	Terminates the connection and exits. You may still be logged in on the remote system; if so, you can issue another **tip** command to reconnect to that remote system.
~c [*Directory*]	Changes, on the local system, to the directory specified by the *Directory* variable. If you do not include the *Directory* variable, the **tip** command changes to your home directory.
~!	Escapes to a shell on the local system. When you exit from the shell, you return to the **tip** command.
~>	Copies a file from the local system to the remote system. The **tip** command prompts you for the name of the local file.
~<	Copies a file from the remote system to the local system. The **tip** command prompts you for the name of the remote file.

A tip file download will only download the file until one of the EOF characters listed in the **eofread** command cariable is encountered. If one of these characters is not encountered, then the file copy will not succeed.

When downloading a file with the ~< signal, the user will be prompted for a local file name. The user may respond with any valid writeable file name. When prompted for the remote command, the user should append the EOF character to the end of the file being read.

This signal can be used as shown in the following example:

```
List command for remote system? echo "\04" | cat /etc/passwd
```

This example assumes that the character 0x4 is present in the tip eofread variable. The best way of ensuring that this character exists in the variable is to assign it in the usr's .tiprc file, which should reside in the user's home directory.

To accomplish this, the following command can be issued:

```
echo"eofread=\04" >> ~/.tiprc
```

~p*Source* [*Dest*] Sends (puts) the *Source* file to a remote UNIX host system, using the **cat** command to copy the *Source* file to the *Dest* file. If the *Dest* file name is not specified, the **cat** command uses the name of the *Source* file. If the *Dest* file exists on the remote host, it will be replaced by the *Source* file. This signal is a UNIX−specific version of the ~> signal.

~t*Source* [*Dest*] Transfers (takes) the *Source* file from a remote UNIX host system to the local system, using the **cat** command to copy the *Source* file to the *Dest* file on the local system. If the *Dest* file name is not specified, the **cat** command uses the name of the *Source* file. If the *Dest* file exists on the local system, it will be replaced by the *Source* file. This signal is a UNIX−specific version of the ~< signal.

~| Pipes the output of a remote command to a local process. The command string sent to the local system is processed by the shell.

A remote pipe will only succeed if the data from the remote pipe is terminated by one of the eof characters listed in the eofread tip command variable. If one of these characters is not encountered, then the output pipe will not succeed.

When piping remote output with the ~| signal, the user will be prompted for a local command name. The user may respond with any valid command name. When prompted for the remote command, the user should append the EOF character to the end of the file being read.

This signal can be used as shown in the following example:

```
Local command? cat
List command for remote system? echo
"asdfasdfasdfasdf\04"
```

This example assumes that the character 0x4 is present in the tip eofread variable. The best way of ensuring that this character exists in the variable is to assign it in the usr's .tiprc file, which should reside in the user's home directory.

To accomplish this, the following command can be issued:

```
echo"eofread=\04" >> ~/.tiprc
```

~$ Pipes the output of a local process to the remote system. The command string sent to the remote system is processed by the shell.

~# Sends a BREAK signal to the remote system.

~s { *Variable*=*Value* | [**!**]*BoolVariable* | **all** | *Variable***?** }

	Sets or queries the **tip** command variables. To change the value of a non−Boolean variable, enter the variable name or abbreviation, followed by an = (equal sign), followed by the new value. For example, enter `~s rc=^U` to change the character used to turn uppercase conversion on or off (the **raisechar** variable). To change the value of a Boolean variable, enter the variable name or abbreviation. To reset the variable to its default value, enter an **!** (exclamation point) in front of the name. For example, enter `~s !ec` to reset the **echocheck** variable to its default value. To display all variables readable by the user, specify **all** as an argument to the ~s signal. You may also request the display of a specific variable by attaching a **?** (question mark) to the variable name. For example, enter the command `~s eol?` to display the current end−of−line string (the **eol** variable).
~^Z	Stops the **tip** command. The ~^Z signal is only available with job control.
~^Y	Stops the local portion of the **tip** command. The remote portion, which displays the output from the remote system, continues to run. The ~^Y signal is only available with job control.
~?	Displays a list of the escape signals.

Variables

The **tip** command uses variables that control its operation. These variables may be numeric, string, character, or Boolean values. Some of these variables can be changed by any user who can run the **tip** command. However, the following variables can be changed only by a user with root user authority: the **baudrate** variable and the **dialtimeout** variable.

Variables may be initialized at run time in the **$HOME/.tiprc** file. Additionally, you can display and set the variables while already running the **tip** command by using the ~s escape signal.

Variables may be numeric, string, character, or Boolean values. To set a non−Boolean variable, enter the variable name or abbreviation followed by an = (equal sign) and the value. For example, enter either `~s host=zeus` or `~s ho=zeus` to change the **host** name to `zeus`. In the **.tiprc** file, enter `host=zeus` or `ho=zeus`.

To change the value of a Boolean variable, enter the variable name or abbreviation as an argument to the ~s signal or on a line of the **.tiprc** file. To reset the variable to its default value, enter an ! (exclamation point) in front of the name. For example, enter `~s !echocheck` to reset the **echocheck** variable to its default value while running the **tip** command.

Following are the common variables, their types, abbreviations, and default values.

Variable	Type	Abbreviation	Default Value
beautify	Boolean	**be**	Instructs the **tip** command to discard unprintable characters when a session is being scripted. Does not discard characters specified with the **exceptions** variable. The default setting is `on`.
baudrate	Numeric	**ba**	Reflects the baud rate of the connection. Changing the value of this variable will *not* change the current baud setting of the connected tty device.

dialtimeout	Numeric	**dial**	Specifies the time in seconds that the **tip** command waits for a connection when dialing a phone number. The default is 60 seconds. The **dialtimout** setting can be changed only by someone with root user authority.
echocheck	**Boolean**	**ec**	Instructs the **tip** command to synchronize with the remote system during a file transfer by awaiting the echo of the last character transmitted before transmitting the next character. The default setting is `off`.
eofread	String	**eofr**	Specifies the set of characters that signifies end−of−transmission during a remote−to−local (~< or ~t) file transfer.
eofwrite	String	**eofw**	Specifies the string that is sent to indicate the end of a transmission during a local−to−remote (~> or~p) file transfer.
eol	String	(none)	Specifies the string that indicates the end of a line. The **tip** command recognizes escape signals only when they follow an end−of−line string.
escape	Character	**es**	Specifies the character prefix for escape signals. The default is ~ (tilde).
etimeout	Numeric	**et**	Specifies the time to wait for a response when the **echocheck** variable is set `on`. If the echo is not received within the designated time, the file transfer is discontinued. The default time is 28 seconds.
exceptions	String	**ex**	Specifies the set of characters that should not be discarded even when the **beautify** switch is set to `on`. The `\t\n\f\b` string is the default.
force	Character	**fo**	Specifies the character that is used to force literal data transmissions during binary transfers. The `^P` character is the default. Literal data transmissions are off until the user types the character specified by the **force** variable.
framesize	**Numeric**	**fr**	Specifies the number of bytes to buffer between files system writes when receiving files from the remote system.
host	String	**ho**	Specifies the name of the remote system to which you were connected when the **tip** command was invoked. This variable cannot be changed.
halfduplex	Boolean	**hdx**	Toggles Half−duplex mode. The default setting is `off`.
localecho	Boolean	**le**	Toggles the Local−echo mode. The default setting is `off`.
log	String	(none)	Defines the file used to log dial−outs with the **tip** command. The default file is the **/var/spool/uucp/.Admin/aculog** file. The log file can be changed only by someone with root authority.
parity	String	**par**	Defines the parity for file transfers. Defaults to the following string: `no parity, 8 data bits`
phones	String	(none)	Specifies the name of the user's phone file. The file can have any valid file name and must be set up in the format of the **/usr/lib/phones−file** file. The default is the **/etc/phones** file. If a file is specified with the **PHONES** environment variable, it is used in place of (not in addition to) the **/etc/phones** file.

prompt	Character	**pr**	Specifies the character that indicates the end of the line on the remote host. This character is used to synchronize during data transfers. The **tip** command counts lines transferred during a file transfer, based on the number of times it receives the **prompt** character. The `\n` character is the default.
raise	Boolean	**ra**	When set to `on`, instructs the **tip** command to convert all lowercase letters to uppercase before transmitting them to the remote system. The default setting is `off`.
raisechar	Character	**rc**	Specifies a character that is used to toggle uppercase conversion. The `^A` character is the default.
rawftp	Boolean	**raw**	If the **rawftp** variable is set to `on`, data is transmitted over the connection during a file transfer with no additional processing carried out. That is, when sending files, line−feeds are not mapped to line−feed/carriage carried out.
record	String	**rec**	Specifies the name of the file in which the **tip** command records the session script. The **tip.record** file is the default. The **tip** command places the file in the user's current directory on the local system.
remote	String	(none)	Specifies the name of the user's remote system definition file. The file can have any valid file name and must be set up in the format of the **/usr/lib/remote−file** file. The default is the **/etc/remote** file. If a file is specified with the **REMOTE** environment variable, it is used in place of (not in addition to) the **/etc/remote** file.
script	Boolean	**sc**	When the **script** switch is set `on`, the **tip** command records everything transmitted by the remote system in a file on the local system. The file name is specified by the **record** variable. If the **beautify** switch is set to `on`, only printable ASCII characters (those between 040 and 0177) will be recorded in the script file. The **exceptions** variable specifies unprintable characters that will be recorded even if the **beautify** switch is set to `on`. The default setting for the **script** switch is `off`.
tabexpand	Boolean	**tab**	Causes the **tip** command to expand tab characters to eight spaces during file transfers. The default setting is `off`.
verbose	Boolean	**verb**	When the **verbose** switch is set `on`, the **tip** command prints messages while dialing, shows the current number of lines transferred during a file transfer, and displays other status information about the connection. The default setting is `on`.
SHELL	String	(none)	Specifies the type of shell to use for the ~! signal. The default value is **/usr/bin/sh** or is taken from the environment.
HOME	String	(none)	Specifies the home directory to use for the ~c signal. The default value is taken from the environment.

Examples

1. To specify a baud rate when making a direct connection, enter:

   ```
   tip −300 hera
   ```

   This instructs the **tip** command to use baud rate of `300` when contacting remote system `hera`.

2. To use a modem to connect to a remote system, enter:

   ```
   tip 9,343-2132
   ```

   The **tip** command connects the local system to the remote system reached by the telephone number `343-2132`, after dialing a `9` to reach an outside line.

3. To connect directly to a remote system and display the variables, enter:

   ```
   tip -v hera
   ```

   The **-v** flag causes the **tip** command to display the values of the variables as it reads them from the **$HOME/.tiprc** file. If the **.tiprc** file contains the following settings:

   ```
   sc
   be
   rec=/home/jimk/callout
   ```

   then output from the **-v** flag is as follows:

   ```
   set script
   set beautify
   set record=/home/jimk/callout
   ```

Files

/usr/bin/tip	Contains the **tip** command.
/etc/locks/*	Contains lock files that prevent multiple uses of devices and multiple calls to systems.
/etc/remote	Contains system descriptions for the **tip** command. **Note:** If the **remote** variable or the **REMOTE** environment variable is set, that file is used instead.
/usr/lib/remote-file	Contains sample **remote** file. **Note:** If the **remote** variable or the **RECORD** environment variable is set, that file is used instead.
/etc/phones	Contains the telephone number database for the **tip** command. **Note:** If the **phones** variable or the **PHONES** environment variable is set, that file is used instead.
/usr/lib/phones-file	Contains the telephone number data base for the **tip** command. **Note:** If the **phones** variable or the **PHONES** environment variable is set, that file is used instead.
$HOME/.tiprc	Defines initial settings for the **tip** command variables.
tip.record	Contains the **tip** command scripts. By default, the file is stored in the current directory. The user can change the file name and directory using the **record** variable.

Related Information

The **cu** command, **uucp** command.

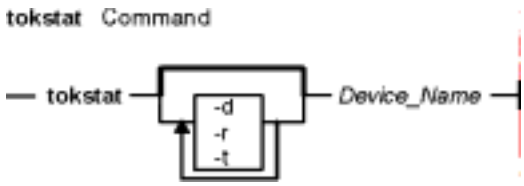
The tip Overview for System Management in *AIX Version 4.3 System Management Guide: Communications and Networks*.

tokstat Command

Purpose

Shows token−ring device driver and device statistics.

Syntax



tokstat [**−d−r−t**] *Device_Name*

Description

The **tokstat** command displays the statistics gathered by the specified Token−Ring device driver. The user can optionally specify that the device−specific statistics be displayed in addition to the device driver statistics. If no flags are specified, only the device driver statistics are displayed.

This command is also invoked when the **netstat** command is run with the **−v** flag. The **netstat** command does not issue any **tokstat** command flags.

If an invalid *Device_Name* is specified, the **tokstat** command produces an error message stating that it could not connect to the device.

Flags

−d Displays all the device driver statistics, including the device−specific statistics.

−r Resets all the statistics back to their initial values. This flag can only be issued by privileged users.

−t Toggles debug trace in some device drivers.

Parameters

Device_Name The name of the Token−Ring device, for example, t**ok0**.

Statistic Fields

> **Note:** Some adapters may not support a specific statistic. The value of non−supported statistic fields is always 0.

The statistic fields displayed in the output of the **toktstat** command and their descriptions are:

Title Fields

`Device Type`	Displays the description of the adapter type.
`Hardware Address`	Displays the Token−Ring network address currently used by the device.
`Elapsed Time`	Displays the real time period which has elapsed since the last time the statistics were

reset. Part of the statistics may be reset by the device driver during error recovery when a hardware error is detected. There will be another Elapsed Time displayed in the middle of the output when this situation has occurred in order to reflect the time differences between the statistics.

Transmit Statistics Fields

Packets	The number of packets transmitted successfully by the device.
Bytes	The number of bytes transmitted successfully by the device.
Interrupts	The number of transmit interrupts received by the driver from the adapter.
Transmit Errors	The number of output errors encountered on this device. This is a counter for unsuccessful transmissions due to hardware/network errors.
Packets Dropped	The number of packets accepted by the device driver for transmission which were not (for any reason) given to the device.
Max Packets on S/W Transmit Queue	The maximum number of outgoing packets ever queued to the software transmit queue.
S/W Transmit Queue Overflow	The number of outgoing packets which have overflowed the software transmit queue.
Current S/W+H/W Transmit Queue Length	The number of pending outgoing packets on either the software transmit queue or the hardware transmit queue.
Broadcast Packets	The number of broadcast packets transmitted without any error.
Multicast Packets	The number of multicast packets transmitted without any error.
Timeout Errors	The number of unsuccessful transmissions due to adapter reported timeout errors.
Current SW Transmit Queue Length	The number of outgoing packets currently on the software transmit queue.
Current HW Transmit Queue Length	The number of outgoing packets currently on the hardware transmit queue.

Receive Statistics Fields

Packets	The number of packets received successfully by the device.
Bytes	The number of bytes received successfully by the device.
Interrupts	The number of receive interrupts received by the driver from the adapter.
Receive Errors	The number of input errors encountered on this device. This is a counter for unsuccessful reception due to hardware/network errors.
Packets Dropped	The number of packets received by the device driver from this device which were not (for any reason) given to a network demuxer.
Bad Packets	The number of bad packets received (saved) by the device driver.
Broadcast Packets	The number of broadcast packets received without error.
Multicast Packets	The number of multicast packets received without error.
Receive Congestion Errors	The number of incoming packets dropped by the hardware due to a no

resource error.

General Statistics Fields

`No mbuf Errors`	The number of times mbufs were not available to the device driver. This usually occurs during receive operations when the driver must obtain mbuf buffers to process inbound packets. If the mbuf pool for the requested size is empty, the packet will be discarded. The **netstat −m** command can be used to confirm this.
`Lobe Wire Faults`	The number of times the adapter detected an open or short circuit in the lobe data path (for example, the cable is unplugged).
`Abort Errors`	The number of times the adapter had problems transmitting.
`AC Errors`	The number of times the adapter received more than one AMP (Active Monitor Present) or SMP (Standby Monitor Present) frame which had the address recognized and frame copied bits set to zero. This indicates a problem with neighbor notification. Every station learns and remembers who its Nearest Active Upstream Neighbor (NAUN) is from AMP and SMP frames. When a station reports a problem, it also reports who its NAUN is. This helps to define the *fault domain*.
`Burst Errors`	The number of times the adapter detected that the polarity of the signal did not switch when necessary.
`Frame Copy Errors`	The number of times the adapter detected that a frame with its specific address has been copied by another adapter.
`Frequency Errors`	The number of times the adapter detected that the frequency of the incoming signal differs from the expected frequency by more than that allowed by the IEEE 802.5 standard. Check the active monitor responsible for master clocking of the ring and compensating for frequency jitter.
`Hard Errors`	The number of times the adapter either transmitted or received a beacon MAC frame.
`Internal Errors`	The number of times the adapter had an internal error.
`Line Errors`	The number of times the adapter detected an invalid character in a frame or token.
`Lost Frame Errors`	The number of times the adapter transmitted a frame and failed to receive it back.
`Only Station`	The number of times the adapter sensed that it is the only adapter on the ring.
`Token Errors`	The number of times the adapter, acting as an active monitor, detected that the token got lost. This may be due to ring reconfiguration. If this occurs often, check to see if other soft errors indicate a specific problem.
`Remove Received`	The number of times the adapter received a Remove Ring Station MAC frame request.
`Ring Recovered`	The number of times the ring is purged and recovered back into a normal operating state.
`Signal Loss Errors`	The number of times the adapter detected the absence of a receive signal.
`Soft Errors`	The number of times the adapter detected a soft error (recoverable by the MAC layer protocols).
`Transmit Beacon Errors`	The number of times the adapter transmitted a beacon frame.
`Driver Flags`	The device driver internal status flags currently turned on.

Device Specific Statistics Fields

This part of the display may be different for each type of adapter. It may contain adapter−specific information and some extended statistics that were not included in the generic statistics. Some adapters may not have any device−specific statistics. Some fields that may be listed in this section are:

`ARI/FCI Errors`	ARI/FCI mismatch is also referred to as receiver congestion. If an adapter gets an address match on a frame going by on the ring, Address Recognized Indication(ARI), and has no place into which to copy the frame, Frame Copied Indication(FCI), an ARI/FCI mismatch has occurred. The adapter will turn on the ARI bits but will not turn on the FCI bits in the FS byte at the end of the frame as it goes by. In other words, the adapter saw a frame that was to be received but, could not receive it because the receive buffers have been depleted. Two seconds later the adapter will send a Report Soft Error MAC frame indicating a receiver congestion error.
`DMA Bus Errors`	The number of times the adapter completed a DMA transfer and detected a bus error.
`DMA Parity Errors`	The number of times the adapter completed a DMA transfer and detected a parity error.
`Receive Overruns`	The number of times the adapter receive FIFO was full when the adapter tried to receive a frame.
`Receive Underruns`	The number of times the adapter transmit FIFO was empty before the end of frame symbol was detected.
`Number of read log commands issued`	The number of times an adapter error counter overruns (reached 255) and the device driver issues a read log command to read (and reset) the error counters.

Examples

1. To display the device driver statistics for **tok0**, enter:

   ```
   tokstat tok0
   ```

   This produces the following output:

   ```
   TOKEN−RING STATISTICS (tok0) :
   Device Type: Token−Ring High−Performance Adapter (8fc8)
   Hardware Address: 10:00:5a:4f:26:c1
   Elapsed Time: 0 days 0 hours 8 minutes 33 seconds

   Transmit Statistics:                  Receive Statistics:
   −−−−−−−−−−−−−−−−−−−                   −−−−−−−−−−−−−−−−−−
   Packets: 191                          Packets: 8342
   Bytes: 17081                          Bytes: 763227
   Interrupts: 156                       Interrupts: 8159
   Transmit Errors: 0                    Receive Errors: 0
   Packets Dropped: 0                    Packets Dropped: 0
   Max Packets on S/W Transmit Queue: 17 Bad Packets: 0
   S/W Transmit Queue Overflow: 0
   Current S/W+H/W Transmit Queue Length: 0
   ```

```
Broadcast Packets: 1                Broadcast Packets: 8023
Multicast Packets: 0                Multicast Packets: 0
Timeout Errors: 0                   Receive Congestion Errors: 0
Current SW Transmit Queue Length: 0
Current HW Transmit Queue Length: 0


General Statistics:
-------------------
No mbuf Errors: 0                   Lobe Wire Faults: 0
Abort Errors: 0                     AC Errors: 0
Burst Errors: 0                     Frame Copy Errors: 0
Frequency Errors: 0                 Hard Errors: 0
Internal Errors: 0                  Line Errors: 0
Lost Frame Errors: 0                Only Station: 0
Token Errors: 0                     Remove Received: 0
Ring Recovered: 0                   Signal Loss Errors: 0
Soft Errors: 0                      Transmit Beacon Errors: 0
Driver Flags: Up Broadcast Running
   AlternateAddress ReceiveFunctionalAddr
```

2. To display the token−ring device driver statistics and the Token−Ring device−specific statistics for **tok0**, enter:

   ```
   tokstat −d tok0
   ```

   This produces the following output:

```
TOKEN−RING STATISTICS (tok0) :
Device Type: Token−Ring High−Performance Adapter (8fc8)
Hardware Address: 10:00:5a:4f:26:c1
Elapsed Time: 0 days 2 hours 48 minutes 38 seconds


Transmit Statistics:                Receive Statistics:
--------------------                -------------------
Packets: 389                        Packets: 153216
Bytes: 42270                        Bytes: 14583150
Interrupts: 354                     Interrupts: 151025
Transmit Errors: 0                  Receive Errors: 0
Packets Dropped: 0                  Packets Dropped: 0
Max Packets on S/W Transmit Queue:17 Bad Packets: 0
S/W Transmit Queue Overflow: 0
Current S/W+H/W Transmit Queue Length: 0


Broadcast Packets: 1                Broadcast Packets: 152642
Multicast Packets: 0                Multicast Packets: 0
Timeout Errors: 0                   Receive Congestion Errors: 0
Current SW Transmit Queue Length: 0
Current HW Transmit Queue Length: 0


General Statistics:
-------------------
No mbuf Errors: 0                   Lobe Wire Faults: 0
Abort Errors: 0                     AC Errors: 0
Burst Errors: 0                     Frame Copy Errors: 0
Frequency Errors: 0                 Hard Errors: 0
Internal Errors: 0                  Line Errors: 0
Lost Frame Errors: 0                Only Station: 0
Token Errors: 0                     Remove Received: 0
Ring Recovered: 0                   Signal Loss Errors: 0
Soft Errors: 0                      Transmit Beacon Errors: 0
Driver Flags: Up Broadcast Running
   AlternateAddress ReceiveFunctionalAddr
```

```
Token-Ring High-Performance Adapter (8fc8) Specific Statistics:
------------------------------------------------------------
DMA Bus Errors: 0                       DMA Parity Errors: 0
ARI/FCI Errors: 0
```

Related Information

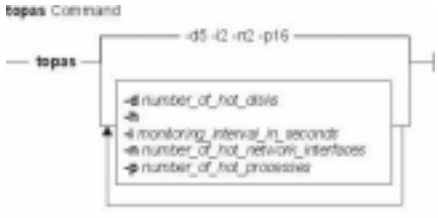
The **atmstat** command, **entstat** command, **fddistat** command, **netstat** command.

topas Command

Purpose

Reports selected local system statistics.

Syntax



topas [**−d***number_of_hot_disks*] [**−h**] [**−i***monitoring_interval_in_seconds*] [**−n***number_of_hot_network_interfaces*] [**−p***number_of_hot_processes*]

Description

The **topas** command reports selected statistics about the activity on the local system. The command uses the curses library to display its output in a format suitable for viewing on an 80x25 character based display or in a window of at least the same size on a graphical display. The **topas** command requires the **perfagent.tools** fileset to be installed on the system.

If the **topas** command is invoked without flags, it runs as if invoked with the command line:

```
topas −d5 −i2 −n2 −p16
```

The program extracts statistics from the system with an interval specified by the *monitoring_interval_in_seconds* argument. The output, as shown below, consists of a two fixed parts and a variable section. The top seven lines at the left of the display shows the name of the system **topas** runs on, the date and time of the last observation, and the monitoring interval. Following this is a section that lists the cpu utilization in numeric and block−graph format.

The second fixed part fills the rightmost 32 positions of the display. It contains five subsections of statistics, as follows:

EVENTS/QUEUES Displays the per−second frequency of selected system−global events and the average size of the thread run− and wait queues:

Cswitch
The number of context switches per second over the monitoring interval.

Syscalls
The total number of system calls per second executed over the monitoring interval.

Reads
The number of read system calls per second executed over the monitoring interval.

Writes
The number of write system calls per second executed over the monitoring interval.

Forks

The number of fork system calls per second executed over the monitoring interval.

Execs
The number of exec system calls per second executed over the monitoring interval.

Runqueue
The average number of threads that were ready to run but were waiting for a processor to become available.

Waitqueue
The average number of threads that were waiting for paging to complete.

FILE/TTY Displays the per−second frequency of selected file and tty statistics.

Readch
The number of bytes read per second through the **read** system call over the monitoring interval.

Writech
The number of bytes written per second through the **write** system call over the monitoring interval.

Rawin
The number of raw bytes read per second from TTYs over the monitoring interval.

Ttyout
The number of bytes written to TTYs per second over the monitoring interval.

Igets
The number of calls per second to the inode lookup routines over the monitoring interval.

Namei
The number of calls per second to the pathname lookup routines over the monitoring interval.

Dirblk
The number of directory blocks scanned per second by the directory search routine over the monitoring interval.

PAGING Displays the per−second frequency of paging statistics.

Faults
Total number of page faults taken per second over the monitoring interval. This includes page faults that do not cause paging activity.

Steals
Physical memory 4K frames stolen per second by the virtual memory manager over the monitoring interval.

PgspIn
Number of 4K pages read from paging space per second over the monitoring interval.

PgspOut
Number of 4K pages written to paging space per second over the monitoring interval.

PageIn
Number of 4K pages read per second over the monitoring interval. This includes paging activity associated with reading from file systems. By subtracting **PgspIn** from this value you get the number of 4K pages read from file systems per second over the monitoring interval.

PageOut
Number of 4K pages written per second over the monitoring interval. This includes paging activity associated with writing to file systems. By subtracting **PgspOut** from this value you get the number of 4K pages written to file systems per second over the monitoring interval.

Sios

The number of I/O requests per second issued by the virtual memory manager over the monitoring interval.

MEMORY Displays the real memory size and the distribution of memory in use.

Real,MB
The size of real memory in megabytes.

% Comp
The percentage of real memory currently allocated to computational page frames. Computational page frames are generally those that are backed by paging space.

% Noncomp
The percentage of real memory currently allocated to non−computational frames. Non−computational page frames are generally those that are backed by file space, either data files, executable files, or shared library files.

% Client
The percentage of real memory currently allocated to cache remotely mounted files.

PAGING SPACE Display size and utilization of paging space.

Size,MB
The sum of all paging spaces on the system, in megabytes.

% Used
The percentage of total paging space currently in use.

% Free
The percentage of total paging space currently free.

The variable part of the **topas** display can have one, two, or three subsections. If more than one appears, they are always shown in the following order:

- Network Interfaces
- Physical Disks
- Processes

Network Interfaces Lists the selected number of network interfaces. The interfaces are ordered after the activity over the monitoring interval. The interface that transferred most bytes (sum of bytes read and written) over the interval is listed first. For each network interface the following fields are displayed:

Interf
The name of the network interface.

KBPS
The total throughput in megabytes per second over the monitoring interval. This field is the sum of kilobytes received and kilobytes sent per second.

I−Pack
The number of data packets received per second over the monitoring interval.

O−Pack
The number of data packets sent per second over the monitoring interval.

KB−In
The number of kilobytes received per second over the monitoring interval.

KB−Out
The number of kilobytes sent per second over the monitoring interval.

Physical Disks Lists the selected number of physical disks. The disks are ordered after the activity over the monitoring interval. The interface that was most busy over the interval is listed first. For each disk the following fields are displayed:

Disk
The name of the physical disk.

Busy%
Indicates the percentage of time the physical disk was active (bandwidth utilization for the drive).

KBPS

The number of kilobytes read and written per second over the monitoring interval. This field is the sum of **KB−Read** and **KB−Read**.

TPS

The number of transfers per second that were issued to the physical disk. A transfer is an I/O request to the physical disk. Multiple logical requests can be combined into a single I/O request to the disk. A transfer is of indeterminate size.

KB−Read

The number of kilobytes read per second from the physical disk.

K −Writ

The number of kilobytes written per second to the physical disk.

Processes Lists the selected number of processes or as many as will fit on the display. The processes are ordered after their cpu usage over the monitoring interval. The process that consumed the most cpu over the interval is listed first. For each process the following fields are displayed:

Name

The name of the executable program executing in the process. The name is stripped of any pathname and argument information and truncated to 9 characters in length.

Process ID

The process ID of the process.

% CPU Utilization

The average cpu utilization of the process over the monitoring interval. The first time a process is shown, this value will be the average cpu utilization over the lifetime of the process.

Paging Space Used

The size of the paging space allocated to this process. This can be considered an expression of the footprint of the process but does not include the memory used to keep the executable program and any shared libraries it may depend on.

Process Owner

The user name of the user that owns the process.

Sample Output

The following is an example of the display generated by the **topas** command:

```
Topas Monitor for host:    niller                EVENTS/QUEUES    FILE/TTY
Mon Mar  1 07:00:27 1999   Interval:  2          Cswitch     383  Readch   504233
                                                 Syscall    2421  Writech   86445
Kernel   35.0   |##########                  |  Reads       254  Rawin         0
User     39.5   |###########                 |  Writes       44  Ttyout      354
Wait     22.5   |######                      |  Forks         7  Igets         8
Idle      3.0   |#                           |  Execs         7  Namei       281
                                                 Runqueue    2.0  Dirblk       72
Interf   KBPS   I−Pack  O−Pack   KB−In  KB−Out  Waitqueue   1.0
tr0        0.0     0.5     0.5     0.0     0.0
lo0        0.0     0.0     0.0     0.0     0.0  PAGING           MEMORY
                                                 Faults     1901  Real,MB     384
Disk    Busy%     KBPS     TPS KB−Read KB−Writ  Steals        0  % Comp     15.0
hdisk0   27.5    110.0    25.5     0.0   110.0  PgspIn        0  % Noncomp  42.3
hdisk1    0.0      0.0     0.0     0.0     0.0  PgspOut       0  % Client    0.0
hdisk2    0.0      0.0     0.0     0.0     0.0  PageIn        0
                                                 PageOut      27  PAGING SPACE
xlcentry (56328)  5.0% PgSp: 0.5mb nchris        Sios         25  Size,MB     512
X        (2692)   4.0% PgSp:30.8mb root                          % Used     25.5
cc       (56794)  2.0% PgSp: 0.1mb nchris                       % Free     74.5
i4lmd    (21418)  1.5% PgSp: 0.5mb root
java     (31246)  1.5% PgSp: 5.4mb nchris
topas    (50452)  1.5% PgSp: 0.5mb nchris          Press "h" for help screen.
make     (53914)  1.0% PgSp: 0.2mb nchris          Press "q" to quit program.
syncd    (4662)   0.0% PgSp: 0.0mb root
```

Flags

−d Specifies the maximum number of disks shown. If this number exceed the number of disks installed, the latter is used. If this argument is omitted a default of 5 (five) is assumed. If a value of zero is specified, no disk information will be displayed.

−h Displays help information in the following format:

```
usage:  topas   [−d number−of−hot−disks]
                [−h show help information]
                [−i monitoring−interval_in_seconds]
                [−n number−of−hot−network−interfaces]
                [−p number−of−hot−processes]

One−character commands:
        a       Show all
        d       Show disks (and more if space allows)
        h       Show help screen
        n       Show network interfaces (and more if space allows)
        p       Show processes (and more if space allows)
        q       Quit the program
```

−i Sets the monitoring interval in seconds. The default is 2 seconds.

−n Specifies the maximum number of network interfaces shown. If this number exceed the number of network interfaces installed, the latter is used. If this argument is omitted a default of 2 (two) is assumed. If a value of zero is specified, no network information will be displayed.

−p Specifies the maximum number of processes shown. If this argument is omitted a default of 16 (sixteen) is assumed. If a value of zero is specified, no process information will be displayed. Retrieval of process information constitutes the majority of the **topas** overhead. If process information is not required, you should always use this option to specify that you don't want process information.

Subcommands

While **topas** is running, it accepts one−character subcommands. Each time the monitoring interval elapses, the program checks for one of the following subcommands and responds to the action requested.

a Show all of the variable sections (network, disk, and process) if space allows.

d Show disk information. If the requested number of disks and the requested number of network interfaces will fit on a 25−line display, both are shown. If there is space left on a 25−line display to list at least three processes, as many processes as will fit are also displayed.

h Show the same help screen as displayed by the **−h** command line argument.

n Show network interface information. If the requested number of disks and the requested number of network interfaces will fit on a 25−line display, both are shown. If there is space left on a 25−line display to list at least three processes, as many processes as will fit are also displayed.

p Show process information. If the requested number of processes leaves enough space on a 25−line display to also display the requested number of network interfaces, those are shown. If there is also space to show the requested number of disks, those are shown as well.

q Quit the program.

Examples

1. To display four "hot" disks every 5 seconds and omit network interface and process information, enter:
   ```
   topas −i5 −n0 −p0
   ```

2. To display the five most active processes and neither network nor disk information, enter:

```
topas -p5 -n0 -d0
```

3. To run the program with default options, enter:
   ```
   topas
   ```

Files

/**usr**/**bin**/**topas** Contains the **topas** command.

Related Information

The **iostat** command, and **vmstat** command.

System Performance Measurement Interface in the *Performance Toolbox Version 1.2 and 2 for AIX: Guide and Reference*.

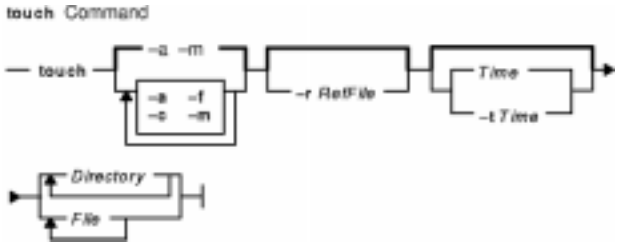
The **lchmon** sample program that ships with **perfagent.server**.

touch Command

Purpose

Updates the access and modification times of a file.

Syntax



touch [**−a**] [**−c**] [**−m**] [**−f**] [**−r** *RefFile*]
[*Time* | **−t** *Time*] { *File ...* | *Directory ...* }

Description

The **touch** command updates the access and modification times of each file specified by the *File* parameter of each directory specified by the *Directory* parameter. If you do not specify a value for the *Time* variable, the **touch** command uses the current time. If you specify a file that does not exist, the **touch** command creates the file unless you specify the **−c** flag.

The return code from the **touch** command is the number of files for which the times could not be successfully modified (including files that did not exist and were not created).

> **Note:** Any dates beyond and including the year 2038 are invalid.

Flags

−a Changes the access time of the file specified by the *File* variable. Does not change the modification time unless **−m** is also specified.

−c Does not create the file if it does not already exist. No diagnostic messages are written concerning this condition.

−f Attempts to force the touch in spite of read and write permissions on a file.

−m Changes the modification time of *File*. Does not change the access time unless **−a** is also specified.

−r *RefFile* Uses the corresponding time of the file specified by the *RefFile* variable instead of the current time.

Time Specifies the date and time of the new timestamp in the format *MMDDhhmm*[YY], where:

- *MM* Specifies the month of the year (01 to 12).
- *DD* Specifies the day of the month (01 to 31).
- *hh* Specifies the hour of the day (00 to 23).
- *mm* Specifies the minute of the hour (00 to 59).
- *YY* Specifies the last two digits of the year. If the *YY* variable is not specified, the default value is the current year.

−t *Time* Uses the specified time instead of the current time. The *Time* variable is specified in the decimal form [[*CC*]*YY*]*MMDDhhmm*[**.***SS*] where:

CC Specifies the first two digits of the year.

YY Specifies the last two digits of the year.

MM Specifies the month of the year (01 to 12).

DD Specifies the day of the month (01 to 31).

hh Specifies the hour of the day (00 to 23).

mm Specifies the minute of the hour (00 to 59).

SS Specifies the second of the minute (00 to 59).

Notes:

1. The **touch** command calls the **utime ()** subroutine to change the modification and access times of the file touched. This may cause the **touch** command to fail when flags are used if you do not actually own the file, even though you may have write permission to the file.
2. Do not specify the full path name **/usr/bin/touch** if you receive an error message when using the **touch** command.

Exit Status

This command returns the following exit values:

0 The command executed successfully. All requested changes were made.

>0 An error occurred.

Examples

1. To update the access and modification times of a file, enter:

   ```
   touch program.c
   ```

   This sets the last access and modification times of the `program.c` file to the current date and time. If the `program.c` file does not exist, the **touch** command creates an empty file with that name.

2. To avoid creating a new file, enter:

   ```
   touch −c program.c
   ```

3. To update only the modification time, enter:

   ```
   touch −m *.o
   ```

   This updates the last modification times (not the access times) of the files that end with a `.o` extension in the current directory. The **touch** command is often used in this way to alter the results of the **make** command.

4. To explicitly set the access and modification times, enter:

   ```
   touch −c −t 02171425 program.c
   ```

   This sets the access and modification dates to 14:25 (2:25 p.m.) February 17 of the current year.

5. To use the time stamp of another file instead of the current time, enter:

   ```
   touch −r file1 program.c
   ```

This gives the `program.c` file the same time stamp as the `file1` file.

6. To touch a file using a specified time other than the current time, enter:

   ```
   touch -t 198503030303.55 program.c
   ```

This gives the `program.c` file a time stamp of 3:03:55 a.m. on March 3, 1985.

Files

/usr/bin/touch Contains the **touch** command.

Related Information

The **date** command, **locale** command.

The **utime** subroutine.

Directory Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* describes the structure and characteristics of directories in the file system.

Files Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* describes files, file types, and how to name files.

File and Directory Access Modes in *AIX Version 4.3 System User's Guide: Operating System and Devices* introduces file ownership and permissions to access files and directories.

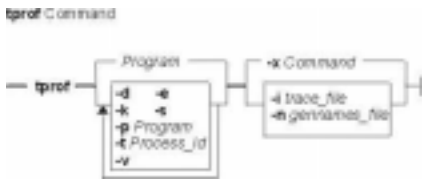
Understanding File Types in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs* introduces the commands that control files.

tprof Command

Purpose

Reports CPU usage.

Syntax



tprof *Program*

tprof { *Program* | { **−d** | **−e** | **−k** | **−p***Program* | **−s** | **−t***Process_Id* | **−v** | { [**−i***Trace_File*] [**−n***Gennames_File*] | { **−x***Command* } }

Notes:

- One of **−s**, **−k**, **−e** or **−p** flags need be specified to get profile.
- The **−x***command* flag must be the last flag issued on command line.
- The **−i** and **−x** flags may not be specified at the same time.
- The **−n** flag should only be used with the **−i** flag.

Description

The **tprof** command reports CPU usage for individual programs and the system as a whole. This command is a useful tool for anyone with a C or C++ or FORTRAN program that might be CPU−bound and who wants to know which sections of this program are most heavily using the CPU. The **tprof** command also reports the fraction of time the CPU is idle. These reports can be useful in determining CPU usage in a global sense.

Note: Only the root user and members of the security group should have execute (x) access to this command.

The **tprof** command specifies the user program to be profiled, executes the user program, and then produces a set of files containing reports. The user specifies the name of the program to be profiled, or alternatively, the name of the program to be profiled and a command line to be executed. Both the *Program* and *Command* variables must be executable.

In the AIX operating system, an interrupt occurs periodically to allow a "housekeeping" kernel routine to run. This housekeeping occurs 100 times per second. When the **tprof** command is invoked, the housekeeping kernel routine records the process ID and the address of the instruction executing when the interrupt occurred. With both the instruction address and process ID, the **tprof** analysis routines can charge CPU time to processes and threads, to subprograms, and even to source lines of programs. Charging CPU time to source program lines is called microprofiling.

> **Note:** Microprofiling can only be performed on code compiled with the **−g** option. It is also necessary for the source code to reside in the working directory, that is, the directory in which the **tprof** command is invoked. The **proftest** working directory is used in the examples for this command.

For subprogram−level profiling, the **tprof** command can be run without modifying your executable program. You do not have to recompile with special compiler flags or linker options. This means that you can obtain a subprogram profile of any executable module that has already been built. However, as previously stated, recompilation is required to get a microprofile.

The files containing the reports are left in the working directory. All files created by the **tprof** command are prefixed by ___ (two underscores). Prefixing with underscores allows the **tprof** command user to easily identify all **tprof**−created files. In this text, it is assumed that all executables are kept in the working directory to simplify the explanation of the tool. (To keep executables in another directory, specify the full path name of the executable file to the **tprof** command.) It is recommended that a working directory be created specifically for profiling and that copies of executables and source files be placed (or linked) in that directory. In the examples , the working directory is **proftest**.

In its simplest form, the **tprof** command is issued as follows:

```
tprof program
```

When used without arguments, the **tprof** usage statement is displayed. When using the **−x***Command* flag, one of the **−s**, **−k**, or **−p** flag must be specified to get the specific profiling report. If none of these flags are specified, only the summary report is produced. Using the **−t** flag limits the report to information about the specified process, which is named in the *Process_Id* parameter.

The **−i***Trace_File* flag allows for offline processing by **tprof** of trace data files created by the system **trace** command. The **−n** flag allows you to specify a *Gennames_File* to be used when procesing an offline file. These flags are useful when it is necessary to postprocess a trace file from a remote machine or perform the trace data collection at one time and postprocess it at another time. In this case **−n** with a *Gennames_File* must be used from the machine that the trace came from. The flags are also useful when system loading is high and trace hooks are being missed by **tprof**. The offline option relieves this problem. Trace hooks relevent to **tprof** must be collected by the **trace** command and are specified by the **trace −j** flag. The *Gennames_File* is then executed to collected additional information for **tprof**. Once the **trace** and *Ggennames_File* has executed **trcrpt −r** must be executed on the trace logfile and redirected to another file. At this point an adjusted trace logfile and a *Gennames_File* is feed into **tprof**.

Example:

```
trace −a −T 768000 −L 10000000 −o trace.out −j 000,001,002,003,005,006,234,106,
10C,134,139,00A,465

gennames > gennames.out
trcrpt −r   trace.out > trace.rpt
```

Then **tprof** with at least **−i** and **−n** flag:

```
tprof −i trace.rpt −n gennames.out −s −k −e
```

The trace hook numbers above are defined as follows:

```
HKWD_TRACE_SYNC      0x000
HKWD_TRACE_TRCON     0x001
HKWD_TRACE_TRCOFF    0x002
HKWD_TRACE_HEADER    0x003
HKWD_TRACE_LWRAP     0x005
HKWD_TRACE_TWRAP     0x006
HKWD_KERN_PROF       0x234
HKWD_KERN_DISPATCH   0x106
HKWD_KERN_IDLE       0x10C
HKWD_SYSC_EXECVE     0x134
HKWD_SYSC_FORK       0x139
HKWD_TRACE_UTIL      0x00A
```

```
HKWD_SYSC_CRTHREAD   0x465
```

Note: Review the **/usr/lpp/perfagent/README.perfagent.tools** file for the latest information on changes to the performance analysis tools.

Reports

A summary report with the suffix **.all** is always produced. If no program is specified, the report is named `__prof.all`. If the `sample` program is specified with the **−p** option, the report is named `__sample.all`. This report contains an estimate of the amount of CPU time spent in each process that was executing while the **tprof** program was monitoring the system. This report also contains an estimate of the amount of CPU time spent in each subprogram of the sample program. The summary report shows the amount of time the CPU was idle as well as the amount of time spent in the kernel. The **tprof** command reports CPU time in ticks, where 100 ticks equals 1 second.

Example 1 Report A shows an example summary report. The first section lists the CPU time estimates for each process. The second section of the report summarizes the first by combining the reports for multiple occurrences of each process into a single line.

There may be numerous processes active in the system which are not related to the user process being profiled. The number and type of processes observed vary from run to run.

A third section of the report is produced only if there are User ticks associated with the program being profiled. This section of the report notes the subprogram names and the number of ticks they collected. The source file from which the subprogram was compiled is also given as well as the length, in bytes, of the subprogram.

When microprofiling occurs, two more reports are produced only if there are User ticks associated with the program being microprofiled.The two reports are left as additional files in the working directory by the **tprof** command:

__t.routine_source
: Profiled source−statement listing, one for each routine in the source file.

__h.source
: Hottest source−statement listing, one for each source file.

Example 2 Reports A and B are the result of microprofiling an example program.

As previously stated, if microprofiling is desired, copies of the source files (usually suffixed with **.c**, **.h**, or **.f**) must be placed in the working directory. Additionally, executable files compiled with the **−g** flag (which are generated by the AIX C or FORTRAN compilers) must be placed in the working directory.

When microprofiling occurs, there is one **__t.routine_source** file for each routine in each source file compiled with the **−g** flag. The source line number and number of ticks associated with that source line are to the left of the microprofile text. Count totals for each routine are at the end of the function. Counts should not be associated with a comment line, data declaration, or macro definition.

The **tprof** command uses the following tools to create reports:

- The **cc** and **xlf** commands. The **−g** flag of the AIX C or FORTRAN compilers is required for microprofiling. Changes in the format of the object files (XCOFF) the compilers may cause the **tprof** command to fail.
- The **stripnmgenkexgenld** and **genkd** commands.
- The **trace** facility. The **tprof** command uses the AIX trace facility to obtain its address samples. Errors in the trace may cause the **tprof** command to fail.

Shared Libraries

When you run executable modules built with shared libraries, the **tprof** command reports an estimate of the aggregate CPU time spent in shared code. When invoked with the **−s** flag, the **tprof** command resolves the time spent in shared libraries to the subprogram level. For example:

```
tprof −p cwhet −v −s −x cwhet
```

provides a report on CPU time spent in shared libraries, and resolves that time to the subprogram level for each shared library incurring ticks.

> **Note:** The **−v** flag is only required if you intend to microprofile shared libraries.

Using the information gathered in this report, it is now possible to microprofile a single shared library subprogram to the source−statement level. If microprofiling of shared libraries is desired, a copy of the associated source file must be placed in the current working directory in which the **tprof** command is invoked. The object file (generated by the AIX C or FORTRAN compiler using the **−g** flag) must also be placed in the working directory. The following is an example:

```
cc −g −c malloc.c −lm
```

This example recompiles the `malloc.c` program and creates an object file (**.o**) that contains symbolic debug information used for microprofiling.

Multiple Runs to Obtain Greater Accuracy

The degree to which CPU activity can be resolved is determined by the number of samples captured and the degree to which "hot spots" dominate. While a program with a few "hot spots" can be profiled with relatively few samples, less−frequently executed sections of the program will not be visible in the profiling reports unless more samples are captured. In cases where user programs run less than a minute, there may be insufficient resolution to have a high degree of confidence in the estimates.

A simple solution is to repeatedly execute the user program or script until you achieve the degree of resolution you need. The longer a program is run, the finer the degree of resolution of the profile. If you doubt the accuracy of a profile, run the **tprof** command several times and compare the resulting profiles.

In order to profile a program or script that is executed multiple times using different sets of arguments, you must invoke the **tprof** command using a script that executes the desired program accordingly. The following is an example of a script called `lin5` that executes the `lin` program five times:

```
lin; lin; lin; lin; lin   # execute program lin 5 times
```

The **tprof** command is then invoked with:

```
tprof −p lin −x lin5
```

Another example can be found in the script `DOIT` , which executes the `lin` program *n* times (where *n* is an argument to `DOIT`).

How tprof Capabilities Differ from prof and gprof

The most significant difference between these three commands is that the **tprof** command provides microprofiling, while the **gprof** and **prof** commands do not.

The **prof** and **gprof** commands are standard, supported profiling tools on many UNIX systems, including the AIX operating system. Both the **prof** and **gprof** commands provide subprogram profiling and exact counts of the number of times every subprogram is called. The **gprof** command also provides a very useful "call graph"

showing the number of times each subprogram was called by a specific parent and the number of times each subprogram called a child. The **tprof** command provides neither subprogram call counts nor call graph information.

Like the **tprof** command, both the **prof** and **gprof** commands obtain their CPU consumption estimates for each subprogram by sampling the address counter of the user program at the rate of 100 times per second.

If only subprogram−level profiling is desired, the **prof** and **gprof** commands require users to relink their programs using a special linker flag. In order to get subprogram call counts, both the **prof** and **gprof** commands require users to both recompile and relink their programs by using the appropriate compile or link flag (**−p** or **−pg**, respectively). This may prove to be an impediment to users who do not have the ability to recompile or relink applications that need to be profiled. The **tprof** command requires recompilation (with the **−g** flag) only if source statement profiling is required. The **tprof** command will profile subprograms in any C or FORTRAN executable without recompilation.

The CPU times of user programs relinked for the **prof** command are the same as the run times of the unprofiled programs. The CPU times of user programs recompiled and relinked for the **gprof** command are increased by as little as a few percent to a factor of three or more. The CPU times of user programs profiled under the **tprof** command are the same as the run times of the unprofiled programs.

Also, the **tprof** command provides a CPU usage summary of all processes active during the execution of the profiled user program, which neither the **prof** nor **gprof** command provides.

> **Notes:** CPU activity is sampled at a rate of 100 samples per second. This means that estimates of CPU consumption for each subprogram may not be sufficiently accurate for short−running programs. However, the accuracy is sufficient for programs and processes that run several minutes or more. The system trace facility is used by the **tprof** command. Only one user can be using the system trace facility at a time. Thus, only one **tprof** command can be active in the system at a time. The **tprof** command is a CPU−activity profiler. It does not profile other system resources, such as disks or memory.

Flags

−d

This flag is not needed to microprofile shared libraries. It has been retained for compatibility purposes.

−e

Profiles the kernel extension.

−i*Trace_File*

Input trace file for offline processing.

> **Note:** The **−i** and **−x** flags may not be specified at the same time.

−k

Profiles the kernel.

−n*Gennames_File*

Specifies a *Gennames_ File* to be used inconjuction with the **−i** flag

> **Note:** Do not specify a *Gennames_File*, with the **−n** file, without providing a Trace input file, with the **−i** file.

−p Program

Profiles the user program; also microprofiles the user program if that program is compiled with the **−g** flag.

Note: In the summary report with the suffix **.all**, you can find the complete list of programs running in the system while the **tprof** command was monitoring CPU utilization. The **−p** option can only succeed if the program name you specified exactly matches a program name that appears in the summary report. If the program you want profiled runs with a full or relative path, then you must enter that full or relative path with the **−p** flag.

−*s*
Profiles shared libraries.

−*t* *Process_Id*
Constrains reporting to the specified process and its threads.

−*v*
Specifies verbose mode, which creates files associated with microprofiling (source level profiling) of shared libraries. If microprofiling of shared libraries or kernel extensions is requested, the **−v** flag produces an error warning that no object file compiled with the **−g** flag is found.

−*x* *Command*
Allows the execution of an arbitrary *Command*. Subprograms of the program specified by the **−p** flag are profiled.
If the **−x***Command* flag is omitted, the **tprof** command uses the `__trc_rpt2` trace report file in the current directory to produce its output.

> **Note:** If you do not enter a path with *Command*, then the **−x** flag searches your PATH environment variable to find the command since it uses the system() call to run *Command*. To guarantee that the correct command runs, always specify the path to the command.

If you do not enter a path with *Command*, and *Command* is the same as the *Program* to profile with the **−p** flag, you must specify *Program* without a path. This implies that *Program* is in the current directory.

If you enter a full or relative path with *Command*, and *Command* is the same as the *Program* to profile with the **−p** flag, then you must enter *Program* with the same full or relative path of *Command*.

Examples

1. Profiling a C Program

> **CAUTION:** The following C program does nothing useful. It simply wastes CPU cycles. The program is used only for illustrative purposes.

```
/* Array Incrementer -- Version 1 */
#include <stdlib.h>
#define Asize 1024
#define RowDim InnerIndex
#define ColDim OuterIndex
main()
{
   int Increment;
   int OuterIndex;
   int InnerIndex;
   int big [Asize][Asize];
   /* Initialize every byte of the array to 0x01 */
   for (OuterIndex=0; OuterIndex<Asize; OuterIndex++)
   {
      for (InnerIndex=0; InnerIndex<Asize; InnerIndex++)
         big [RowDim][ColDim] = 0x01010101;
   }
   Increment = rand();
   /* Increment every element in the array */
   for (OuterIndex=0; OuterIndex<Asize; OuterIndex++)
   {
      for (InnerIndex=0; InnerIndex<Asize; InnerIndex++)
      {
         big [RowDim][ColDim] += Increment;
         if (big [RowDim][ColDim] <0)
            printf("negative number. %d\n", big[rowdim][coldim]);
```

```
      }
   }
   printf("version 1 check num: %d\n", big[rand()%asize][rand()%asize]);
   return(0);
}
```

In this example, `version1.c` is the source file and `version1` will be the program. The source program, `version1.c`, already exists in a directory called `proftest`.

1. Go to the subdirectory `proftest`.
   ```
   cd proftest
   ```
2. Compile `version1.c`.
   ```
   xlc -o version1 version1.c
   ```
3. Profile and run the program `version1`.
   ```
   tprof version1
   ```
4. Examine the profiling output report in the `__version1.all` file.

An example of the`__version1.all` file is found in Example 1 Report. Of course, the `__version1.all` file that you create is certain to be different from the example `__version1.all` reported shown here.

Example 1 Report A

```
        Process      PID      TID    Total   Kernel     User   Shared    Other
        =======      ===      ===    =====   ======     ====   ======    =====
       version1     6624     6633      663       41      622        0        0
      rpc.lockd    11002    11267       13       13        0        0        0
             sh     6624     6633        9        5        0        4        0
          tprof    11472    13017        6        6        0        0        0
            gil     1032     1291        3        3        0        0        0
          tprof    13790    16359        3        1        2        0        0
            gil     1032     1549        2        2        0        0        0
           wait      516      517        1        1        0        0        0
          tprof     6610     6619        1        1        0        0        0
          tprof     6624     6633        1        1        0        0        0
           llbd     8478     8743        1        0        1        0        0
        rlogind    14100    14109        1        1        0        0        0
          trace    14036    16349        1        1        0        0        0
        =======      ===      ===    =====   ======     ====   ======    =====
          Total                        705       76      625        4        0
        Process     FREQ    Total   Kernel     User   Shared    Other
        =======      ===    =====   ======     ====   ======    =====
       version1        1      663       41      622        0        0
      rpc.lockd        1       13       13        0        0        0
          tprof        4       11        9        2        0        0
             sh        1        9        5        0        4        0
            gil        2        5        5        0        0        0
           wait        1        1        1        0        0        0
           llbd        1        1        0        1        0        0
        rlogind        1        1        1        0        0        0
          trace        1        1        1        0        0        0
        =======      ===    =====   ======     ====   ======    =====
          Total       13      705       76      625        4        0


   Total Ticks For version1(    USER) =    622

          Subroutine   Ticks     %           Source   Address  Bytes
        =============  ======  ======        =======   =======  =====
                .main     622    88.2      version1.c  268436032    560
```

Process names and process identification (PID) numbers can be reused during normal *AIX* operation. When a process forks another process, the forked process inherits the process name of the original process. When a process execs another process, the execed process inherits the PID of the original process.

The first section of the report shows the number of ticks consumed by, or on behalf of, each process and its thread. The columns give tick counts for Kernel Code, User Code, and Shared Code. The name of the program is `version1`, so the `version1` process (with process ID number `6624`) is the process being profiled. This process/program collected a total of `663` ticks: `41` ticks for services performed for it in the kernel's address space, `622` ticks in the user's address space, and `0` ticks in the shared library space. `Other` is a catchall category that normally is `0`.

The second section of the report summarizes the results by program, regardless of process ID. It shows the number (`FREQ`) of different processes that ran each program at some point.

The third section of the report shows the number of ticks used by each function in the executable and the percentage of the total run's CPU ticks that each function's ticks represent.

> **Note:** If there are no ticks in the User column for the program being profiled, then the third section of the report is not produced.

Sampling occurs at the rate of 100 per second. Each tick corresponds to 1/100 sec or 10 msec. Thus, in Example 1 Report A , `version1` is estimated to have consumed 6.63 seconds: 6.22 in user space and 0.41 seconds in kernel (system) space.

2. Microprofiling a C Program

The example of profiling contains summary information at the subprogram level. In this example, we provide CPU−usage profiling at the microprofiling level. Additional information must be provided to the **tprof** command in the working directory in order to make the correspondences between execution−time estimates and source statements:

- Source file `version1.c`
- Compile with the **−g** flag.

Profile:

```
tprof version1        # invoke the profiler, telling
                      # it to execute and profile "version1"
```

The output file`__version1.all` should be nearly identical to that obtained from the previous steps. Of course, the report that summarizes all processes active during the execution of the `version1` program will change. You will also find the `__t.main_version1.c` file, which contains profiled source for main in source, and the `__h.version1.c` file, which contains the "hottest" lines in the source.

> **Note:** If there are no ticks in the User column in the `__program.all` file for the program being profiled, then the `__t.routine_source` and `__h.source` files are not produced.

Example 2 Reports A and B contain results from the microprofiling example shown above. Example 2 Report A shows the contents of the `__t.main_version1.c` file. Example 2 Report B shows the contents of the `__h.version1.c` file.

Example 2 Report A

```
Ticks Profile for main in version1.c
   Line   Ticks   Source
    15     29     for (OuterIndex=0; OuterIndex<Asize; OuterIndex++)
```

```
 16      -        {
 17     32           for (InnerIndex=0; InnerIndex<Asize; InnerIndex++)
 18    209              big [RowDim][ColDim] = 0x01010101;
 19      -        }
 20      -        Increment = rand();
 21      -
 22      -        /* Increment every element in the array */
 23     52        for (OuterIndex=0; OuterIndex<Asize; OuterIndex++)
 24      -        {
 25      -           for (InnerIndex=0; InnerIndex<Asize; InnerIndex++)
 26      -           {
 27     55              big [RowDim][ColDim] += Increment;
 28     54              if (big [RowDim][ColDim] <0)
 29    191                 printf("negative number.  %d\n", big[rowdim][coldi m]);
 30                    }
 31                 }
 32                 printf("version 1 check num: %d\n", big[rand()%asize][rand( )%asize]);
 33                 return(0);
 34              }

622 total ticks for main in version1.c
```

Example 2 Report B

```
Hot Line Profile for version1.c
   Line   Ticks
    18    209
    29    191
    27     55
    28     54
    23     52
    17     32
    15     29
```

3. Profiling a FORTRAN Program with Multiple Source Files

This example uses two FORTRAN source files, `linless.f` and `daxpy.f`, which already exist in a directory called `proftest`.

1. Go to the subdirectory `proftest`.
   ```
   cd proftest
   ```
2. Compile `lin2` as the program file using the two source files, `linless.f` and `daxpy.f`.
   ```
   xlf -o lin2 linless.f daxpy.f
   ```
3. Profile and run the program `lin2`.
   ```
   tprof lin2
   ```
4. Examine the profiling output report in the `__lin2.all` file.

4. Profiling a Shell Script

Sometimes you want to profile a script of sequential executions of programs and shell commands rather than a single program. In this example, a script, `DOIT`, runs a program *n* times, with both the program and *n* specified as arguments to the script. For example, to run the program `lin` program three times, enter:

```
DOIT lin 3
In this example, the shell script, DOIT, and the FORTRAN source file called lin.f, already exis
```

1. Go to the subdirectory `proftest`.
   ```
   cd proftest
   ```
2. Compile `lin.f` as the executable file `lin` unoptimized with the **-g** flag.

```
xlf -g -o lin lin.f
```

3. Profile program `lin` using **-p** flag, and specify the name of the script and any script arguments to run using the **-x** flag.
   ```
   tprof -p lin -x DOIT lin 3
   ```

   This provides statement-level profiling

4. Examine the profiling output report in the `__lin.all` file.

Messages

If your system displays the following message:

```
/dev/systrace: device busy or trcon: TRCON:no such device
```

This means the trace facility is in use already. Stop your program and try again after typing `trcstop`, stops the trace.

Related Information

The **cc** command, **gprof** command, **prof** command, **stripnm** command.

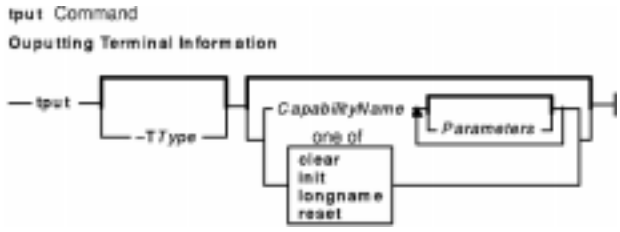
tput Command

Purpose

Queries the **terminfo** database for terminal−dependent information

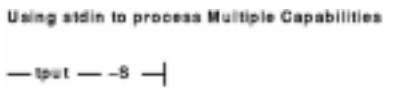
Syntax

For Outputing Terminal Information



tput [**−T***Type*] [*CapabilityName {clear, init, longname, reset}* [*Parameters...*]]

For Using stdin to Process Multiple Capabilities



tput [**−S**]

Description

The **tput** command uses the terminfo database to make terminal−dependent information available to the shell. The **tput** command outputs a string if the attribute *CapabilityName* is of type *string.* The output string is an integer if the attribute is of type *integer*. If the attribute is of type *Boolean*, the **tput** command sets the exit value (0 for TRUE, 1 for FALSE), and produces no other output.

XTERM DESCRIPTION LIMITATION

The xterm terminal description in the DEC.TI file on AIX Version 4 provides underline mode by using the SGR attribute. The SMUL and RMUL attributes are not currently defined in the XTERM terminal description on AIX Version 4. Use the more generic capability named SGR.

```
tput sgr x y
```

Where *x* is either a 1 or a 0 to turn standout mode on or off respectively, and *y* is either a 1 or a 0 to turn underline mode on or off respectively. See the article "**terminfo** file format" for more details on the SGR capability.

```
tput sgr 0 1     turn off standout; turn on underline
tput sgr 0 0     turn off standout; turn off underline
tput sgr 1 1     turn on standout; turn on underline
tput sgr 1 0     turn on standout; turn off underline
```

Flags

In addition to the capability names, the following strings are supported as arguments to the **tput** subroutine.

clear	Displays the clear screen sequence (this is also a capability name).
init	Displays the sequence that initializes the user's terminal in an implementation−dependent manner.
reset	Displays the sequence that will reset the user's terminal in an implementation−dependent manner.
longname	Displays the long name and the specified terminal (or current terminal if none specified).
−S	Uses stdin. This allow the tput to process multiple capabilities. When using the −S option, the capabilities cannot be entered on the command line. Enter ^D token finished.
−T*Type*	Indicates the type of terminal. If **−T** is not specified, the **TERM** environment variable is used for the terminal.

Exit Status

This command returns the following exit values:

0 The requested string was written successfully.
1 Unspecified.
2 Usage error.
3 No information is avaialble about the specified terminal type.
4 The specified operand is invalid.
>4 An error occurred.

Examples

1. To clear the screen for the current terminal, enter:

   ```
   tput clear
   ```

2. To display the number of columns for the current terminals, enter:

   ```
   tput cols
   ```

3. To display the number of columns for the aixterm terminal, enter:

   ```
   tput −Taixterm cols
   ```

4. To set the shell variable **bold** to the begin standout mode sequence and the shell variable **offbold** to the end standout mode sequence, enter:

   ```
   bold=`tput smso`
   ```

   ```
   offbold='tput rmso'
   ```

   Entering these commands might be followed by the following prompt:

   ```
   echo "${bold}Name: ${offbold} \c"
   ```

5. To set the exit value to indicate if the current terminal is a hardcopy terminal, enter:

   ```
   tput hc
   ```

6. To initialize the current terminal, enter:

   ```
   tput init
   ```

Files

/usr/share/lib/terminfo/?/*	Contains the terminal descriptor files.
/usr/include/term.h	Contains the definition files.

Related Information

The **stty** command.

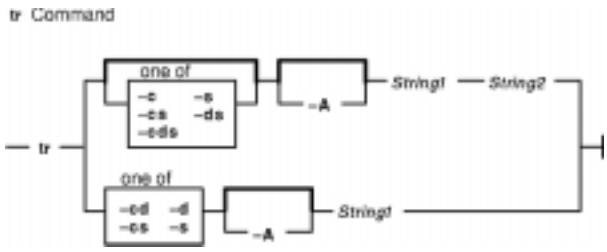
The **terminfo** file format.

tr Command

Purpose

Translates characters.

Syntax



tr [**−c** | **−cds** | **−cs** | **−ds** | **−s**] [**−A**] *String1String2*

tr { **−cd** | **−cs** | **−d** | **−s** } [**−A**] *String1*

Description

The **tr** command deletes or substitutes characters from standard input and writes the result to standard output. The **tr** command performs three kinds of operations depending on the strings specified by the *String1* and *String2* variable and on the flags specified.

Transforming Characters

If *String1* and *String2* are both specified and the **−d** flag is not specified, the **tr** command replaces each character contained in *String1* from the standard input with the character in the same position in *String2*.

Deleting Characters Using the −d Flag

If the **−d** flag is specified, the **tr** command deletes each character contained in *String1* from standard input.

Removing Sequences Using the −s Flag

If the **−s** flag is specified, the **tr** command removes all but the first character in any sequence of a character string represented in *String1* or *String2*. For each character represented in *String1*, the **tr** command removes all but the first occurrence of the character from standard output. For each character represented in *String2*, the **tr** command removes all but the first occurrence in a sequence of occurrences of that character in the standard output.

Special Sequences for Expressing Strings

The strings contained in the *String1* and *String2* variables can be expressed using the following conventions:

C1−C2	Specifies the string of characters that collate between the character specified by *C1* and the character specified by *C2*, inclusive. The character specified by *C1* must collate before the character specified by *C2*. **Note:** The current locale has a

	significant effect on results when specifying subranges using this method. If the command is required to give consistent results irrespective of locale, the use of subranges should be avoided.
[*C********Number***]**	*Number* specifies, as an integer, the number of repetitions of the character specified by *C*. *Number* is considered a decimal integer unless the first digit is a 0; then it is considered an octal integer.
[*C****]**	Fills out the string with the character specified by *C*. This option, used only at the end of the string contained within *String2*, forces the string within *String2* to have the same number of characters as the string specified by the *String1* variable. Any characters specified after the ***** (asterisk) are ignored.

[:*ClassName***:]**	Specifies all of the characters in the character class named by *ClassName* in the current locale. The class name can be any of the following names:
	`alnum lower`
	`alpha print`
	`blank punct`
	`cntrl space`
	`digit upper`
	`graph xdigit`
	For more information on character classes, see the **ctype** subroutines.

[=*C***=]**	Specifies all of the characters with the same equivalence class as the character specified by *C*.
Octal	Specifies the character whose encoding is represented by the octal value specified by *Octal*. *Octal* can be a one−, two− or three−digit octal integer. The NULL character can be expressed with `'\0'`, and is processed like any other character.
ControlCharacter	Specifies the control character that corresponds to the value specified by *ControlCharacter*. The following values can be represented: **\a** Alert **\b** Backspace **\f** Form−feed **\n** New line **\r** Carriage return **\t** Tab **\v** Vertical tab
\\	Specifies the backslash character as itself, without any special meaning as an escape character.
\[	Specifies the left bracket character as itself, without any special meaning as the beginning of a special string sequence.
\−	Specifies the minus sign character as itself, without any special meaning as a range separator.

If a character is specified more than once in *String1*, the character is translated into the character in *String2* that corresponds to the last occurrence of the character in *String1*.

If the strings specified by *String1* and *String2* are not the same length, the **tr** command ignores the extra characters in the longer string.

Flags

−A Performs all operations on a byte−by−byte basis using the ASCII collation order for ranges and character classes, instead of the collation order for the current locale.

−c Specifies that the value of *String1* be replaced by the *complement* of the string specified by *String1*. The complement of *String1* is all of the characters in the character set of the current locale, *except* the characters specified by *String1*. If the **−A** and **−c** flags are both specified, characters are complemented with respect to the set of all 8−bit character codes. If the **−c** and **−s** flags are both specified, the **−s** flag applies to characters in the complement of *String1*.

−d Deletes each character from standard input that is contained in the string specified by *String1*.

−s Removes all but the first in a sequence of a repeated characters. Character sequences specified by *String1* are removed from standard input before translation, and character sequences specified by *String2* are removed from standard output.

String1 Specifies a string of characters.

String2 Specifies a string of characters.

Exit Status

This command returns the following exit values:

0 All input was processed successfully.

>0 An error occurred.

Examples

1. To translate braces into parentheses, enter:

   ```
   tr '{}' '()' < textfile > newfile
   ```

   This translates each { (left brace) to ((left parenthesis) and each } (right brace) to) (right parenthesis). All other characters remain unchanged.

2. To translate braces into brackets, enter:

   ```
   tr '{}' '\[]' < textfile > newfile
   ```

   This translates each { (left brace) to [(left bracket) and each } (right brace) to] (right bracket). The left bracket must be entered with a \ (backslash) escape character.

3. To translate lowercase characters to uppercase, enter:

   ```
   tr 'a−z' 'A−Z' < textfile > newfile
   ```

4. To create a list of words in a file, enter:

   ```
   tr −cs '[:lower:][:upper:]' '[\n*]' < textfile > newfile
   ```

   This translates each sequence of characters other than lowercase letters and uppercase letters into a single newline character. The * (asterisk) causes the **tr** command to repeat the new line character enough times to make the second string as long as the first string.

5. To delete all NULL characters from a file, enter:

   ```
   tr -d '\0' < textfile > newfile
   ```

6. To replace every sequence of one or more new lines with a single new line, enter:

   ```
   tr -s '\n' < textfile > newfile
   ```

   OR

   ```
   tr -s '\012' < textfile > newfile
   ```

7. To replace every nonprinting character, other than valid control characters, with a ? (question mark), enter:

   ```
   tr -c '[:print:][:cntrl:]' '[?*]' < textfile > newfile
   ```

   This scans a file created in a different locale to find characters that are not printable characters in the current locale.

8. To replace every sequence of characters in the <space> character class with a single # (pound sign) character, enter:

   ```
   tr -s '[:space:]' '[#*]'
   ```

Related Information

The **ed** command, **trbsd** command.

The **ctype** subroutines.

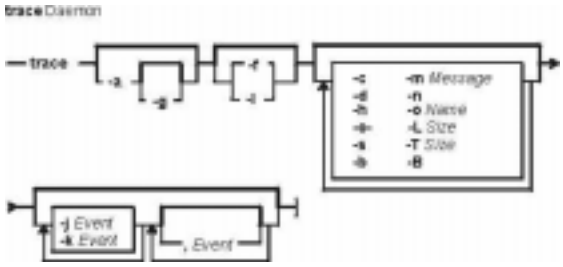
National Language Support Overview for Programming in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

trace Daemon

Purpose

Records selected system events.

Syntax



trace [**−a** [**−g**]] [**−f** | **−l**] [**−b** | **−B**] [**−c**] [**−d**] [**−h**] [**−j** *Event* [**,***Event*]] [**−k** *Event* [**,***Event*]] [**−m** *Message*] [**−n**] [**−o** *Name*] [**−o−**] [**−s**] [**−L** *Size*] [**−T** *Size*]

Description

The **trace** daemon configures a trace session and starts the collection of system events. The data collected by the trace function is recorded in the trace log. A report from the trace log can be generated with the **trcrpt** command.

When invoked with the **−a** flag, the trace daemon is run asynchronously (i.e. as a background task). Otherwise, it is run interactively and prompts you for subcommands.

You can use the System Management Interface Tool (SMIT) to run the **trace** daemon. To use SMIT, enter:

```
smit trace
```

There are three modes of trace data collection:

Alternate (the default) All trace events are captured in the trace log file.

Circular (**−l**)

The trace events wrap within the in−memory buffers and are not captured in the trace log file until the trace data collection is stopped.

Single (**−f**)

The collection of trace events stops when the in−memory trace buffer fills up and the contents of the buffer are captured in the trace log file.

Buffer Allocation

Trace buffers are allocated from either the kernel heap, or are put into separate segments. By default, buffers are allocated from the kernel heap unless the buffer size requested is too large for buffers to fit in the kernel heap, in which case they are allocated in separate segments. Allocating buffers from separate segments hinders trace performance somewhat. However, buffers in separate segments will not take up paging space, just pinned memory. The type of buffer allocation can be

specified with the optional **−b** or **−B** flags.

Flags

−a Runs the **trace** daemon asynchronously (i.e. as a background task). Once **trace** has been started this way, you can use the **trcon**, **trcoff**, and **trcstop** commands to respectively start tracing, stop tracing, or exit the trace session. These commands are implemented as links to **trace.**

−b Allocate buffers from the kernel heap. If the requested buffer space can not be obtained from the kernel heap, the command fails.

−B Allocate buffers in separate segments.

−c Saves the trace log file, adding **.old** to its name.

−d Disables the automatic start of trace data collection. Delays starting of trace data collection. Normally, the collection of trace data starts automatically when you issue the **trace** daemon. Use the **trcon** command to start the collection of trace data.

−f Runs **trace** in a single mode. Causes the collection of trace data to stop as soon as the in−memory buffer is filled up. The trace data is then written to the trace log. Use the **trcon** command to restart trace data collection and capture another full buffer of data. If you issue the **trcoff** subcommand before the buffer is full, trace data collection is stopped and the current contents of the buffer are written to the trace log.

−g Starts a trace session on a generic trace channel (channels 1 through 7). This flag works only when **trace** is run asynchronously (**−a**). The return code of the command is the channel number; the channel number must subsequently be used in the generic trace subroutine calls. To stop the generic trace session, use the command **trcstop −**<channel_number>.

−h Omits the header record from the trace log. Normally, the **trace**daemon writes a header record with the date and time (from the **date** command) at the beginning of the trace log; the system name, version and release, the node identification, and the machine identification (from the **uname −a** command); and a user−defined message. At the beginning of the trace log, the information from the header record is included in the output of the **trcrpt** command.

−j *Event*[**,***Event*]

−k *Event*[**,***Event*] Specifies the user−defined events for which you want to collect (**−j**) or exclude (**−k**) trace data. The *Event* list items can be separated by commas, or enclosed in double quotation marks and separated by commas or blanks.

> **Note:** The following events are used to determine the pid, the cpuid and the exec path name in the **trcrpt** report:

```
001 TRACE ON
002 TRACE OFF
106 DISPATCH
10C DISPATCH IDLE PROCESS
134 EXEC SYSTEM CALL
139 FORK SYSTEM CALL
465 KTHREAD CREATE
```

If any of these events is missing, the information reported by the **trcrpt** command will be incomplete. Consequently: when using the **−j** flag, you should include all these events in the *Event* list; conversely, when using the **−k** flag, you should not include these events in the *Event* list.

−l Runs **trace** in a circular mode. The **trace** daemon writes the trace data to the trace log when the collection of trace data is stopped. Only the last buffer of trace data is captured. When you stop trace data collection using the **trcoff** command, restart it using the **trcon**command.

−L *Size* Overrides the default trace log file size of 1MB with the value stated. Specifying a file size of zero sets the trace log file size to the default size. HERE

Note: In the circular and the alternate modes, the trace log file size must be at least twice the size of the trace buffer. In the single mode, the trace log file must be at least the size of the buffer. See the **−T** flag for information on controlling the trace buffer size.

−m *Message* Specifies text to be included in the message field of the trace log header record.

−n Adds information to the trace log header: lock information, hardware information, and, for each loader entry, the symbol name, address, and type.

−o *Name* Overrides the **/var/adm/ras/trcfile** default trace log file and writes trace data to a user−defined file.

−o − Overrides the default trace log name and writes trace data to standard output. The **−c** flag is ignored when using this flag.

−s Stops tracing when the trace log fills. The **trace** daemon normally wraps the trace log when it fills up and continues to collect trace data. During asynchronous operation, this flag causes the **trace** daemon to stop trace data collection. (During interactive operation, the **quit** subcommand must be used to stop trace.)

−T *Size* Overrides the default trace buffer size of 128KB with the value stated. You must be root to request more than 1MB of buffer space. The maximum possible size is 268435184 bytes.

Note: In the circular and the alternate modes, the trace buffer size must be one−half or less the size of the trace log

file. In the single mode, the trace log file must be at least the size of the buffer. See the **−L** flag for information on controlling the trace log file size. Also note that trace buffers use pinned memory, in other words, they are not pageable. Therefore, the larger the trace buffers, the less physical memory is available to applications.

Unless the **−b** or **−B** flags are specified, the system attempts to allocate the buffer space from the kernel heap. If this request can not be satisfied, the system then attempts to allocate the buffers as separate segments.

Subcommands

When run interactively, trace recognizes the following subcommands:

trcon	Starts the collection of trace data.
trcoff	Stops the collection of trace data.
q or **quit**	Stops the collection of trace data and exits **trace**.
! *Command*	Runs the shell command specified by the *Command* parameter.
?	Displays the summary of **trace** subcommands.

Signals

The **INTERRUPT** signal acts as a toggle to start and stop the collection of trace data. Interruptions are set to **SIG_IGN** for the traced process.

Examples

1. To use trace interactively, enter `trace`, (the `>` prompt is displayed), then specify the subcommands you want. For example, to trace system events during the run of the *anycmd* command, enter:

   ```
   trace
   > !anycmd
   > q
   ```

2. To avoid delays when the command finishes, start trace asynchronously (**−a**), using only one command line. Enter:

   ```
   trace −a; anycmd; trcstop
   ```

3. To trace the system itself for a period of 10 seconds, enter:

   ```
   trace −a; sleep 10; trcstop
   ```

4. To output trace data to a specific trace log file (instead of the **/var/adm/ras/trcfile** default trace log file), :

   ```
   trace −a −o /tmp/my_trace_log; anycmd; trcstop
   ```

5. To capture the execution of a **cp** command, excluding specific events from the collection process:

   ```
   trace −a −k "20e,20f"; cp /bin/track /tmp/junk; trcstop
   ```

In the example above, the **−k** option suppresses the collection of events from the **lockl** and **unlockl** functions (20e and 20f events).

Files

/usr/include/sys/trcmacros.h	Defines **trchook** and **utrchook** macros.
/var/adm/ras/trcfile	Contains the default trace log file.

Related Information

The **trcnm** command, the **trcrpt** command, the **trcstop** command.

The **trchook** subroutine, **trcgen** subroutine, **trcstart** subroutine, **trcon** subroutine, **trcoff** subroutine, **trcstop** subroutine.

The **trcgenk** kernel service.

Trace Facility Overview in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs.*

Performance Analysis with the Trace Facility in *AIX Versions 3.2 and 4 Performance Tuning Guide*.

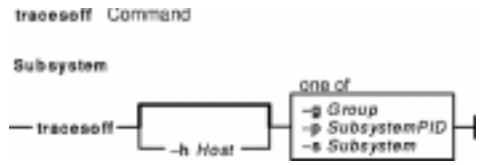
tracesoff Command

Purpose

Turns off tracing of a subsystem, a group of subsystems, or a subserver.

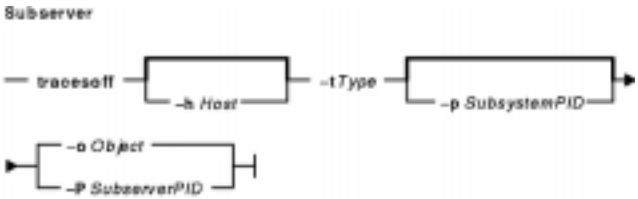
Syntax

Subsystem



tracesoff [**−h** *Host*] { **−g***Group* | **−p***SubsystemPID* | **−s***Subsystem*}

Subserver



tracesoff [**−h** *Host*] **−t***Type* [**−p** *SubsystemPID*] { **−o***Object* | **−P***SubserverPID*}

Description

The **tracesoff** command sends the System Resource Controller a subsystem request packet that is forwarded to the subsystem to turn tracing off. Tracing is unsuccessful if the communication method for the subsystems is signals.

> **Note:** Tracing is subsystem dependent.

Flags

−g*Group*	Specifies a group of subsystems to turn tracing off. The command is unsuccessful if the *Group* name is not contained in the subsystem object class.
−h *Host*	Specifies the foreign host on which this trace action is requested. The local user must be running as "root". The remote system must be configured to accept remote System Resource Controller requests. That is, the **srcmstr** daemon (see **/etc/inittab**) must be started with the **−r** flag and the **/etc/hosts.equiv** or **.rhosts** file must be configured to allow remote requests.
−o *Object*	Specifies that a subserver *Object* name is to be passed to the subsystem as a character string.
−p*SubsystemPID*	Specifies a particular instance of the subsystem to turn tracing off, or a particular instance of the subsystem to which the trace off subserver request is to be passed.
−P*SubserverPID*	Specifies that a *SubserverPID* is to be passed to the subsystem as a character string.

−s*Subsystem*	Specifies a subsystem to turn tracing off. The *Subsystem* name can be the actual subsystem name or the synonym name for the subsystem. The command is unsuccessful if the *Subsystem* name is not contained in the subsystem object class.
−t *Type*	Specifies a subsystem subserver to turn tracing off. The command is unsuccessful if the *Type* is not contained in the subserver object class.

Examples

To turn off the tracing of a subsystem, enter:

```
tracesoff −s
tcpip
```

This turns the tracing off for the `tcpip` subsystem.

Files

/usr/bin/tracesoff	Contains the **tracesoff** command.
/etc/objrepos/SRCsubsys	Specifies the SRC Subsystem Configuration Object Class.
/etc/objrepos/SRCsubsvr	Specifies the SRC Subserver Configuration Object Class.
/etc/services	Defines the sockets and protocols used for Internet services.
/dev/SRC	Specifies the **AF_UNIX** socket file.
/dev/.SRC−unix	Specifies the location for temporary socket files.

Related Information

The **traceson** command.

The System Resource Controller Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* gives an explanation of subsystems, subserver, and the System Resource Controller.

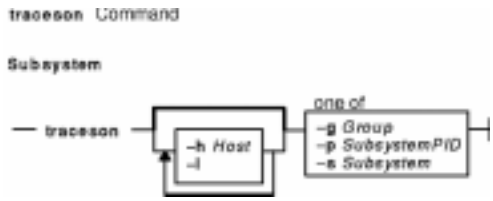
traceson Command

Purpose

Turns on tracing of a subsystem, a group of subsystems, or a subserver.

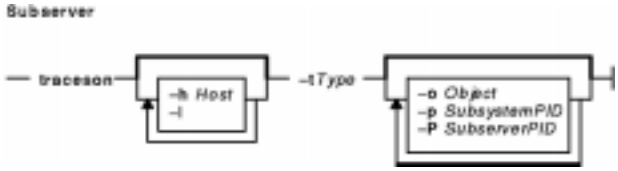
Syntax

Subsystem



traceson [**−h***Host*] [**−l**] {**−g***Group* | **−p***SubsystemPID*|**−s***Subsystem*}

Subserver



traceson [**−h***Host*] [**−l**] **−t***Type* [**−o***Object*] [**−p***SubsystemPID*] [**−P***SubserverPID*]

Description

The **traceson** command sends the System Resource Controller a subsystem request packet that is forwarded to the subsystem to turn tracing on. Tracing is unsuccessful if the communication method for the subsystems is signals.

> **Note:** Tracing is subsystem dependent.

Tracing may occur in either short or long form. When the **−l** flag is absent, the trace request is assumed to be a short trace.

Flags

−g*Group*	Specifies a group of subsystems to turn tracing on. The command is unsuccessful if the *Group* name is not contained in the subsystem object class.
−h *Host*	Specifies the foreign host on which this trace action is requested. The local user must be running as "root". The remote system must be configured to accept remote System Resource Controller requests. That is, the **srcmstr** daemon (see **/etc/inittab**) must be started with the **−r** flag and the **/etc/hosts.equiv** or **.rhosts** file must be configured to allow remote requests.
−l	Specifies that a long trace is requested.
−o *Object*	Specifies that a subserver object is to be passed to the subsystem as a character string.

−p*SubsystemPID*	Specifies a particular instance of the subsystem to turn tracing on, or a particular instance of the subsystem to which the trace subserver request is to be passed.
−P *SubserverPID*	Specifies that a subserver PID is to be passed to the subsystem as a character string.
−s*Subsystem*	Specifies the subsystem to turn tracing on. The *Subsystem* name can be either the actual subsystem name or the synonym name for the subsystem. The command is unsuccessful if the *Subsystem* name is not contained in the subsystem object class.
−t *Type*	Specifies a subserver to turn tracing on. The command is unsuccessful if the *Type* is not contained in the subserver object class.

Examples

To turn on tracing of the `tcpip` subsystem on a foreign host, enter:

```
traceson −h odin −s tcpip
```

This turns on the tracing for the `tcpip` subsystem on the `odin` foreign host.

Files

/usr/bin/traceson	Contains the **traceson** command.
/etc/objrepos/SRCsubsys	Specifies the SRC Subsystem Configuration Object Class.
/etc/objrepos/SRCsubsvr	Specifies the SRC Subserver Configuration Object Class.
/etc/services	Defines the sockets and protocols used for Internet services.
/dev/SRC	Specifies the **AF_UNIX** socket file.
/dev/.SRC−unix	Specifies the location for temporary socket files.

Related Information

The **tracesoff** command.

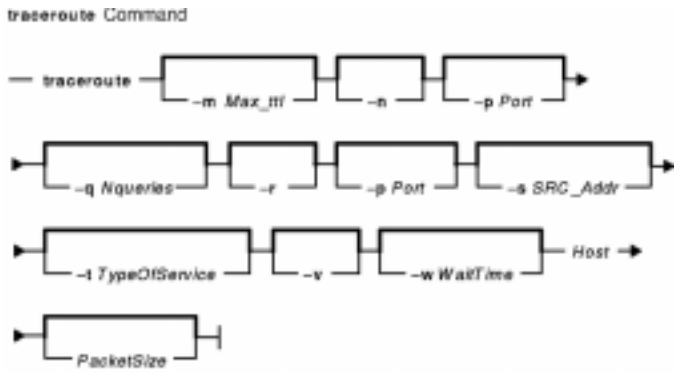
The System Resource Controller Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* gives an explanation of subsystems, subservers, and the System Resource Controller.

traceroute Command

Purpose

Prints the route that IP packets take to a network host.

Syntax



traceroute [**−m***Max_ttl*] [**−n**] [**−p***Port*] [**−q***Nqueries*] [**−r**] [**−s***SRC_Addr*] [**−t** *TypeOfService*] [**−v**] [**−w** *WaitTime*] *Host* [*PacketSize*]

Description

> **Attention:** The **traceroute** command is intended for use in network testing, measurement, and management. It should be used primarily for manual fault isolation. Because of the load it imposes on the network, the **traceroute** command should not be used during normal operations or from automated scripts.

The **traceroute** command attempts to trace the route an IP packet follows to an Internet host by launching UDP probe packets with a small maximum time−to−live (*Max_ttl* variable), then listening for an ICMP **TIME_EXCEEDED** response from gateways along the way. Probes are started with a *Max_ttl* value of one hop, which is increased one hop at a time until an ICMP **PORT_UNREACHABLE** message is returned. The ICMP **PORT_UNREACHABLE** message indicates either that the host has been located or the command has reached the maximum number of hops allowed for the trace.

The **traceroute** command sends three probes at each *Max_ttl* setting to record the following:

- *Max_ttl* value
- Address of the gateway
- Round−trip time of each successful probe

The number of probes sent can be increased by using the **−q** flag. If the probe answers come from different gateways, the command prints the address of each responding system. If there is no response from a probe within a 3−second time−out interval, an * (asterisk) is printed for that probe.

The **traceroute** command prints an ! (exclamation mark) after the round−trip time if the *Max_ttl* value is one hop or less. A maximum time−to−live value of one hop or less generally indicates an incompatibility in the way ICMP replies are handled by different network software. The incompatibility can usually be resolved by doubling the last *Max_ttl* value used and trying again.

Other possible annotations after the round−trip notation are:

!H Host unreachable
!N Network unreachable
!P Protocol unreachable
!S Source route failed
!F Fragmentation needed

If the majority of probes result in an error, the **traceroute** command exits.

The only mandatory parameter for the **traceroute** command is the destination host name or IP number. The default probe length is 38 bytes, but may be increased by specifying the packet size (in bytes) after the destination host name. The UDP probe packets are set to an unlikely value so as to prevent processing by the destination host.

Flags

−m *Max_ttl*	Sets the maximum time−to−live (maximum number of hops) used in outgoing probe packets. The default is 30 hops (the same default used for TCP connections).
−n	Prints hop addresses numerically rather than symbolically and numerically. This flag saves a name−server address−to−name lookup for each gateway found on the path.
−p *Port*	Sets the base UDP port number used in probes. The default is 33434. The **traceroute** command depends on an open UDP port range of *base* to *base* + *nhops* − 1 at the destination host. If a UDP port is not available, this option can be used to pick an unused port range.
−q*Nqueries*	Specifies the number of probes the **traceroute** command sends at each *Max_ttl* setting. The default is three probes.
−r	Bypasses the normal routing tables and sends the probe packet directly to a host on an attached network. If the specified host is not on a directly attached network, an error is returned. This option can be used to issue a **ping** command to a local host through an interface that is not registered in the **routed** daemon's routing table.
−s *SRC_Addr*	Uses the next IP address in numerical form as the source address in outgoing probe packets. On hosts with more than one IP address, the **−s** flag can be used to force the source address to be something other than the IP address of the interface on which the probe packet is sent. If the next IP address is not one of the machine's interface addresses, an error is returned and nothing is sent.
−t*TypeOfService*	Sets the *TypeOfService* variable in the probe packets to a decimal integer in the range of 0 to 255. The default is 0. This flag can be used to investigate whether different service types result in different paths. For more information, see "TCP/IP Protocols" in *AIX Version 4.3 System Management Guide: Communications and Networks*. Useful values are **−t16** (low delay) and **−t8** (high throughput).
−v	Receives packets other than **TIME_EXCEEDED** and **PORT_UNREACHABLE** (verbose output).
−w *WaitTime*	Sets the time (in seconds) to wait for a response to a probe. The default is 3 seconds.

Parameters

Host	Specifies the destination host, either by host name or IP number. This parameter is required.
PacketSize	Specifies the probe datagram length. The default is 38 bytes. This number can be increased by specifying the packet size, in bytes, after the destination host name.

Examples

1. A sample use and output is:

```
[yak 71]% traceroute nis.nsf.net.
traceroute to nis.nsf.net (35.1.1.48), 30 hops max, 56 byte packet
 1 helios.ee.lbl.gov (128.3.112.1) 19 ms 19 ms 0 ms
 2 lilac-dmc.Berkeley.EDU (128.32.216.1) 39 ms 39 ms 19 ms
 3 lilac-dmc.Berkeley.EDU (128.32.216.1) 39 ms 39 ms 19 ms
 4 ccngw-ner-cc.Berkeley.EDU (128.32.136.23) 39 ms 40 ms 39 ms
 5 ccn-nerif22.Berkeley.EDU (128.32.168.22) 39 ms 39 ms 39 ms
 6 128.32.197.4 (128.32.197.4) 40 ms 59 ms 59 ms
 7 131.119.2.5 (131.119.2.5) 59 ms 59 ms 59 ms
 8 129.140.70.13 (129.140.70.13) 99 ms 99 ms 80 ms
 9 129.140.71.6 (129.140.71.6) 139 ms 239 ms 319 ms
10 129.140.81.7 (129.140.81.7) 220 ms 199 ms 199 ms
11 nic.merit.edu (35.1.1.48) 239 ms 239 ms 239 ms
```

   Lines 2 and 3 are the same due to a bug in the kernel on the second hop system (`lbl-csam.arpa`) that forwards packets with a zero time-to-live. Host names are not printed in lines 6 through 10 because the National Science Foundation Network (NSFNet, `129.140`) does not provide address-to-name translations for its nodes.

2. Another output example might be:

```
[yak 72]% traceroute rip.Berkeley.EDU (128.32.131.22)
traceroute to rip.Berkeley.EDU (128.32.131.22), 30 hops max
 1 helios.ee.lbl.gov (128.3.112.1) 0 ms 0 ms 0 ms
 2 lilac-dmc.Berkeley.EDU (128.32.216.1) 39 ms 19 ms 39 ms
 3 lilac-dmc.Berkeley.EDU (128.32.216.1) 19 ms 39 ms 19 ms
 4 ccngw-ner-cc.Berkeley.EDU (128.32.136.23) 39 ms 40 ms 19 ms
 5 ccn-nerif35.Berkeley.EDU (128.32.168.35) 39 ms 39 ms 39 ms
 6 csgw/Berkeley.EDU (128.32.133.254) 39 ms 59 ms 39 ms
 7 * * *
 8 * * *
 9 * * *
10 * * *
11 * * *
12 * * *
13 rip.Berkeley.EDU (128.32.131.22) 59 ms! 39 ms! 39 ms!
```

In this example, exactly half of the 12 gateway hops (13 is the final destination) are "missing." However, these hops were actually not gateways. The destination host, a Sun-3 workstation running Sun OS3.5, used the ttl from the arriving datagram as the ttl in its ICMP reply; thus, the reply timed out on the return path. Because ICMPs are not sent for ICMPs, no notice was received. The ! (exclamation mark) after each round-trip time indicates some type of software incompatibility problem. (The cause was diagnosed after the **traceroute** command issued a probe of twice the path length. The destination host was really only seven hops away.)

Related Information

The **netstat** command, **nslookup** command, **ping** command.

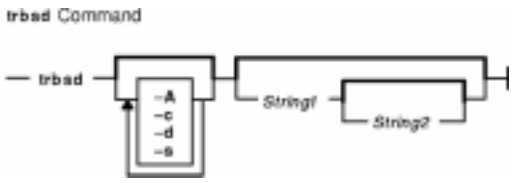
TCP/IP Name Resolution in *AIX Version 4.3 System Management Guide: Communications and Networks*.

trbsd Command

Purpose

Translates characters (BSD version).

Syntax



trbsd [**−c**] [**−d**] [**−s**] [**−A**] [*String1* [*String2*]]

Description

The **trbsd** command deletes or substitutes characters from standard input and then writes the result to standard output. The **trbsd** command is the BSD version of the **tr** command. The **trbsd** command performs three kinds of operations, depending on the character strings specified by the parameters and flags specified. The default value for either the *String1* or *String2* parameter is a null string.

Transforming Characters

If both the *String1* and *String2* parameters are specified and the **−d** flag is not specified, the **trbsd** command replaces each character from standard input that is specified by the *String1* parameter with the character in the same position in the *String2* parameter.

If the *String1* parameter specifies a character more than once, the character is translated into the character in the *String2* parameter that corresponds to the last occurrence of the character in the *String1* parameter.

Deleting Characters Using the −d Flag

If the **−d** flag is specified, the **trbsd** command deletes each character from standard input that is specified by the *String1* parameter.

Removing Sequences of Characters Using the −s Flag

If the **−s** flag is specified, the **trbsd** command deletes from standard input all but the first character in a sequence of two or more repetitions of any character specified by the *String2* parameter.

Both the *String1* and *String2* parameters must be specified when both the **−d** and **−s** flags are specified.

> **Note:** The **trbsd** command deletes all null characters from standard input before it begins processing.

Special Sequences for Expressing Strings

The strings contained in *String1* and *String2* parameters can be expressed using the following conventions:

C1−C2 Specifies the string of characters that collate between the character specified by the *C1* string and

the character specified by the *C2* string, inclusive. The character specified by the *C1* string must collate before the character specified by the *C2* string.

Octal Specifies the character whose encoding is represented by the specified octal value. The octal value can be a one−, two−, or three−digit octal integer. Multibyte characters can be expressed by writing backslash−octal sequences for each byte.

\− The \− (backslash, minus sign) specifies the minus sign character itself, without any special meaning as an escape character.

If the strings specified by the *String1* and *String2* parameters are not the same length, the **trbsd** command pads the shorter string to equal the length of the longer string. Padding is accomplished by duplicating the last character in the shorter string as many times as necessary.

Flags

−A Performs all operations on a byte−by−byte basis using the ASCII collation order for ranges and character classes, instead of the collation order of the current locale.

−c Specifies that the value of the *String1* parameter be replaced by the complement of that string. The complement is all of the characters in the character set of the current locale, except for the characters specified by the *String1* parameter. If the **−A** and **−c** flags are specified together, characters are complemented with respect to the set of all 8−bit character codes.

−d Deletes each character from standard input that is contained in the *String1* parameter.

−s Deletes from standard input all but the first character in a sequence of two or more repetitions of any character contained in the *String2* parameter.

Examples

1. To translate braces into parentheses, enter:

   ```
   trbsd '{}' '()' < textfile > newfile
   ```

   This translates each { (left brace) to ((left parenthesis) and each } (right brace) to) (right parenthesis). All other characters remain unchanged.

2. To interchange plus signs with minus signs, and slash characters with asterisks, enter:

   ```
   trbsd '+\−/*' '\−+*/' < textfile > newfile
   ```

   The minus sign must be entered with a backslash escape character.

3. To translate lowercase characters to uppercase, enter:

   ```
   trbsd 'a−z' 'A−Z' < textfile > newfile
   ```

4. To create a list of words in a file, enter:

   ```
   trbsd −cs 'a−zA−Z' '\012' < textfile > newfile
   ```

   This translates each sequence of characters other than lowercase letters and uppercase letters into a single newline character. The octal value 012 is the code for the newline character.

5. To replace every sequence of one or more newlines with a single newline, enter:

   ```
   trbsd −s '\012'  < textfile > newfile
   ```

Files

/usr/bin/trbsd Contains the **trbsd** command.
/usr/ucb/tr Contains a symbolic link to the **trbsd** command.

Related Information

The **ed** command, **tr** command.

National Language Support Overview for Programming in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

trcdead Command

Purpose

Extracts the trace buffer from a system dump image.

Syntax

trcdead [**−o***FileName*] *DumpImage*

Description

The **trcdead** command extracts the eight active trace channels from a system dump. If the system halts while the trace facilities are active, the contents of the internal trace buffers are captured. If the **−o** flag is used, the **trcdead** command extracts the trace event data from the dump and writes it to the specified file. If the flag is not chosen, the command writes to the trace log file.

Use the **trcrpt** command to format a report from the trace log file.

Flag

−o*FileName* Specifies the file to which data is written.

Examples

> **Note:** To determine which example is more appropriate for your system, use the **sysdumpdev** command to display the current dump device assignments.

1. To extract information from a dump image written to a file named `trace_extract`, enter:

   ```
   trcdead −o trace_extract /var/adm/ras/dumpfile
   ```

2. To extract information from a dump image written to a device, enter:

   ```
   trcdead /dev/hd7
   ```

Files

/usr/bin/trcdead	Contains the **trcdead** command.
/var/adm/ras/dumpfile	Contains the default system dump file.
/var/adm/ras/trcfile	Contains the default system trace log.

Related Information

The **errdead** command, **sysdumpdev** command, **trcnm** command, **trcrpt** command.

The **trace** daemon.

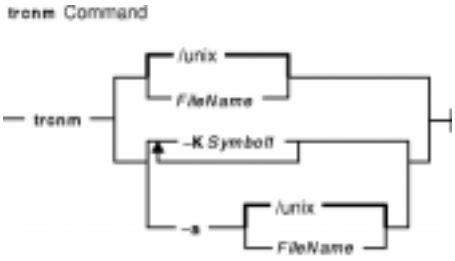
The Trace Facility Overview in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

trcnm Command

Purpose

Generates a kernel name list.

Syntax



trcnm [**−a** [*FileName*]] | [*FileName*] | **−K***Symbol1 ...*

Description

The **trcnm** command generates a kernel name list used by the **trcrpt** command. A kernel name list is composed of a symbol table and a loader symbol table of an object file. The **trcrpt** command uses the kernel name list file to interpret addresses when formatting a report from a trace log file. For more information, see the **trcrpt−n** command.

If the *FileName* parameter is not specified, the default *FileName* is /unix.

Flags

−a Writes all loader symbols to standard output. The default is to write loader symbols only for system calls.

−K*Symbol...* Obtains the value of all command line symbols through the **knlist** command system call.

Examples

1. To obtain the value of the symbols in /unix, enter:

   ```
   trcnm −K environ errno
   ```

   This command sequence displays the following:

   ```
   environ 2FF7FFF8
   errno 2FF7FFFC
   ```

2. To print a symbol table for system calls, enter:

   ```
   trcnm
   ```

   A list similar to the following is generated:

   ```
   pin_obj_start    00000000
   ```

```
header_offset    00000008
ram_disk_start   0000000C
ram_disk_end     00000010
dbg_avail        00000014
base_conf_start  00000018
base_conf_end    0000001C
base_conf_disk   00000020
pin_com_start    00000024
start            00000028
ipl_cb           00000028
...
```

Files

/var/adm/ras/trcfile	Contains the default log file.
/tlo−tvl2/trcnam	Contains the **trcnm** command.
/etc/trcfmt	Contains the trace format file.

Related Information

The **trcdead** command, **trcrpt** command, **trcstop** command, **trcupdate** command.

The **trace** daemon.

The **trcfmt** file format.

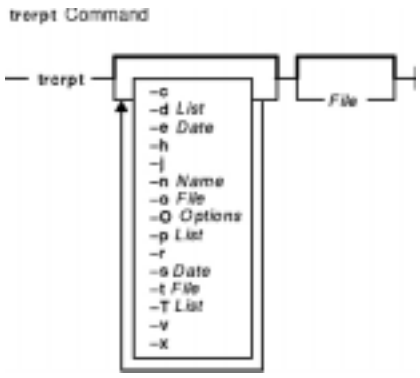
Trace Facility Overview in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

trcrpt Command

Purpose

Formats a report from the trace log.

Syntax



trcrpt [**−c**] [**−d** *List*] [**−e** *Date*] [**−h**] [**−j**] [**−n***Name*] [**−o** *File*] [**−p** *List*] [**−r**]
[**−s** *Date*] [**−t** *File*] [**−T** *List*] [**−v**] [**−O***Options*] [**−x**] [*File*]

Description

The **trcrpt** command reads the trace log specified by the *File* parameter, formats the trace entries and writes a report to standard output. The default file from which the system generates a trace report is the **/var/adm/ras/trcfile** file, but you can specify an alternate *File* parameter.

You can use the System Management Interface Tool (SMIT) to run the **trcrpt** command by entering the SMIT fast path:

```
smit trcrpt
```

Flags

−c	Checks the template file for syntax errors.
−d*List*	Limits report to hook IDs specified with the *List* variable. The *List* parameter items can be separated by commas or enclosed in double quotation marks and separated by commas or blanks.
−e*Date*	Ends the report time with entries on, or before the specified date. The *Date* variable has the form *mmddhhmmssyy* (month, day, hour, minute, second, and year). Date and time are recorded in the trace data only when trace data collection is started and stopped. If you stop and restart trace data collection multiple times during a trace session, date and time are recorded each time you start or stop a trace data collection. Use this flag in combination with the **−s** flag to limit the trace to data collected during a certain time interval.
−h	Omits the header information from the trace report and writes only formatted trace entries to standard output.
−j	Displays the list of hook IDs. The **trcrpt −j** command can be used with the **trace −j** command that includes IDs of trace events or the **trace −k** command that excludes IDs of trace events.
−n*Name*	Specifies the kernel name list file to be used to interpret address for output. Usually, this flag is used when moving a trace log file to another system.

−o*File* Writes the report to a file instead of to standard output.

−O*Options* Specifies options that change the content and presentation of the **trcrpt** command. Arguments to the options must be separated by commas. Valid options are:

2line=[**on**|**off**] Uses two lines per trace event in the report instead of one. The default value is **off**.

cpuid=[**on**|**off**] Displays the physical processor number in the trace report. The default value is **off**.

endtime=*Seconds* Displays trace report data for events recorded before the seconds specified. Seconds can be given in either an integral or rational representation. If this option is used with the **starttime** option, a specific range can be displayed.

exec=[**on**|**off**] Displays exec path names in the trace report. The default value is **off**.

hist=[**on**|**off**] Logs the number of instances that each hook ID is encountered. This data can be used for generating histograms. The default value is **off**. This option cannot be run with any other option.

ids=[**on**|**off**] Displays trace hook identification numbers in the first column of the trace report. The default value is **on**.

pagesize=*Number* Controls the number of lines per page in the trace report and is an integer within the range of 0 through 500. The column headings are included on each page. No page breaks are present when the default value of 0 is set.

pid=[**on**|**off**] Displays the process IDs in the trace report. The default value is **off**.

starttime=*Seconds* Displays trace report data for events recorded after the seconds specified. The specified seconds are from the beginning of the trace file. Seconds can be given in either an integral or rational representation. If this option is used with the **endtime** option, a specific range of seconds can be displayed.

svc=[**on**|**off**] Displays the value of the system call in the trace report. The default value is **off**.

tid=[**on**|**off**] Displays the thread ID in the trace report. The default value is **off**.

timestamp=[**0**|**1**|**2**|**3**]

Controls the time stamp associated with an event in the trace report. The possible values are:

0 Time elapsed since the trace was started. Values for elapsed seconds and milliseconds are returned to the nearest nanosecond and microsecond, respectively. This is the default value.

1 Short elapsed time.

2 Microseconds.

3	No time stamp.

−p*List* Reports the process IDs for each event specified by the *List* variable. The *List* variable may be a list of process IDs or a list of process names. List items that start with a numeric character are assumed to be process IDs. The list items can be separated by commas or enclosed in double quotation marks and separated by commas or blanks.

−r Outputs unformatted (raw) trace entries and writes the contents of the trace log to standard output one entry at a time. Use the **−h** flag with the **−r** flag to exclude the heading.

−s*Date* Starts the report time with entries on, or before the specified date. The *Date* variable has the form *mmddhhmmssyy* (month, day, hour, minute, second, and year). Date and time are recorded in the trace data only when trace data collection is started and stopped. If you stop and restart trace data collection multiple times during a trace session, date and time are recorded each time you start or stop a trace data collection. Use this flag in combination with the **−e** flag to limit the trace to data collected during a certain time interval.

−t *File* Uses the file specified in the *File* variable as the template file. The default is the **/etc/trcfmt** file.

−T*List* Limits the report to the kernel thread IDs specified by the *List* parameter. The list items are kernel thread IDs separated by commas. Starting the list with a kernel thread ID limits the report to all kernel thread IDs in the list. Starting the list with a **!** (exclamation point) followed by a kernel thread ID limits the report to all kernel thread IDs not in the list.

−v Prints file names as the files are opened. Changes to verbose setting.

−x Displays the exec path name and value of the system call.

Examples

1. To format the trace log file and print the result, enter:

   ```
   trcrpt | qprt
   ```

2. To send a trace report to the **/tmp/newfile** file, enter:

   ```
   trcrpt −o /tmp/newfile
   ```

3. To display process IDs and exec path names in the trace report, enter:

   ```
   trcrpt −O pid=on,exec=on
   ```

4. To create trace ID histogram data, enter:

   ```
   trcrpt −O hist=on
   ```

Files

/usr/bin/trcrpt	Contains the **trcrpt** command.
/var/adm/ras/trcfile	Contains the default log file.
/etc/trcfmt	Contains the trace format file.

Related Information

The **trcdead** command, **trcnm** command, **trcstop** command, **trcupdate** command.

The **trace** daemon.

The **trcfmt** file format.

Trace Facility Overview in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

AIX Performance Monitoring and Tuning Commands in *AIX Versions 3.2 and 4 Performance Tuning Guide*.

trcstop Command

Purpose

Stops the trace function.

Syntax

trcstop

Description

The **trcstop** command issues an **ioctl** subroutine to the trace device driver to end the trace session.

You can use the System Management Interface Tool (SMIT) to run the **trcstop** command. To use SMIT, enter:

```
smit trcstop
```

Example

To terminate the trace background process, enter:

```
trcstop
```

File

/**usr**/**bin**/**trcstop** Contains the **trcstop** command.

Related Information

The **trcrpt** command.

The **trace** daemon.

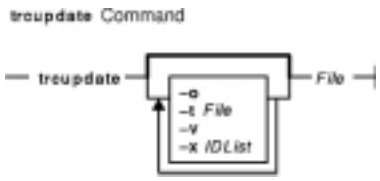
The Trace Facility Overview in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

trcupdate Command

Purpose

Adds, replaces, or deletes trace report format templates.

Syntax



trcupdate [**−o**] [**−t** *File*] [**−v**] [**−x** *IDList*] [*File*]

Description

The **trcupdate** command adds, replaces, or deletes trace report format templates in the **/etc/trcfmt** or the **/etc/trcfmt.Z** file. When the **/etc/trcfmt.Z** file is used, the **trcupdate** command uncompresses the file, updates it, and recompresses it. The **trcupdate** command creates an "undo" file named *File***.undo.trc** in the specified directory.

The **trcupdate** command adds the extension **.trc** to the file name and reads update commands from that file. The undo file is input to the **trcupdate** command if the **−o** (override) flag is specified. When the **−o** flag is specified, the **trcupdate** command undoes the changes previously made to the file.

The first field of each template contains an operator:

+ The plus sign indicates that a template is to be added or replaced. The field that follows this operator contains the template to be replaced.
− The minus sign indicates that a template is to be deleted. The field after this operator contains the hook ID of the template to delete. Delete operations are performed before add or replace operations.

The input to the **trcupdate** command must contain the following as the first line:

```
* /etc/trcfmt
```

The following is a sample trace file:

```
* /etc/trcfmt
+ 15A 1.0 new_fmt
− 1B3
− A14
```

When adding or replacing, the **trcupdate** command compares the version numbers of each input template with the version number of the template with the same hook ID. If the version number of the input template is later than the version of the existing template, the **trcupdate** command replaces the old template with the input template. If a template does not exist, then the input template is added to the file.

The **trcupdate** command will not modify the **/etc/trcfmt** file if a syntax error is detected in the update file.

Flags

−o	Overrides the old template with the input template without verifying the version number of either template.
−t*File*	Specifies a file, instead of the **/etc/trcfmt** or the **/etc/trcfmt.Z** file, to be used as the template file.
−v	Prints the file names as each file is opened.
−x *IDList*	Extracts the templates specified in the *IDList* from the template file and writes them to standard output. The *IDList* parameter lists the hook IDs.

Security

Access Control: Only the root user can run this command.

Examples

1. To add a template, enter:

   ```
   trcupdate
   * /etc/trcfmt
   + 15A 1.0 new_fmt
   ```

2. To delete a template, enter:

   ```
   trcupdate
   * /etc/trcfmt
   − 15A 1.0 new_fmt
   ```

3. To replace a template, enter:

   ```
   trcupdate
   * /etc/trcfmt
   + 15A 1.0 new_fmt
   ```

Files

/usr/bin/trcupdate	Contains the **trcupdate** command.
/etc/trcfmt	Contains the trace format file.
/usr/include/sys/trcmacros.h	Defines **trchook** and **utrchook** macros.

Related Information

The **trcdead** command, **trcrpt** command.

The **trace** daemon.

The **trcfmt** file format.

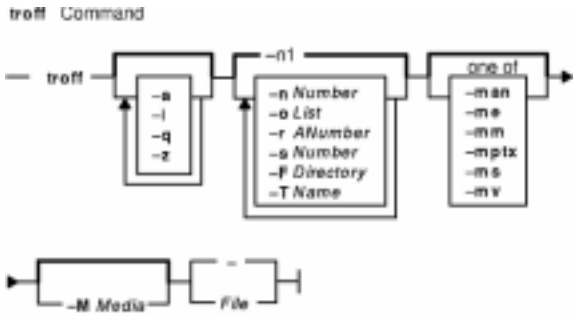
The Trace Facility Overview in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

troff Command

Purpose

Formats text for printing on typesetting devices.

Syntax



troff [**−a**] [**−i**] [**−q**] [**−z**] [**−F** *Directory*] [**−n** *Number*] [**−o** *List*] [**−r** *ANumber*] [**−s** *Number*] [**−T** *Name*] [**−mm** | **−me** | **−mptx** | **−ms** | **−man** | **−mv**] [**−M***Media*] [*File ...* | **−**]

Description

The **troff** command reads one or more files and formats the text for printing on a phototypesetter or comparable device. A postprocessor is then required to post process the output of the **troff** command to the target device. See the accompanying example.

If no file is specified or the **−** (minus) flag is not the last parameter, standard input is read by default.

For the 3812, 3816, and Hewlett−Packard LaserJet Series II printer, the default fonts are the native fonts for the printer. Additional fonts also are available for these printers, which may be loaded through the use of the **troff.fp** directive. These fonts are stored on the host in the directory **/usr/lib/font/dev**Printer/**bitmaps**, and downloaded to the printer as necessary.

Typefaces

Three different typefaces are provided in four styles. The following chart shows the relationship between typeface, style, and the name that the **troff** command uses to access the font.

> **Note:** The fonts in this set are based on the Computer Modern letter forms developed by Donald E Knuth. (Refer to Knuth, Donald: *Computer Modern Typefaces*. Addison−Wesley, 1986.)

Typeface	Regular	Italic	Bold	Italic
Roman	cr	cR	Cr	CR
Sans Serif	cs	cS	Cs	CS

```
Typewriter    ct          cT          Ct          CT

troff special sp
```

These fonts are all provided in the standard 15 troff sizes: 6, 7, 8, 9, 10, 11, 12, 14, 16, 28, 20, 22, 24, 28, and 36 points.

For example, `.fp 1 Cr` loads the Roman bold font into position 1.

> **Note:** The **.tl** request cannot be used before the first break−producing request in the input to the **troff** command.

Flags

−a Sends a printable ASCII approximation of the results to standard output.

−F*Directory* Accesses font information from the Directory/**dev**Name directory instead of the default **/usr/lib/font/dev**Name directory (where *Name* is specified by the **−T** flag).

−i Reads standard input after there are no more files.

−M*Media* Specifies a paper size in order to determine the amount of imageable area on the paper. Valid values for the *Media* variable are:

A4	Specifies a paper size of 8.3 X 11.7 inches (210 X 297 mm).
A5	Specifies a paper size of 5.83 X 8.27 inches (148 X 210 mm).
B5	Specifies a paper size of 6.9 X 9.8 inches (176 X 250 mm).
EXEC	Specifies a paper size of 7.25 X 10.5 inches (184.2 X 266.7 mm).
LEGAL	Specifies a paper size of 8.5 X 14 inches (215.9 X 355.6 mm).
LETTER	Specifies a paper size of 8.5 X 11 inches (215.9 X 279.4 mm). This is the default value.

> **Note:** The *Media* variable is not case−sensitive.

−n*Number* Numbers the first printed page with the value specified by the *Number* variable.

−o*List* Prints only pages specified by the *List* variable, which consists of a comma−separated list of page numbers and ranges:

- A range of *Start*−*Stop* means print pages *Start* through *Stop.* For example: `9−15` prints pages 9 through 15.
- An initial −*Stop* means print from the beginning to page *Stop.*
- A final *Start*− means print from page*Start* to the end.
- A combination of page numbers and ranges prints the specified pages. For example: `−3,6−8,10,12−` prints from the beginning through page 3, pages 6 through 8, page 10, and page 12 to the end.

> **Note:** When this flag is used in a pipeline (for example, with one or more of the **pic**, **eqn,** or **tbl** commands), you may receive a `broken pipe` message if the last page in the document is not specified in the *List* variable. This broken pipe message is not an indication of any problem and can be ignored.

−q Calls the simultaneous input and output mode of the **.rd** request.

−r*ANumber* Sets the register specified by the *A* variable to the specified number. The *A* variable value must have a one−character ASCII name.

−s*Number* Generates output to make the typesetter stop every specified number of pages.

−T*Name* Prepares the output for the specified printing device. Phototypesetters or comparable printing devices use the following *Name* variables for AIX international extended characters. The

default is **ibm3816**.

Note: You get a message that reads `bad point size`if your device does not support the point size that you specified. The **troff** command uses the closest valid point size to continue formatting.

canonls	Canon Lasershot LBP−B406S/D/E,A404/E,A304E.
ibm3812	3812 Pageprinter II.
ibm3816	3816 Pageprinter.
hplj	Hewlett−Packard LaserJet II.
ibm5585H−T	5585−H01 Traditional Chinese Language support.
ibm5587G	5587−G01, 5584−H02, 5585−H01, 5587−H01, and 5589−H01 Kanji Printer multibyte language support.
psc	PostScript printer.
X100	AIXwindows display.

Note: You also can set the **TYPESETTER** environment variable to one of the preceding values instead of using the **−T***Name* flag of the **troff** command.

−man Selects the **man** macro processing package.
−me Selects the **me** macro processing package.
−mm Selects the **mm** macro processing package.
−mptx Selects the **mptx** macro processing package.
−ms Selects the **ms** macro processing package.
−mv Selects the **mv** macro processing package.

Note: See Macro Packages for Formatting Tools for more inforamtion on the macros.

−z Prints only messages generated by **.tm** (workstation message) requests.
− Forces input to be read from standard input.

Environment Variables

TYPESETTER Contains information about a particular printing device.

Examples

The following is an example of the **troff** command:

```
troff −Tibm3812 File | ibm3812 | qprt
```

Macro Packages for Formatting Tools

The following macro packages are part of the Formatting Tools in the Text Formatting System and are described in more detail on the next pages:

man Enables you to create your own manual pages from online manual pages.
me Provides macros for formatting papers.
mm Formats documents with **nroff** and **troff** formatters.
mptx Formats a permuted index.
ms Provides a formatting facility for various styles of articles, theses, and books.

mv Typesets English−language view graphs and slides by using the **troff**command.

man Macro Package for the nroff and troff Commands

The **man** macro package is provided to enable users to create their own manual pages from online manual pages that have been processed with either the **nroff** command or **troff** command. The **man** macro package is used with either the **nroff** command or the **troff** command.

> **Note:** The **man** macro package cannot be used to process the InfoExplorer information bases into manual pages.

Special macros, strings, and number registers exist, internal to the **man** macro package, in addition to the following lists of format macros, strings, and registers. Except for the names predefined by the **troff** command and the **d**, **m**, and **y** number registers, all such internal names are of the form *SymbolAlpha*, where *Symbol* is one of **)**, **]**, or **}**, and *Alpha*is any alphanumeric character.

The **man** macro package uses only the Roman font. If the input text of an entry contains requests for other fonts (for example, the **.I** format macro, **.RB** request, or **\fI** request) the corresponding fonts must be mounted.

Format Macros

The following macros are used to alter the characteristics of manual pages that are formatted using the **man**macro package.

Type font and size are reset to default values before each paragraph and after processing font− and size−setting macros (for example, the **.I** format macro, **.SM** format macro, and **.B** format macro).

Tab stops are neither used nor set by any of the format macros except the **.DT** format macro and the **.TH** format macro.

.B [*Text*] Makes text bold.

The *Text* variable represent up to six words; use " " (double quotation marks) to include character spaces in a word. If the variable is empty, this treatment is applied to the next input text line that contains text to be printed. For example, use the **.I** format macro to italicize an entire line, or use the **.SM** and **.B** format macros to produce an entire line of small−bold text. By default, hyphenation is turned off for the **nroff** command, but remains on for the **troff** command.

.DT Restores default tab settings every 5 ens for the **nroff** command and every 7.2 ens for the **troff** command.

.HP [*Indent*] Begins a paragraph with a hanging indent as specified by the *Indent*variable.

If the *Indent* variable is omitted, the previous *Indent* value is used. This value is set to its default (5 ens for the **nroff** command and 7.2 ens for the **troff** command) by the **.TH** format macro, **.P** format macro, and **.RS** format macro, and restored by the **.RE** format macro. The default unit for *Indent* is ens.

.I [*Text*] Makes text italic.

The *Text* variable represent up to six words; use " " (double quotation marks) to include character spaces in a word. If the variable is empty, this treatment is applied to the next input text line that contains text to

be printed. For example, use the **.I** format macro to italicize an entire line, or use the **.SM** and **.B** format macros to produce an entire line of small−bold text. By default, hyphenation is turned off for the **nroff** command, but remains on for the **troff** command.

.IP [*Tag*] [*Indent*]

Same as the **.TP***Indent* macro with the *Tag* variable; if the value of the *Tag* variable is **NULL**, begin indented paragraph. This macro is often used to get an indented paragraph without a tag.

If the *Indent* variable is omitted, the previous *Indent* value is used. This value is set to its default (5 ens for the **nroff** command and 7.2 ens for the **troff** command) by the **.TH** format macro, **.P** format macro, and **.RS** format macro, and restored by the **.RE** format macro. The default unit for *Indent* is ens.

.P

Begins paragraph with normal font, point size, and indent. The **.PP** macro is a synonym for the **mm** macro package **.P** macro.

.PD [*Number*]

Sets inter−paragraph distance the number of vertical spaces specified by the *Number* parameter. The default *Number* variable value is 0.4v for the **troff** command and 1v for the **nroff** command.

.PM [*Indicator*]

Sets proprietary marking as follows:

Indicator	**Marking**
P	PRIVATE
N	NOTICE
No *Indicator* specified	Turns off proprietary marking.

.RE [*Number*]

Ends relative indent (**.RS**) at indent level position specified by the *Number* variable. If the *Number* variable value is omitted, return to the most recent lower indent level.

.RI *Character1Character2...*

Concatenates the Roman *Character1* with the italic *Character2*; alternate these two fonts up to six sets of *Character1Character2*. Similar macros alternate between any two of Roman, italic, and bold: the **.IR**, **.RB**, **.BR**, **.IB**, and **.BI** macros.

.RS [*Indent*]

Increases relative indent (initially zero). Indent all output an extra number of units from the left margin as specified by the *Indent* variable.

If the *Indent* variable is omitted, the previous *Indent* value is used. This value is set to its default (5 ens for the **nroff** command and 7.2 ens for the **troff** command) by the **.TH** format macro, **.P** format macro, and **.RS** format macro, and restored by the **.RE** format macro. The default unit for *Indent* is ens.

.SH [*Text*]

Places subhead text.

The *Text* variable represent up to six words; use " " (double quotation marks) to include character spaces in a word. If the variable is empty, this treatment is applied to the next input text line that contains text to be printed. For example, use the **.I** format macro to italicize an entire line, or use the **.SM** and **.B** format macros to produce an entire line of small−bold text. By default, hyphenation is turned off for the **nroff** command, but remains on for the **troff** command.

.SM [*Text*]

Makes text one point smaller than default point size.

The *Text* variable represent up to six words; use " " (double quotation

marks) to include character spaces in a word. If the variable is empty, this treatment is applied to the next input text line that contains text to be printed. For example, use the **.I** format macro to italicize an entire line, or use the **.SM** and **.B** format macros to produce an entire line of small−bold text. By default, hyphenation is turned off for the **nroff** command, but remains on for the **troff** command.

.SS [*Text*] Places sub−subhead text.

The *Text* variable represent up to six words; use " " (double quotation marks) to include character spaces in a word. If the variable is empty, this treatment is applied to the next input text line that contains text to be printed. For example, use the **.I** format macro to italicize an entire line, or use the **.SM** and **.B** format macros to produce an entire line of small−bold text. By default, hyphenation is turned off for the **nroff** command, but remains on for the **troff** command.

.TH [*Title*][*Section*][*Commentary*][*Name*]

Sets the title and entry heading. This macro calls the **.DT** format macro.

Variable	**Marking**
Title	Title
Section	Section number
Commentary	Extra commentary
Name	New manual name.

Note: If the **.TH** format macro values contain character spaces that are not enclosed in " " (double quotation marks), irregular dots are displayed on the output.

.TP [*Indent*] Begins indented paragraph with hanging tag. The next input line that contains text is the tag. If the tag does not fit, it is printed on a separate line.

If the *Indent* variable is omitted, the previous *Indent* value is used. This value is set to its default (5 ens for the **nroff** command and 7.2 ens for the **troff** command) by the **.TH** format macro, **.P** format macro, and **.RS** format macro, and restored by the **.RE** format macro. The default unit for *Indent* is ens.

Strings

***R** Adds trademark, (Reg.) for the **nroff** command and the registered trademark symbol for the **troff** command.

***S** Changes to default type size.

***(Tm** Adds trademark indicator.

Registers

IN Indent left margin relative to subheads. The default is 7.2 ens for the **troff** command and 5 ens for the **nroff** command.

LL Line length including the value specified by the **IN** register.

PD Current inter−paragraph distance.

Flags

−rs1 Reduces default page size of 8.5 inches by 11 inches with a 6.5−inch by 10−inch text area to a 6−inch by 9−inch page size with a 4.75−inch by 8.375−inch text area. This flag also reduces the default type size from 10−point to 9−point and the vertical line spacing from 12−point to 10−point.

Examples

1. To process the file `your.book` and pipe the formatted output to the local line printer, `qprt`, enter:

   ```
   nroff −Tlp −man your.book | qprt −dp
   ```

2. To process the files `my.book` and `dept.book`, which contain tables, and pipe the formatted output to the local line printer, `qprt`, enter:

   ```
   tbl my.book dept.book | nroff −Tlp −man | col −Tlp | qprt −dp
   ```

   Note: Before the output is sent to `qprt`, it is first filtered through the **col** command to process reverse linefeeds used by the **tbl** command.

3. To process the file `group`, which contains pictures, graphs, and tables, and prepare the formatted output for processing on the IBM 3816 printer, enter:

   ```
   grap group | pic | tbl | troff −Tibm3816 −man \
     | ibm3816 | qprt −dp
   ```

   Notes:

   1. If manual pages created with the **man** macro package are intended for an online facility, components requiring the **troff** command, such as the **grap** or **pic** command, should be avoided.
   2. The **grap** command precedes the **pic**command since it is a preprocessor to the **pic** command; the reverse does not format correctly.
   3. The **col** command is not required as a filter to the **tbl** command; typeset documents do not require reverse linefeeds.

me Macro Package for the nroff and troff Commands

The **me** package of the **nroff** and **troff** command macro definitions provides a formatting facility for technical papers in various formats. The **col** command may be required to postprocess **nroff** output in certain cases.

The macro requests are defined in the following section, in **me Requests**. Many **nroff**/**troff** requests can have unpredictable results in conjunction with this package. However, the following requests can be used after the first **.pp** request:

.bp	Begins new page.
.br	Breaks output line here.
.ce [*Number*]	Centers next specified number of lines. Default is 1 (one).
.ls [*Number*]	Sets line spacing. Text is single−spaced if *Number* is set to 1 (one);

	double−spaced if the value is set to 2.
.na	Leaves right margin unjustified.
.sp [*Number*]	Inserts the specified number of spacing lines.
.sz [+]*Number*	Adds the specified number to point size.
.ul [*Number*]	Underlines next specified number of lines. Default is 1 (one).

Output of the **eqn**, **neqn**, **refer**, and **tbl** commands preprocessors for equations and tables can be used as input.

me Requests

The following list contains all macros, strings, and number registers available in the **me** macros. Selected **troff** commands, registers, and functions are included.

\(space)	Defines unpaddable space (**troff** command built−in function).
\"	Comments to end of line (**troff** command built−in function).
***#**	Indicates optional delayed text tag string.
\$*Number*	Interpolates the value specified by the *Number* variable (**troff** command built−in function).
\n($0	Defines section depth (number register).
.$0	Started after section title printed (user−definable macro).
\n($1	Defines first section number (number register).
.$1	Started before printing depth 1 (one) section (user−definable macro).
\n($2	Defines second section number (number register).
.$2	Started before printing depth 2 section (user−definable macro).
\n($3	Defines third section number (number register).
.$3	Started before printing depth 3 section (user−definable macro).
\n($4	Defines fourth section number (number register).
.$4	Started before printing depth 4 section (user−definable macro).
\n($5	Defines fifth section number (number register).
.$5	Started before printing depth 5 section (user−definable macro).
\n($6	Defines sixth section number (number register).
.$6	Started before printing depth 6 section (user−definable macro).
.$C	Called at beginning of chapter (user−definable macro).
.$H	Indicates text header (user−definable macro).
\n($R	Defines relative vertical spacing in displays (number register defined by default; changing is not recommended).
\n($c	Defines current column header (number register).
.$c	Prints chapter title (macro defined by default; changing is not recommended).
\n($d	Indicates delayed text number (number register).
\n($f	Indicates footnote number (number register).
.$f	Prints footer (macro defined by default; changing is not recommended).
.$h	Prints header (macro defined by default; changing is not recommended).

\n($i	Defines paragraph base indent (number register).
\n($l	Defines column width (number register).
\n($m	Indicates number of columns in effect (number register).
***($n**	Indicates section name (string).
\n($p	Defines numbered paragraph number (number register).
.$p	Prints section heading (macro defined by default; changing is not recommended).
\n($r	Defines relative vertical spacing in text (number register defined by default; changing is not recommended).
\n($s	Defines column indent (number register).
.$s	Separates footnoter from text (macro defined by default; changing is not recommended).
\n%	Defines current page number (number register defined by default; changing is not recommended).
\&	Indicates zero−width character; useful for hiding controls (**troff** command built−in function).
\(*XX*	Interpolates special character specified by the *XX* variable (**troff** command built−in function).
.(b	Begins block (macro).
.(c	Begins centered block (macro).
.(d	Begins delayed text (macro).
.(f	Begins footnote (macro).
.(l	Begins list (macro).
.(q	Begins quote (macro).
.(x*Index*	Begins indexed item in the specified index (macro).
.(z	Begins floating keep (macro).
.)b	Ends block (macro).
.)c	Ends centered block (macro).
.)d	Ends delayed text (macro).
.)f	Ends footnote (macro).
.)l	Ends list (macro).
.)q	Ends quote (macro).
.)x	Ends index entry (macro).
.)z	Ends floating keep (macro).
***** *String*	Interpolates the value specified by the *String* variable (**troff** command built−in function).
***** *String1String2*	Interpolates the value specified by the *String1String2* variable (**troff** command built−in function).
******	Indicates optional footnote tag string.
.++ *mH*	Macro to define paper section. The value specified by the *m* variable defines the part of the paper. The *m* variable can have the following values: **C** Defines chapter. **A** Defines appendix. **P** Defines preliminary information, such as abstract and table of contents. **B** Defines bibliography.

RC Defines chapters to be renumbered from page 1 (one) of each chapter.

RA Defines appendix to be renumbered from page 1 (one).

The *H* parameter defines the new header. If there are any spaces in it, the entire header must be quoted. If you want the header to have the chapter number in it, use the string **\\\n(ch**. For example, to number appendixes A.1, A.2, ..., type: `.++ RA '''\\\n(ch.%'.` Each section (such as chapters and appendixes) should be preceded by the **.+c** request.

.+c*Title* Begins chapter (or appendix, for instance, as set by the **.++**macro). The value specified by the *Title* variable is the chapter title (macro).

***,** Indicates cedilla (string).

\− Indicates minus sign (**troff** command built−in function).

***−** Indicates 3/4 em dash (string).

\0 Defines unpaddable digit−width space (**troff** command built−in function).

.1c Reverts to single−column output (macro).

.2c Begins two−column output (macro).

***:** Indicates umlaut (string).

***<** Begins subscript (string).

***>** Ends subscript (string).

.EN Ends equation. Space after equation produced by the **eqn** command or **neqn** command (macro).

.EQ*XY* Begins equation; breaks out and adds space. The value specified by the *Y* variable is the equation number. The optional *X* variable value may be any of the following:

I Indents equation (default).

L Left−adjusts equation.

C Centers equation (macro).

\L'*Distance*' Indicates vertical line−drawing function for the specified distance (**troff** command built−in function).

.PE Ends pic picture (macro).

.PF Ends pic picture with flyback (macro).

.PS Starts pic picture (macro).

.TE Ends table (macro).

.TH Ends header of table (macro).

.TS *X* Begins table. If the value of the *X* variable is **H**, the table has a repeated heading (macro).

***[** Begins superscript (string).

\n(.$ Defines number of options to macro (number register defined by default; changing is not recommended).

\n(.i Indicates current indent (number register defined by default; changing is not recommended).

\n(.l Indicates current line length (number register defined by default; changing is not recommended).

\n(.s Indicates current point size (number register defined by default; changing is not recommended).

***(**4 Indicates acute accent (string).

***(`**	Indicates grave accent (string).
\(4	Indicates acute accent (**troff** command built−in function).
\(`	Indicates grave accent (**troff** command built−in function).
***]**	Ends superscript (string).
\^	Indicates 1/12 em narrow space (**troff** command built−in function).
***^**	Indicates caret (string).
.ac*AuthorNumber*	Sets up for ACM−style output. The *Author* variable specifies the author name or names. The *Number* variable specifies the total number of pages. Must be used before the first initialization (macro).
.ad	Sets text adjustment (macro).
.af	Assigns format to register (macro).
.am	Appends to macro (macro).
.ar	Sets page numbers in Arabic (macro).
.as	Appends to string (macro).
.b *X*	Prints in boldface the value specified by the *X* variable. If the *X* variable is omitted, boldface text follows (macro).
.ba +*Number*	Augments the base indent by the specified *Number* value. Sets the indent on regular text such as paragraphs (macro).
.bc	Begins new column (macro).
.bi *X*	Prints in bold italic the value specified by the *X* parameter, in no−fill mode only. If the *X* parameter is not used, bold italic text follows (macro).
\n(bi	Displays block indent (number register).
.bl	Requests blank lines, even at top of page (macro).
\n(bm	Sets bottom title margin (number register).
.bp	Begins page (macro).
.br	Sets break; starts new line (macro).
\n(bs	Displays block pre− or post−spacing (number register).
\n(bt	Blocks keep threshold (number register).
.bu	Begins bulleted paragraph (macro).
.bx *X*	Prints in no−fill mode only the value specified by the *X* variable in box (macro).
\c	Continues input (**troff** command built−in function).
.ce	Centers lines (macro).
\n(ch	Defines current chapter number (number register).
.de	Defines macro (macro).
\n(df	Displays font (number register).
.ds	Defines string (macro).
\n(dw	Defines current day of week (number register).
***(dw**	Defines current day of week (string).
\n(dy	Defines current day of month (number register).
\e	Indicates printable version of \ (backslash) (**troff** command built−in function).
.ef'*X*'*Y*'*Z*'	Sets even−page footer to the values specified by the *XYZ* variables (macro).
.eh'*X*'*Y*'*Z*'	

	Sets even−page header to the values specified by the *XYZ* variables (macro).
.el	Specifies the else part of an if/else conditional (macro).
.ep	Ends page (macro).
\n(es	Indicates equation pre− or post−space (number register).
\f*Font*	Sets inline font change to the specified *Font* variable value (**troff** command built−in function).
\f(*Fontf*	Sets inline font change to the specified *Fontf* variable value (**troff** command built−in function).
.fc	Sets field characters (macro).
\n(ff	Sets footnote font (number register).
.fi	Fills output lines (macro).
\n(fi	Indicates footnote indent, first line only (number register).
\n(fm	Sets footer margin (number register).
.fo '*X*'*Y*'*Z*'	Sets footer to the values specified by the *XYZ* variables (macro).
\n(fp	Sets footnote point size (number register).
\n(fs	Sets footnote pre−space (number register).
\n(fu	Sets footnote indent from right margin (number register).
\h'*Distance*'	Sets local horizontal motion for the specified distance (**troff** command built−in function).
.hc	Sets hyphenation character (macro).
.he '*X*'*Y*'*Z*'	Sets header to the values specified by the *XYZ* variables (macro).
.hl	Draws horizontal line (macro).
\n(hm	Sets header margin (number register).
.hx	Suppresses headers and footers on next page (macro).
.hy	Sets hyphenation mode (macro).
.i *X*	Italicizes the value specified by the *X* variable. If the *X*variable is omitted, italic text follows (macro).
.ie	Specifies the else part of an if/else conditional (macro).
.if	Designates a conditional (macro).
\n(ii	Sets indented paragraph indent (number register).
.in	Indents (transient); use the **.ba** macro if pervasive (macro).
.ip *X Y*	Starts indented paragraph, with hanging tag specified by the *X* variable. Indentation is the en value specified by the *Y* variable. Default is 5 (macro).
.ix	Indents, no break (macro).
\l'*Distance*'	Starts horizontal line−drawing function for the specified distance (**troff** command built−in function).
.lc	Sets leader repetition character (macro).
.lh	Interpolates local letterhead (macro).
.ll	Sets line length (macro).
.lo	Reads in a file of local macros of the form .*x. Must be used before initialization (macro).
.lp	Begins left−justified paragraph (macro).
***(lq**	Designates left quotation marks (string).

.ls	Sets multi−line spacing (macro).
.m1	Sets space from top of page to header (macro).
.m2	Sets space from header to text (macro).
.m3	Sets space from text to footer (macro).
.m4	Sets space from footer to bottom of page (macro).
.mc	Inserts margin character (macro).
.mk	Marks vertical position (macro).
\n(mo	Defines month of year (number register).
***(mo**	Defines current month (string).
\n*X*	Interpolates number register specified by the *X* variable value (number register).
\n(*XX*	Interpolates number register specified by the *XX* variable (number register).
.n1	Sets number lines in margin (macro).
.n2	Sets number lines in margin (macro).
.na	Turns off text adjustment (macro).
.ne*Number*	Sets the specified number of lines of vertical space (macro).
.nf	Leaves output lines unfilled (macro).
.nh	Turns off hyphenation (macro).
.np	Begins numbered paragraph (macro).
.nr	Sets number register (macro).
.ns	Indicates no−space mode (macro).
***o**	Indicates superscript circle (such as for Norse A; string).
.of'*X*'*Y*'*Z*'	Sets odd footer to the values specified by the *XYZ* variables (macro).
.oh'*X*'*Y*'*Z*'	Sets odd header to the values specified by the *XYZ* variables (macro).
.pa	Begins page (macro).
.pd	Prints delayed text (macro).
\n(pf	Indicates paragraph font (number register).
\n(pi	Indicates paragraph indent (number register).
.pl	Sets page length (macro).
.pn	Sets next page number (macro).
.po	Sets page offset (macro).
\n(po	Simulates page offset (number register).
.pp	Begins paragraph, first line indented (macro).
\n(pp	Sets paragraph point size (number register).
\n(ps	Sets paragraph pre−space (number register).
.q	Indicates quoted (macro).
***(qa**	For all (string).
***qe**	There exists (string).
\n(qi	Sets quotation indent; also shortens line (number register).
\n(qp	Sets quotation point size (number register).
\n(qs	Sets quotation pre− or post−space (number register).
.r	Sets Roman text to follow (macro).
.rb	Sets real bold font (macro).

.re	Resets tabs to default values (macro).
.rm	Removes macro or string (macro).
.rn	Renames macro or string (macro).
.ro	Sets page numbers in Roman (macro).
***(rq**	Indicates right quotation marks (string).
.rr	Removes register (macro).
.rs	Restores register (macro).
.rt	Returns to vertical position (macro).
\s*Size*	Changes inline size to specified size (**troff** command built−in function).
.sc	Reads in a file of special characters and diacritical marks. Must be used before initialization (macro).
\n(sf	Sets section title font (number register).
.sh*LevelTitle*	Indicates section head to follow; font automatically bold. The *Level* variable specifies the level of section. The *Title* variable specifies the title of section (macro).
\n(si	Sets relative base indent−per−section depth (number register).
.sk	Leaves the next page blank. Only one page is remembered ahead (macro).
.sm*X*	Sets, in a smaller point size, the value specified by the *X* variable (macro).
.so	Indicates source input file (macro).
\n(so	Sets additional section title offset (number register).
.sp	Indicates vertical space (macro).
\n(sp	Indicates section title point size (number register).
\n(ss	Indicates section prespace (number register).
.sx	Changes section depth (macro).
.sz *+Number*	Augments point size by the specified number of points (macro).
.ta	Sets tab stops (macro).
.tc	Sets tab repetition character (macro).
***(td**	Sets today's date (string).
n(tf	Indicates title font (number register).
.th	Produces paper in thesis format. Must be used before initialization (macro).
.ti	Indicates temporary indent, next line only (macro).
.tl	Indicates 3−part title (macro).
\n(tm	Sets top title margin (number register).
.tp	Begins title page (macro).
\n(tp	Sets title point size (number register).
.tr	Translates (macro).
.u *X*	Underlines the value specified by the *X* variable, even in the **troff** command. No−fill mode only (macro).
.uh	Sets section head to follow; font automatically bold. Similar to the **.sh** macro, but unnumbered (macro).
.ul	Underlines next line (macro).
\v'*Distance***'**	Local vertical motion for the specified distance (**troff** command built−in function).
***v**	Inverts v for Czech e (string).

\w'*String*'	Returns width of the specified string (**troff** command built−in function).
.xl	Sets local line length (macro).
.xp*Index*	Prints the specified index (macro).
\n(xs	Sets index entry prespace (number register).
\n(xu	Sets index indent, from right margin (number register).
\n(yr	Indicates year, last two digits only (number register).
\n(zs	Sets floating keep pre− or post−space (number register).
\{	Begins conditional group (**troff** command built−in function).
|	1/6 em, narrow space (**troff** command built−in function).
\}	Ends conditional group (**troff** command built−in function).
***~**	Indicates tilde (string).

For further information, see the *−ME Reference Manual* by E. P. Allman.

mm Macro Package for the mm, mmt, nroff, and troff Commands

The **mm** macro package provides macros to format text in a wide variety of document forms, such as memos, letters, and reports. The manner in which you type and edit a document is essentially independent of whether the document is later formatted at a terminal or phototypeset.

The **col** command may be required to postprocess **nroff** output. See the **col**command for specific requirements.

The **mm** macros and additional information are summarized under the following headings:

- Beginning Macros for Formal Memoranda
- Business Letter Macros
- Ending Macros (Trailing Information)
- Paragraphs
- Section Headings
- Lists
- Displays, Tables, Equations, and Footnotes
- Page Headers and Footers
- Miscellaneous Macros
- **mm** Registers
- **mm** Strings
- String Names
- Reserved Names.

Beginning Macros for Formal Memoranda

.ND *Date*	Sets new date.
.TL [*ChgNumber*] [*FileNumber*]	Sets title information. Text on the following line is used as the title of the document.
.AF [*CompanyName*]	Specifies author's company name.
.AU*Name* [*Initials*] [*Loc*] [*Dept*] [*Ext*] [*Room*] [*Option...*]	Sets author information.
.AT *AuthorTitle* [...]	Specifies title to follow signer's name (up to nine options).
.TM [*Number*]	Sets technical memorandum number.
.AS [**0** | **1** | **2**] [*Indent*]	Starts abstract, for technical memorandum and released paper only:

0 Abstract on cover sheet and first page

1 Abstract only on cover sheet

2 Abstract only on memorandum for file cover sheet.

.AE Ends abstract.

.NS Starts notation, allowed on memorandum for file cover sheets following an **.AS2/.AE** macro pair (see "Ending Macros").

.NE Ends notation, allowed on memorandum for file cover sheets following an **.AS 2/.AE** macro pair (see "Ending Macros").

.OK [*Keyword ...*] Specifies other keywords (up to nine options).

.MT [*type*] [*title*] Sets document type:

""	No type.
0	No type (internal letter).
1	Memorandum for file.
2	Programmer's notes.
3	Engineer's notes.
4	Released paper.
5	External letter.
"String"	The specified string is printed.

Title User−supplied text prefixed to page number

Business Letter Macros

.WA Starts writer's address.

.WE Ends writer's address.

.LO CN [*Notation*] Specifies confidential notation.

.LO RN [*Notation*] Specifies reference notation.

.IA Starts inside (recipient's) address.

.IE Ends inside (recipient's) address.

.LO AT [*Notation*] Specifies attention line.

.LO SA [*Notation*] Specifies salutation.

.LO SJ [*Notation*] Specifies subject line.

.LT [{ none **BL SB FB SP**}] Specifies business letter type:

none	Blocked
BL	Blocked
SB	Semiblocked
FB	Full−Blocked
SP	Simplified.

Ending Macros (Trailing Information)

.FC [*Closing*] Prints formal closing.

.SG [*Initials*] [**1**] Prints signature line.

.NS [{**" "0 1 2 3 4 5 6 7 8 9 10 11 12 13** *String*}] Starts notation:

" "

Copy to

0	Copy to
1	Copy (with attachment) to
2	Copy (without attachment) to
3	Attachment
4	Attachments
5	Enclosure
6	Enclosures
7	Under Separate Cover
8	Letter to
9	Memorandum to
10	Copy (with attachments) to
11	Copy (without attachments) to
12	Abstract Only to
13	Complete Memorandum to
String	Copy (*String*) to.

.NE Ends notation.

.AV *Name* [**1**] Prints approval signature.

.CS [*Pgs*] [*Other*] [*Tot*] [*Figs*] [*Tbls*] [*Ref*]
Prints cover sheet.

.TX Calls user exit for table−of−contents titles.

.TY Calls user exit for table−of−contents header.

.TC [*Slev*] [*Spacing*] [*Tlev*] [*Tab*] [*H1*] [*H2*] [*H3*] [*H4*] [*H5*]
Prints table of contents.

Paragraphs

.P [{**0 1 2**}] Starts paragraph:
0 Left−justified (default)

1 Indented

2 Indented except after **.H**, **.LE**, **.DE**.

Section Headings

.H {**1 2 3 4 5 6 7**} [*HeadingText*] [*FootnoteMark*]
Specifies numbered headings.

.HU *HeadingText*
Specifies unnumbered headings.

.HM {**1 0001 A a I i**}**...**
Specifies heading mark style:

- **1** Arabic
- **0001** Arabic with leading 0s (zeros)
- **A** Uppercase alphabetic
- **a** Lowercase alphabetic
- **I** Uppercase Roman
- **i** Lowercase Roman.

.HX [*Dlev*] [*Rlev*] [*HeadingText*]
Calls user−defined exit macro before headings.

.HY [*Dlev*] [*Rlev*] [*HeadingText*]
Calls user−defined exit macro in the middle of headings.

.HZ [*Dlev*] [*Rlev*] [*HeadingText*]
Calls user−defined exit macro after headings.

Lists

If the last option [**1**] is present in the list−start macros, there is no space between items.

.AL [{**1 A a I i**}] [*TextIndent*] [**1**]
Starts automatically incremented list (**1**).

.BL [*TextIndent*] [**1**]
Starts a bullet list.

.DL [*TextIndent*] [**1**]
Starts a dash list.

.ML *Mark* [*TextIndent*] [**1**]
Starts a list in which each list item is tagged with a specified mark. If the value of the *TextIndent* is **NULL** or omitted, it is set to [*Mark* − width + 1]. If the 3rd argument is specified, no blank lines separate items in the list.

.RL [*TextIndent*] [**1**]
Starts a reference list.

.VL *TextIndent* [*MarkIndent*] [**1**]
Starts a variable tag list.

.LI [*Mark*] [**1**]
Starts list item; **1** means that the *Mark* variable value is to be prefixed to the current mark.

.LE [**1**]
Ends list item; **1** means to output a blank line after list. The default is no blank line.

.LB *TextIndentMarkIndentPad Type* [*Mark*] [{**0 1**}] [{**0 1**}]
Begins list:
The value of the *Type* variable is:

1=. 2=) 3=() 4=[] 5=<> 6={}.

Sixth option:

0 No blank line before each list item.

Seventh option:

0 No blank line before list.

.LC [*Level***]** Clears list status up to the *Level* variable value.

Displays, Tables, Equations, and Footnotes

.DS [{**0 1 2 3** }] [{**0 1**}] [*Number*]

.DS [{**L I C CB**}] [{**N F**}] [*Number*]

Starts static display:

0 or **L**	No indent
1 or **I**	Indent from left
2 or **C**	Center each line
3 or **CB**	Center as a block
0 or **N**	No−fill
1 or **F**	Fill.
Number	Indent from right the number of spaces specified by the *Number* parameter.

.DF [{**0 1 2 3** }] [{**0 1**}] [*Number*]

.DF [{**L I C CB**}] [{**N F**}] [*Number*]

Starts floating display:

0 or **L**	No indent
1 or **I**	Indent from left
2 or **C**	Center each line
3 or **CB**	Center as a block
0 or **N**	No−fill
1 or **F**	Fill.
Number	Indent from right the number of spaces specified by the *Number* parameter.

.DE Ends display.

.FG [*Title*] [*Override*] [**0 1 2**]

The value of the *Override* variable replaces or enhances the default numbering. Specifies figure caption:

0	*Override* value is used as a prefix.
1	*Override* value becomes a suffix.

2 Replace *Override* value becomes a replacement.

.TS [**H**] Starts table:
H Multipage table.

.TH [**N**] Must be used when specifying option **H** to **.TS**:
N Suppresses table headers unless on top of new page.

.TE Ends table.

.TB [*Title*] [*Override*] [**0 1 2**]

The value of the *Override* variable replaces or enhances the default numbering. Specifies table caption:

0 *Override* value is used as a prefix.

1 *Override* value becomes a suffix.

2 Replace *Override* value becomes a replacement.

.EX [*Title*] [*Override*] [**0 1 2**]

The value of the *Override* variable replaces or enhances the default numbering. Specifies exhibit caption:

0 *Override* value is used as a prefix.

1 *Override* value becomes a suffix.

2 Replace *Override* value becomes a replacement.

.EQ [*Label*] Starts equation display using the specified label.

.EN Ends equation display.

.EC [*Title*] [*Override*] [**0 1 2**]

The value of the *Override* variable replaces or enhances the default numbering. Specifies equation caption:

0 *Override* value is used as a prefix.

1 *Override* value becomes a suffix.

2 Replace *Override* value becomes a replacement.

.FS [*Label*] Starts footnote using the specified label as an indicator. Default is numbered footnote.

.FE Ends footnote.

.FD [{**0 1 2 3 4 ... 11**}] [**1**] Sets footnote format:

First option:

Set up formatting style for footnote text. Default is 0 for **mmt** command. Default is 10 for **mm** command. See the following table for the value.

Second option:

Reset footnote counter on first−level heading.

.FD Arg.	Hyphens	Adjusted	Text Indented	Label Justified
0	**.nh**	**.ad**	Yes	Left
1	**.hy**	**.ad**	Yes	Left
2	**.nh**	**.na**	Yes	Left
3	**.hy**	**.na**	Yes	Left
4	**.nh**	**.ad**	No	Left
5	**.hy**	**.ad**	No	Left
6	**.nh**	**.na**	No	Left
7	**.hy**	**.na**	No	Left
8	**.nh**	**.ad**	Yes	Right
9	**.hy**	**.ad**	Yes	Right
10	**.nh**	**.na**	Yes	Right
11	**.hy**	**.na**	Yes	Right

Page Headers and Footers

.PH "'*Left*'*Center*'*Right*'"

Specifies page header.

.OH "'*Left*'*Center*'*Right*'"

Specifies odd−page header.

.EH "'*Left*'*Center*'*Right*'"

Specifies even−page header.

.PF "'*Left*'*Center*'*Right*'"

Specifies page footer.

.OF "'*Left*'*Center*'*Right*'"

Specifies odd−page footer.

.EF "'*Left*'*Center*'*Right*'"

Specifies even−page footer.

.BS Starts bottom−block.

.BE Ends bottom−block.

.PX Calls user exit for page−header.

.TP Calls top of page macro.

Miscellaneous Macros

.B [*Option*] [*Prev−Font−option*]

Prints in bold (up to six options).

.I [*Option*] [*Prev−Font−option*]

Prints in italics (up to six options); underlines with the **nroff** command.

.R Returns to Roman font.

.PM [*Option*] Sets proprietary marking. If you do not give the **.PM** macro an option, you turn off proprietary markings. The **/usr/lib/macros/string.mm** file contains some proprietary markings. This file should be edited to meet the user's needs.

.RD [*Prompt*] [*Diversion*] [*String*]

Stops code macro. The *Prompt* variable should be a user−defined string without spaces. The *Diversion* variable allows the typed−in text to be saved. The *String* variable contains the first line typed following the prompt.

.RP [{**0 1** }] [{**0 1 2 3**}]

Produces reference page:

First option:

0	Resets reference counter (default).
1	Does not reset reference counter.

Second option:

0 Causes an **.SK** macro after (default).

1 Does not cause an **.SK** macro after.

2 Does not cause an **.SK** macro before.

3 Does not cause an **.SK** macro before or after.

.RS/.RF Numbers references automatically.

.WC [{**N WF −WF FF −FF WD −WD FB −FB**}]

Controls width for footnotes and displays when using two columns:

N	Normal mode (**−WF**, **−FF**, **−WD**).
WF	Footnotes always wide.
−WF	Footnotes follow page style.
FF	First footnote determines width of remaining footnotes on that page.
−FF	Footnotes follow setting of **WF** or **−WF** option.
WD	Always wide displays.
−WD	Displays follow page style.
FB	Floating display causes page break (default).
−FB	Floating display does not cause page break.

.SP [*Lines*] Skips lines down.

.SK [*Number*] Skips the specified number of pages. (The default is 1.)

.OP Breaks to an odd page.

.2C Prints output in two columns.

.1C Prints output in one column (normal line width restored).

.SA [*Option*] Sets right−margin justification

Options:

0 Sets default to **off** (default for the **nroff** command).
1 Sets default to **on** (default for the **troff** command).

If no option is specified, macro reverts to current default.

.SM *String1* [*String2*] [*String3*]

Reduces size of the *String1* variable value by 1 point if the *String3* variable value is omitted; otherwise, reduces size of the *String2* variable value by one point.

.HC *Character* Sets hyphenation character to the *Character* variable value.

.S [*PointSize*] [*VerticalSpacing*]

Sets point size and vertical spacing (the **troff** command only).

Defaults:

Point size = **10p**

Vertical spacing = **12p**

Options 1 and 2:

Number	New value.
+/−Number	Increment to current value.
D	Default.
C	Current value.
P	Previous value.

.VM [*Top*] [*Bottom*] Sets variable vertical margins.

.nP Starts double−line indent on paragraph.

The following macros are for alternating fonts and all take one to six options:

.IB Alternates italics (underlines for **nroff**) and bold.
.BI Alternates bold and italics.
.RI Alternates Roman and italics.
.IR Alternates italics (underlines for **nroff**) and Roman.
.RB Alternates Roman and bold.
.BR Alternates bold and Roman.

mm Registers

If an * (asterisk) follows a register name, that register can be set one of two ways: from the command line (see the example in the **mm** command) or before the formatter reads **mm** macro definitions. In the following list, the number shown in parentheses is the default value.

A * Handle preprinted forms.
Au Inhibit author information on first page (1).

C * Copy type (such as Original and Draft) (0).

Cl Contents level (2).

Cp Placement of figures, tables, equations, and exhibits (1).

D * Debug flag (0). If set to 1, the **mm** command continues even if it encounters a normally fatal error.

De Eject page after floating displays (0).

Df If set to 1, format register for floating displays (5).

Ds Static display pre− and post−space (1).

E * Control font of the Subject/Date/From fields (0): 0 = bold; 1 = Roman.
0 Bold (0)
1 Roman.

Ec Equation counter.

Ej Page−ejection flag for headings (0).

Eq Equation label placement (0).

Ex Exhibit counter.

Fg Figure counter.

Fs Vertical footnote separation (1).

H1...H7 Heading counters.

Hb Heading break level (after **.H** and **.HU**) (2).

Hc Heading centering level for **.H** and **.HU** (0).

Hi Heading temporary indent (after **.H** and **.HU**) (1).

Hs Heading space level (after **.H** and **.HU**) (2).

Ht Heading type:
0 Concatenated numbers (0)
1 Single numbers (0).

Hu Heading level for unnumbered heading (2).

Hy Hyphenation control:
0 No hyphenation (0)
1 Enable hyphenation.

L * Length of page (66v).

Le List of equations following table of contents (0):
0 Do not print
1 Print.

Lf List of figures following table of contents (0):
0 Do not print
1 Print.

Li List indent (5, **troff** command); (6, **nroff** command).

Ls List level down to which there is spacing between items (6).

Lt List of tables following table of contents (0):
0 Do not print
1 Print

Lx List of exhibits following table of contents (1):
0 Do not print
1 Print.

N * Numbering style (0).

Np Numbered paragraphs:

0 Unnumbered

1 Numbered (0).

O * Offset of page.

Oc Page numbering style for table of contents:

0 Lowercase Roman

1 Arabic (0).

Of Figure caption style (0).

P Page number; managed by the **mm** command (0). The register accepts a value of 0, or positive integers.

Pi Paragraph indent (5).

Ps Paragraph spacing (1).

Pt Paragraph type (0).

Pv PRIVATE header:

0 Do not print PRIVATE

1 On first page only

2 On all pages (0).

Rf Reference counter; used by **.RS** macro.

S * The **troff** command's default point size (10).

Si Display indent (5).

T * Type of the **nroff** command output device (0).

Tb Table counter.

U * Underlining style (the **nroff** command) for **.H** and **.HU** (0).

W * Width of page (line and title length).

mm Strings

Print special strings by using the following escape sequences:

***x** For strings with single−character names (**x**)

***(xx** For strings with two−character names (**xx**).

String Names

BU Bullet.

Ci Indent of heading levels in the table of contents.

DT Current date. The locale−specific date format specified by the locale setting for the **LC_TIME** category is used as the default setting. This corresponds to the **%x** format specifier of the **strftime** subroutine. Use the **.ND** macro to change the current date.

EM Em dash.

F Footnote numbering.

HF Heading level font string:

1 Roman

2 italics

3 Bold (2 2 2 2 2 2 2).

HP Point sizes of the various heading levels.

Le Title of the list of equations.

Lf Title of the list of figures.

Lt Title of the list of tables.
Lx Title of the list of exhibits.
RE SCCS SID of **mm** macros.
Rf Reference numberer.
Rp Title of the reference page.
Tm Trademark.
` Grave accent.
' Acute accent.
^ Circumflex.
~ Tilde.
: Lowercase umlaut.
; Uppercase umlaut.
, Cedilla.

Reserved Names

If you define your own strings, macros, and registers, use only names that consist of either a single lowercase letter, or a lowercase letter followed by any character other than a lowercase letter. The names **c2** and **nP** are exceptions to this; they are reserved.

mptx Macro Package for the nroff and troff Commands

The **mptx** macro package provides a definition for the **.xx** macro that is used for formatting a permuted index produced by the **ptx** command. The **mptx** macro package does not provide any other formatting capabilities, such as headers and footers. Use the **mptx** macro package in conjunction with the **mm** macro package if such capabilities are required. In this case, call the **−mptx** option after the **−mm** call, as follows:

nroff −mm −mptx *File...* | *Printer*

ms Macro Package for the nroff and troff Commands

The ms macro package of **nroff** and **troff** command macro definitions provides a formatting facility for various styles of articles, theses, and books. In certain cases, the **col** command may be required to postprocess output.

The macro requests are defined in the **ms** Requests section. Many **nroff** and **troff** command requests can have unpredictable results in conjunction with this package. However, the first 4 requests in the following list can be used after initialization, and the last 2 requests can be used before initialization.

.bp Begins new page.
.br Breaks output line.
.ce [*Number*]
Centers the next specified number of lines.
.ls [*Number*]
Sets line spacing. Set the value of the *Number* variable to 1 (one) to single−space text; and to 2 to double−space text.
.na Turns off alignment of right margin.
.sp [*Number*]
Inserts the specified number of spacing lines.

Font and point−size changes with the**\f** and **\s** macros are also allowed. For example, `\fIword\fR` italicizes `word`. Output of the **tbl**, **eqn**, and **refer** command preprocessors for equations, tables, and references is acceptable as input.

Formatting distances can be controlled in **ms** macros by means of built−in number registers. For example, the following number register sets the line length to 6.5 inches:

```
.nr LL 6.5i
```

For more information on **ms** macro registers, see ms Registers.

ms Requests

Following are external **ms** macro requests:

.AB [**X**] Begins abstract. If **X** is no, do not label abstract.

Initial Value: −

Break: **yes**

.AE Ends abstract.

Break: **yes**Initial Value: −

Break: **yes**

.AI*Name* Author's institution.

Initial Value: −

Break: **yes**

.AM Sets accent mark definitions.

Initial Value: −

Break: **no**

.AU*Name* Sets author's name.

Initial Value: −

Break: **yes**

.B [**X**] Puts **X** in boldface. If no **X**, switches to boldface.

Initial Value**:** −

Break: **no**

.B1 Begins text to be enclosed in a box.

Initial Value: −

Break: **yes**

.B2 Ends boxed text and prints it.

Initial Value: −

Break: **yes**

.BT Prints bottom title at foot of page.

Initial Value: **date**

Break: **no**

.BX X Prints word **X** in a box.

Initial Value: −

Break: **no**

.CM Cuts mark between pages.

Initial Value: **if t**

Break: **no**

.CT Indicates chapter title; page number moved to CF (TM).

Initial Value: −

Break: **yes**

Reset: **yes**

.DA [**X**] Forces date **X** at bottom of page. If no **X**, date is today.

Initial Value: **if n**

Break: **no**

.DE Ends display (unfilled text) of any kind.

Initial Value: −

Break: **yes**

.DS X Y Begins display with keep. **X**=I, L, C, B; **Y**=indent.

Initial Value: **I**

Break: **yes**

.ID Y Indents display with no keep; **Y**=indent.

Initial Value: **8n, .5i**

Break: **yes**

.LD Sets left display with no keep.

Initial Value: −

Break: **yes**

.CD Centers display with no keep.

Initial Value: −

Break: **yes**

.BD Block display; centers entire block.

Initial Value: −

Break: **yes**

.EF X Sets even page footer **X** (3 part as for **troff** command, **.tl** request).

Initial Value: −

Break: **no**

.EH X Sets even page header **X** (3 part as for **troff** command, **.tl** request).

Initial Value: −

Break: **no**

.EN Ends displayed equation produced by **eqn** command.

Initial Value: −

Break: **yes**

.EQ [**X**] [**Y**] Breaks out equation. **X**=L, I, C; **Y** is equation number.

Initial Value: −

Break: **yes**

.FE Ends footnote to be placed at bottom of page.

Initial Value: −

Break**: no**

.FP Numbers footnote paragraph; can be redefined.

Initial Value: −

Break: **no**

FS [**X**] Starts footnote; **X** is optional footnote label.

Initial Value: −

Break: **no**

.HD Sets optional page header below header margin.

Initial Value: **undef**

Break: **no**

.I [**X**] Italicizes **X**. If no **X**, equivalent to italics font **.ft 2**.

Initial Value: −

Break: **no**

.IP X Y Indents paragraph, with hanging tag **X**. **Y** specifies spaces to indent.

Initial Value: −

Break: **yes**

Reset: **yes**

.IX X Y Indexes words such as **X** and **Y**, up to five levels.

Initial Value: −

Break: **yes**

.KE Ends keep of any kind.

Initial Value: −

Break: **no**

.KF Begins floating keep; text fills remainder.

Initial Value: −

Break: **no**

.KS Begins keep; keeps unit together on a single page.

Initial Value: −

Break: **yes**

.LG Sets larger type size; increases point size by 2. Valid only for the **troff** command.

Initial Value: −

Break: **no**

.LP Begins left block paragraph.

Initial Value: −

Break: **yes**

Reset: **yes**

.MC X Sets multiple columns. **X** is column width.

Initial Value: −

Break: **yes**

Reset: **yes**

.ND [**X**] Indicates no date in page footer; **X** is date on cover.

Initial Value: **if t**

Break: **no**

.NH X Y Sets numbered header: **X**=level; **X**=0, resets; **X**=S, sets to **Y**.

Initial Value: −

Break: **yes**

Reset: **yes**

.NL Sets point size back to default. Valid for the **troff** command only.

Initial Value: **10p**

Break: **no**

.OF X Sets odd page footer **X** (3 part as for **me** macro, **.tl**request).

Initial Value: −

Break: **no**

.OH X Sets odd page header **X** (3 part as for **me** macro, **.tl**request).

Initial Value: −

Break: **no**

.P1 Prints header on first page.

Initial Value: **if TM**

Break: **no**

.PP Indents first line of paragraph.

Initial Value: −

Break: **yes**

Reset: **yes**

.PT Prints page title at head of page.

Initial Value: **%**

Break: **no**

.PX X Prints index (table of contents); **X**=do not suppress title.

Initial Value: −

Break: **yes**

.QP Quotes paragraph (indented and shorter).

Initial Value: −

Break: **yes**

Reset: **yes**

.R [**X**] Returns to Roman font. Prints in Roman font. If **X** is missing, equivalent to font **.ft1**.

Initial Value: **on**

Break**: no**

.RE Retreats (end level of relative indentation). Used with the **.RS** request.

Initial Value: **5n**

Break: **yes**

Reset: **yes**

.RP [**X**] Prints title page in released paper format; **X**=no, stops title on first page.

Initial Value: −

Break: **no**

.RS Right−shifts in one indentation level (start level of relative indentation). Used with the **.IP** request.

Initial Value: **5n**

Break: **yes**

Reset: **yes**

.SG Sets signature line.

.SH Sets unnumbered section header (in boldface).

Initial Value: −

Break: **yes**

Reset: **yes**

.SM Sets smaller type size; decrease point size by 2. Valid for the **troff** command only.

Initial Value: −

Break: **no**

.TA Sets tabs to 8n, 16n, ... (**nroff**); 5n, 10n, ... (**troff**).

Initial Value: **8n**, **5n**

Break: **no**

.TC X Prints table of contents at end; **X**=do not suppress title.

Initial Value: −

Break: **yes**

.TE Ends table processed by **tbl** command.

Initial Value: −

Break: **yes**

.TH Ends multipage header of table. Must be used with the **.TS** H request.

Initial Value: −

Break: **yes**

.TL Sets title line (in boldface and 2 points larger).

Initial Value: −

Break: **yes**

.TM Sets UC Berkeley thesis mode.

Initial Value: **off**

Break: **no**

.TS X Begins table. If **X** is H, table prints header on all pages.

Initial Value: −

Break: **yes**

Reset: **yes**

.UL X Underlines **X**, even for the **troff** command.

Initial Value: −

Break: **no**

.UX X Sets UNIX; trademark message first time; **X** appended.

Initial Value: −

Break: **no**

.XA X Y Sets another index entry; **X**=page; **X**=no, for none.

Initial Value: −

Break: **yes**

.XE Ends index entry or series of **.IX** request entries.

Initial Value: −

Break: **yes**

.XP Exdents first line of paragraph; others indented.

Initial Value: −

Break: **yes**

Reset: **yes**

.XS X Y Begins index entry; **X**=page; **X**=no, for none; **Y**=indent.

Initial Value: −

Break: **yes**

.1C Begins one−column format, on a new page.

Initial Value: **on**

Break: **yes**

Reset: **yes**

.2C Begins two−column format.

Initial Value: −

Break: **yes**

Reset: **yes**

.]− Sets beginning of **refer** command reference.

Initial Value: −

Break: **no**

.[0 Sets end of unclassifiable type of reference.

Initial Value: −

Break: **no**

.[N For journal article, **N**=1 (one). For book, **N**=2. For book article, **N**=3.

Initial Value: −

Break: **no**

ms Registers

Following is a list of number registers and their default values:

PS Sets point size. Takes effect for paragraph. Default is 10.
VS Sets vertical spacing. Takes effect for paragraph. Default is 12.
LL Sets line length. Takes effect for paragraph. Default is 6i.
LT Sets title length. Takes effect on next page. Defaults to the **LL** register value.
FL Sets footnote length. Takes effect at next **.FS** request. Default is 5.5i.
PD Sets paragraph distance. Takes effect for paragraph. Default is 1v (in **nroff**), .3v (in **troff**).
DD Sets display distance. Takes effect for displays. Default is 1v (in **nroff**), .5v (in **troff**).
PI Sets paragraph indent. Takes effect for paragraph. Default is 5n.
QI Sets quotation indent. Takes effect at next **.QP** request. Default is 5n.
FI Sets footnote indent. Takes effect at next **.FS** request. Default is 2n.
PO Sets page offset. Takes effect on next page. Default is 0 (zero) (in **nroff**), 1i (in **troff**).
HM Sets header margin. Takes effect on next page. Default is 1i.
FM Sets footer margin. Takes effect on next page. Default is 1i.
FF Sets footnote format. Takes effect at next **.FS** request. Default is 0 (zero) (1, 2, 3 available).

When resetting number register values, make sure to specify the appropriate units. Set the line length to 7i instead of just 7, which would result in output with one character per line. Setting the **FF** register to 1 (one) suppresses footnote superscripting. Setting it to 2 also suppresses indentation of the first line. Setting the **FF** register to 3 produces a footnote paragraph like the .**IP** request.

Following is a list of string registers available in the **ms** macros. These string registers can be used anywhere in the text.

***Q** Open quotation marks (**"** in **nroff**; **` `** in **troff**)
***U** Close quotation marks (**"** in **nroff**; **' '** in**troff**)
***−** Dash (**−−** in **nroff**; **−** in **troff**)
***(MO** Month of year
***(DY** Day (current date)
****** Automatically numbered footnote

*' Acute accent (before letter)
*` Grave accent (before letter)
*^ Circumflex accent (before letter)
*, Cedilla (before letter)
*: Umlaut (before letter)
*~ Tilde (before letter).

When using the extended accent mark definitions available with the **.AM** request, these strings should come after, rather than before, the letter to be accented.

Notes:

1. It is important to note that floating keeps and regular keeps are diverted to the same space, so they cannot be mixed.
2. The date format is restricted to U.S. English format.

mv Macro Package for the mvt and troff Commands

This package simplifies the typesetting of view graphs and projection slides in a variety of sizes. Although a few macros accomplish most of the formatting tasks needed in making transparencies, the entire facilities of the **troff**, **tbl**, **pic**, and **grap** commands are available for more difficult tasks.

The output can be previewed on most terminals, in particular the Tektronix 4014. For this device, specify the **−rX1** flag (which is automatically specified by the **mvt** command when that command is called with the **−D4014** flag). To preview output on other terminals, specify the **−a** flag.

The **mv** macros are summarized under the following headings:

- Foil−Start Macros
- Level Macros
- Text−Control Macros
- Default−Setting Macros.

Foil−Start Macros

For the following nine macros, the first character of the name (**V** or **S**) distinguishes between view graphs and slides, respectively, while the second character indicates whether the foil is square (**S**), small wide (**w**), small high (**h**), big wide (**W**), or big high (**H**). Slides are narrower than the corresponding view graphs. The ratio of the longer dimension to the shorter one is larger for slides than for view graphs. As a result, slide foils can be used for view graphs, but view graphs cannot be used for slide foils. On the other hand, view graphs can accommodate a bit more text.

.VS [*FoilNumber*] [*FoilID*] [*Date*]

Starts a square view graph. Foil size is to be 7 inches by 7 inches. The foil−start macro resets all variables (such as indent and point size) to initial default values, except for the values of the *FoilID* and *Date* variables inherited from a previous foil−start macro. The **.VS**macro also calls the **.A** macro.

.Vw, **.Vh**,**.VW**, **.VH**, **.Sw**, **.Sh**, **.SW**, **.SH**

Same as the **.VS** macro, except that these macros start view graphs (**V**) or slides (**S**) that are small wide (**w**), small high (**h**), large wide (**W**), or large high (**H**).

The following macros are recommended:

- **.VS** for square view graphs and slides
- **.Sw** (and, if necessary, **.Sh**) for 35mm slides.

.Vw [*FoilNumber*] [*FoilID*] [*Date*]

Same as the **.VS** macro, except that foil size is 7 inches wide by 5 inches high.

.Vh [*FoilNumber*] [*FoilID*] [*Date*]

Same as the **.VS** macro, except that foil size is 5 inches wide by 7 inches high.

.VW [*FoilNumber*] [*FoilID*] [*Date*]

Same as the **.VS** macro, except that foil size is 7 inches wide by 5.4 inches high.

.VH [*FoilNumber*] [*FoilID*] [*Date*]

Same as the **.VS** macro, except that foil size is 7 inches wide by 9 inches high.

.Sw [*FoilNumber*] [*FoilID*] [*Date*]

Same as the **.VS** macro, except that foil size is 7 inches wide by 5 inches high.

.Sh [*FoilNumber*] [*FoilID*] [*Date*]

Same as the **.VS** macro, except that foil size is 5 inches wide by 7 inches high.

.SW [*FoilNumber*] [*FoilID*] [*Date*]

Same as the **.VS** macro, except that foil size is 7 inches wide by 5.4 inches high.

.SH [*FoilNumber*] [*FoilID*] [*Date*]

Same as the **.VS** macro, except that foil size is 7 inches wide by 9 inches high.

Note: The **.VW** and **.SW** foils are meant to be 9 inches wide by 7 inches high. However, because the typesetter paper is generally only 8 inches wide, **.VW** and **.SW** foils are printed 7 inches wide by 5.4 inches high and have to be enlarged by a factor of 9/7 before use as view graphs.

Level Macros

.A [*X*] Places text that follows at the first indentation level (left margin). The presence of the *X* variable suppresses the half−line spacing from the preceding text.

.B [*Mark* [*Size*]] Places text that follows at the second indentation level. Text is preceded by a specified mark (default is a large bullet). The *Size* variable is the increment or decrement to the point size of the mark with respect to the *prevailing* point size (default is 0). A value of 100 for the *Size* variable makes the point size of the mark equal to the default value of the *Mark* variable.

.C [*Mark* [*Size*]] Same as the **.B** macro, but for the third indentation level. The default value of the *Mark* variable is an em dash.

.D [*Mark* [*Size*]] Same as the **.B** macro, but for the fourth indentation level. The default value of the *Mark* variable is a small bullet.

Text−Control Macros

.I [+/−] [*Indentation*] [A[X]]	Changes the current text indent (does not affect titles). The specified indentation is in inches unless dimensioned. The default is 0. If the *Indentation* variable is signed, it is an increment or decrement. The presence of the *A* variable calls the **.A** macro and passes the *X* variable (if any) to it.
.S [*Size*] [*Length*]	Sets the point size and the line length. The value specified in the *Size* variable is the point size (default is previous). If the *Size* variable value is 100, the point size reverts to the *initial* default for the current foil−start macro. If the *Size* variable is signed, it is an increment or decrement (default is 18 for the **.VS**, **.VH**, and **.SH** macros, and 14 for the other foil−start macros). The *Length*variable specifies the line length (in inches unless dimensioned; the default is 4.2 inches for the **.Vh** macro, 3.8 inches for the **.Sh** macro, 5 inches for the **.SH** macro, and 6 inches for the other foil−start macros).
.T*String*	Prints the *String* variable value as a centered, enlarged title.
.U*String1*[*String2*]	Underlines the *String1* variable value and concatenates the *String2 variable* value (if any) to it. Using this operation is not recommended.

Default−Setting Macros

.DF [*Number Name*]**...**	Sets font positions. It cannot be displayed within foil input text; that is, it must follow the input text for a foil, but it must precede the next foil−start macro. The specified number is the position of the font specified by the *Name* variable. The **.DF** macro takes up to four pairs of *Number Name* variables, such as `1 H`. The first *Name* variable specifies the prevailing font. For example: **.DF 1 H 2 I 3 B 4 S**.
.DV [*A*] [*B*] [*C*] [*D*]	Alters the vertical spacing between indentation levels. The value specified by the *A*, *B*, *C*, or *D* variable is the spacing for the **.A**, **.B**, **.C**, or **.D** macro, respectively. All non−null parameters must be dimensioned. Null parameters leave the corresponding spacing unaffected. The default setting is: **.DV .5v .5v .5v 0v**.

The **.S**, **.DF**, **.DV**, and **.U** macros do not cause a break. The **.I** macro causes a break only if it is called with more than one variable. All the other macros cause a break.

The **mv** macro package also recognizes the following uppercase synonyms for the following corresponding lowercase **troff** command requests:

- **.AD**
- **.BR**
- **.CE**
- **.FI**
- **.HY**
- **.NA**
- **.NF**
- **.NH**
- **.NX**
- **.SO**
- **.SP**
- **.TA**
- **.TI**

The **Tm** string produces the trademark symbol.

Environment Variable

LANG Determines the locale's equivalent of y for yes or no queries. The allowed affirmative responses are defined in the locale variable **YESSTR**. If **LANG** is not set, or if it is set to an empty string, the **YESSTR** from the default C locale is used.

nroff and troff Requests for the nroff and troff Commands

The following **nroff** and **troff**requests are included in a specified working file or in standard input. The **nroff** and **troff** requests control the characteristics of the formatted output when the file or standard input is processed with the **nroff** or **troff** commands. The **nroff** and **troff**requests are grouped by function, in the following sections:

- Numerical Parameter Input
- Font and Character Size Control
- Page Control
- Text Filling, Adjusting, and Centering
- Vertical Spacing
- Line Length and Indenting
- Macros, Strings, Diversions, and Position Traps
- Number Registers
- Tabs, Leaders, and Fields
- Input and Output Conventions and Character Translations
- Hyphenation
- Three−Part Titles
- Output Line Numbering
- Conditional Acceptance of Input
- Environment Switching
- Insertions from Standard Input
- Input and Output File Switching
- Miscellaneous

For number variables written as *+Number*, the variable can be expressed as follows:

- The *Number* variable by itself is an absolute value.
- The *+Number* variable increases the currently set value.
- The *−Number* variable decreases the variable relative to its current value.

> **Note:** For all numeric parameters, numbers are expressed using ASCII Arabic numerals only.

The notes at the end of this article are referenced in the specific requests where applicable.

Numerical Parameter Input

Both **nroff** and **troff** requests accept numerical input with the appended scale indicators shown in the following table, where *S* is the current type size in points, *V* is the current vertical line spacing in basic units, and *C* is a nominal character width in basic units.

Scale		**Number of Basic Units**	
Indicator	**Meaning**	**troff**	**nroff**
i	Inch	machine−	240
c	Centimeter	dependent	240x50/127
P	Pica = 1/6 inch		240/6

m	Em = *S* points	*C*
n	En = Em/2	*C (same as Em)*
p	Point = 1/72 inch	240/72
u	Basic unit	1
v	Vertical line space	*V*
***k**	Width single−width kana	*C*
****K**	Width double−width kanji	Two *C*s
none	Default	

* If a non−kanji output device is selected, an en−width is used instead.

** If a non−kanji output device is selected, an em−width is used instead.

In the **nroff** request, both the em and the en are taken to be equal to the *C*, which is output−device dependent; frequent values are 1/10 and 1/12 inch. Actual character widths in the **nroff** request need not be all the same, and characters constructed with predefined strings such as − > are often extra wide.

Japanese Language Support: In the output from the **nroff** command, all double−width Japanese characters such as all kanji and some katakana characters have a fixed width equal to two *C*s. All single−width Japanese characters such as some katakana characters have a fixed width equal to *C*.

The scaling for horizontally−oriented control characters, vertically oriented control characters, and the requests **.nr**, **.if**, and **.ie** are as follows:

Orientation	Default Measure	Request or Function
Horizontal	Em (**m**)	**.ll**, **.in**, **.ta**, **.lt**, **.po**, **.mc**,**\h**, **\l**
Vertical	Vertical line space (**v**)	**.pl**, **.wh**, **.ch**, **.dt**, **.sp**, **.sv**, **.ne**, **.rt**, **\v**, **\x**, **\L**
Register−oriented or Conditional	Basic unit (**u**)	**.nr**, **.if**, **.ie**
Miscellaneous	Point (**p**)	**.ps**, **.vs**, **\H**, **\s**

All other requests ignore scale indicators. When a number register containing an already appropriately scaled number is interpreted to provide numerical input, the unit scale indicator **u** may need to be appended to prevent an additional inappropriate default scaling. The *Number* may be specified in decimal−fraction form, but the parameter that is finally stored is rounded to an integer number of basic units.

Font and Character Size Control

.bd *Font Number*

Makes the characters in the specified font artificially bold by overstriking them the specified number of times when using **nroff**, or by printing each character twice separated by *Number* −1 basic units when using **troff**. If the *Number* variable is not specified, the bold mode is turned off. The *Font* value must be an ASCII font name or font position. For the **nroff** command, the default setting of the **.bd** request is 3 3, specifying that characters on the font mounted at position 3 (normally bold) are to be overstruck 3 times (that is, printed in place a total of 4 times).

The font name itself can be substituted for the font position; for example, **.bdI 3**. The *Number* variable is functionally identical to the **−u** flag of the **nroff** command. (The bold mode must be in effect when the characters are physically printed.) This request can affect the contents of the **.b** general−number register.

The bold mode still must be in effect, or restarted at the time of physical output. You cannot turn off the bold mode in the **nroff** command if it is being controlled locally by the printing device as with, for example, a DASI 300.

Initial Value: Off

If No Value Specified: −

.bd S *Font Number*

Makes the characters in the special font bold whenever the specified font is the current font. The mode must be in effect when the characters are physically printed. The *Font* value must be an ASCII font name or font position. The mode still must be in effect, or again so, at the time of physical output.

Initial Value: Off

If No Value Specified: −

.cs *Font Number M*

Sets constant character space (width) mode to the *Font* variable value (if mounted). The width of every character is taken to be the value specified in the *Number* variable divided by 36 ems. If the *M* variable is not specified, the em width is that of the character's point size; if the *M* variable is given, the width is the value specified by the *M* variable minus points. All affected characters are centered in this space, including those with an actual width larger than this space. Special font characters occurring while the specified font is the current font are also so treated. The *Font* value must be an ASCII font name or font position. If the *Number* variable is absent, the mode is turned off. The mode must be in effect when the characters are physically printed. This request is ignored by the **nroff** command. Relevant values are part of the current environment. The mode still must be in effect, or again so, at the time of physical output.

Initial Value: Off

If No Value Specified: −

.fp *Font Number*[*File*]

Specifies the font position. This is a statement that the specified font is mounted on the position specified by the *Number* variable. The *Font* variable must be a one− or two−character ASCII font name.

> **Attention:** It is an irrecoverable error if the *Font* variable is not specified.

The **.fp** request accepts a third optional variable, the *File* variable, which is the actual path name of the file containing the specified font. The *File* variable value can be any legal file name and can contain extended characters.

Japanese Language Support**:** The *File*value can be any legal file name. Values are typesetter− or printer−dependent.

Initial Value: −

If No Value Specified: Ignored

.ft *Font*
Changes the font style to the specified font, or if *Font* value is numeric, to the font mounted on that position. Alternatively, embed **\f***Font* command. The font name **P** is reserved to mean the previous font. The *Font* variable value must be an ASCII font name or font position.

If using a font name consisting of two characters, use the alternative form of **.ft**, **\f**. Relevant values are part of the current environment. Values are typesetter or printer−dependent.

Initial Value: Roman

If No Value Specified: Previous

.ps [+/−][*Number*]
Sets the point size to that specified by the +/−*Number* variable. Although any positive size value can be requested, an invalid size results in the nearest valid size being used. Size 0 refers to the previous size. Alternatively, **\s***Number* or **\s**+/−*Number*; if the *Number* value is two digits, use **\s(***Number* or **\s**+/−**(***Number*. For compatibility with older versions of the **troff** command, the form is valid for two−digit values of *n* = **10**, **11**, **12**, **14**, **16**, **18**, **20**, **22**, **24**, **28**, and **36**.

This request is ignored by the **nroff** command. Relevant values are part of the current environment.

Initial Value: 10 point

If No Value Specified: Previous

.ss *Number*
Sets space−character size to the specified number divided by 36 ems. This size is the minimum word spacing in adjusted text. This request is ignored by the **nroff** command. Relevant values are part of the current environment.

Initial Value: 12/36 em

If No Value Specified: Ignored

Page Control

.bp [+/−][*Number*]
Specifies a break page. The current page is ejected and a new page is begun. If the +/−*Number* variable is specified, its value becomes the new page number. Also refer to the **.ns** request.

This request normally causes a line break similar to the **.br** request. Calling this request with the control character " ' " (instead of "**.**") suppresses that break function.

Initial Value: *Number*=**1**

If No Value Specified: −

.mk *Register*
Marks the current vertical place (or a place in the current diversion) in an internal register (associated with the current diversion level) or in the specified register, if given. The *Register* variable is the ASCII name of a number register. Mode or relevant values are associated with the current diversion level. For more information, refer to the **.rt** request.

Initial Value: None

If No Value Specified: Internal

.ne *Number D* Indicates a need for the specified vertical space. If the page space needed (*Number*) is greater than the distance to the next trap (*D*), a forward vertical space of size *D* occurs, which springs the trap. If there are no remaining traps on the page, the size specified by the *D* variable is the distance to the bottom of the page. If the distance to the next trap (*D*) is less than one vertical line space (**v**), another line could still be output before the trap is sprung. In a diversion, the size specified by *D* is the distance to the diversion trap, if any, or is very large.

The value of *D* is also normally contained in the **.t** *Number* register. Mode or relevant values are associated with the current diversion level.

Initial Value: *Number*=**1***V*

If No Value Specified: −

.pl [+/−][*Number*] Sets page length to the +/−*Number* variable value. The internal limitation is approximately 136 inches in the **nroff** command, but varies with the device type in the **troff** command. A good working maximum for the **troff** command is 75 inches. The current page length is available in the **.p** register.

Initial Value: 11 inches

If No Value Specified: 11 inches

.pn [+/−][*Number*] Specifies that the next page (when it occurs) has the page number specified by the +/−*Number* variable. A **.pn** request must occur either before text is initially printed or before a break occurs to affect the page number of the first page. The current page number is in the **%** register.

Initial Value: *Number*=**1**

If No Value Specified: Ignored

.po [+/−][*Number*] Specifies a page offset. The current left margin is set to the +/−*Number*variable value. The initial **troff** command value provides 1 inch of left margin. For more information, refer to "Line Length and Indenting". The current page offset is available in the **.o** register.

Initial Value: **0** for the **nroff** command; **1** for the **troff** command.

If No Value Specified: Previous

.rt [+/−][*Number*] Returns upward only to a marked vertical place in the current diversion. If the +/−*Number* variable value (relative to the current place) is given, the place is the value specified by the +/−*Number* variable from the top of the page or diversion. If the *Number* variable is not specified, the place is marked by a previous **.mk** request. Mode or relevant values are associated with the current diversion level.

The **.sp** request can be used in all cases, instead of the **.rt** request, by spacing to the absolute place stored in an explicit register as, for example, when using the sequence **.mk***Register* . . . **.sp**|**n**R**u**.

Initial Value: None

If No Value Specified: Internal

Text Filling, Adjusting, and Centering

.ad *Indicator* Begins line adjustment. If the fill mode is not on, adjustment is deferred until the fill mode is back on. If the *Indicator* variable is present, the adjustment type is changed as shown in the following list:

Indicator	Adjustment Type
l	Adjust left margin only.
r	Adjust right margin only.
c	Center.
b or *n*	Adjust both margins.
blank	Unchanged.

The adjustment indicator can also be a number obtained from the **.j** register.

Japanese Language Support:

Indicator	Adjustment Type
k	Turn on kinsoku shori processing (turned off with **.ad n**, **.ad b**, or **.ad l**).

Normally, lines of Japanese text are filled to the margins without regard for the characters beginning or ending lines. When kinsoku shori processing is enabled, lines are prevented from ending with an open bracket character or from beginning with a close bracket or punctuation character. If a line ends with an open bracket, the line is left short and the bracket begins the next line. If a line begins with a close bracket or punctuation character, the preceding line is extended and the character ends the preceding line. Requesting Japanese kinsoku shori processing on an output device that does not support kanji characters has no effect.

Relevant values are part of the current environment.

Initial Value: Adjust, both

If No Value Specified: Adjust

.br Specifies a break. The filling of the line currently being collected is stopped and the line is output without adjustment. Text lines beginning with space characters and empty text lines (blank lines) also cause a break.

Initial Value: −

If No Value Specified: −

.ce [*Number*] Centers the next specified number of input text lines within the current line length, minus indent. If the *Number* variable equals 0, any residual count is cleared. A break occurs after each of the *Number* variable input lines. If the input line is too long, it is left adjusted. Relevant values are part of the current environment. This request normally causes a line break similar to the **.br** request. Calling this request with the control character " ' " (instead of "**.**") suppresses that break function.

Initial Value: Off

If No Value Specified: *Number*=**1**

.fi Fills subsequent output lines. The **.u** register has a value of 1 (one) in fill mode and a value of 0 (zero) in no−fill mode. Relevant values are part of the current environment. This request normally causes a line break similar to the **.br** request. Calling this request with the control

character " ' " (instead of "**.**") suppresses that break function.

Initial Value: Fill

If No Value Specified: −

.na Specifies no−adjust mode. Adjustment is turned off; the right margin is ragged. The adjustment type for the **.ad** request is not changed. Output−line filling still occurs if the fill mode is on. Relevant values are part of the current environment.

Initial Value: None

If No Value Specified: −

.nf Specifies no−fill mode. Subsequent output lines are neither filled nor adjusted. Input−text lines are copied directly to output lines without regard for the current line length. Relevant values are part of the current environment. This request normally causes a line break similar to the **.br** request. Calling this request with the control character " ' " (instead of "**.**") suppresses that break function.

Initial Value: Fill

If No Value Specified: −

Vertical Spacing

Blank text line Causes a break and outputs a blank line exactly like an **.sp1** request.

.ls *Number* Sets line spacing to the value specified by the +/−*Number* variable. The *Number*−**1***V***s** (blank lines) variable values are appended to each output−text line. Appended blank lines are omitted if the text or previous appended blank line reached a trap position. Relevant values are part of the current environment.

Initial Value: **1**

If No Value Specified: Previous

.ns Turns on no−space mode. When on, the no−space mode inhibits **.sp** and **.bp** requests without a next page number. The no−space mode is turned off when a line of output occurs or with the **.rs** request. This request normally causes a break.

Initial Value: Space

If No Value Specified: −

.os Outputs saved vertical space. The no−space mode has no effect. Used to output a block of vertical space requested by the previous **.sv** request.

Initial Value: −

If No Value Specified: −

.rs Restores spacing. The no−space mode is turned off. This request normally causes a break.

Initial Value: None

If No Value Specified: −

.sp *Number* Spaces vertically in either direction. If the *Number* variable value is negative, the motion is backward (upward) and is limited to the distance to the top of the page. Forward (downward)

motion is truncated to the distance to the nearest trap. If the no−space mode is on, no spacing occurs. Refer to the **.ns** and **.rs** requests. This request normally causes a line break similar to the **.br** request. Calling this request with the control character "'" (instead of ".") suppresses that break function.

Initial Value: −

If No Value Specified: **1***V*

.sv *Number* Saves a contiguous vertical block of the specified size. If the distance to the next trap is greater than the *Number* variable value, the specified vertical space is output. The no−space mode has no effect. If this distance is less than the specified vertical space, no vertical space is immediately output, but is remembered for later output (refer to the **.os** request). Subsequent **.sv** requests overwrite any still−remembered *Number* variable value.

Initial Value: −

If No Value Specified: *Number*=**1***V*

.vs *Number* Sets vertical base−line spacing size *V* to the *Number* variable. Transient extra vertical space can be specified by **\x** *N*. Relevant values are part of the current environment.

Initial Value: The *Number* variable equals 1/16 inch for the **nroff** command and 12 points for the **troff** command.

If No Value Specified: Previous

Line Length and Indenting

.in [+/−]*Number* Sets indent to the +/−*Number* variable value. The indent is prepended to each output line. Relevant values are part of the current environment. This request normally causes a line break similar to the **.br** request. Calling this request with the control character " ' " (instead of "**.**") suppresses that break function.

Initial Value: *Number*=**0**

If No Value Specified: Previous

.ll [+/−]*Number* Sets line length to the +/−*Number* variable value. In the **troff** command, the maximum line length plus page offset is device−dependent. Relevant values are part of the current environment.

Initial Value: 6.5 inches

If No Value Specified: Previous

.ti [+/−]*Number* Specifies a temporary indent. The next output text line is indented a distance of the value specified by the +/−*Number* variable with respect to the current indent. A negative value for the *Number* variable can result in spacing backward over the current indent, so that the resulting total indent can be a value of 0 (zero) (equal to current page offset), but cannot be less than the current page offset. The temporary indent applies only for the one output line following the request; the value of the current indent, which is stored in the **.i** register, is not changed.

Relevant values are part of the current environment. This request normally causes a line break similar to the **.br** request. Calling this request with the control character " ' " (instead of "**.**") suppresses that break function.

Initial Value: −

If No Value Specified: Ignored

Macros, Strings, Diversions, and Position Traps

.am *Macro1* [*Macro2*]
Appends to *Macro 1*; appends version of the **.de** request. Both the *Macro1* and *Macro2* variables must be either one or two ASCII characters. *Macro2* is a termination sequence to end the diversion.

Initial Value: −

If No Value Specified: **.***Macro2***=..**

.as *StringName String*
Appends the specified string to the value specified by the *StringName* variable; appended version of the **.ds** request. The *StringName* variable value must be one or two ASCII characters.

Initial Value: −

If No Value Specified: Ignored

.ch *Macro* [*Number*]
Changes the trap position for the specified macro to the value specified by the *Number* variable. In the absence of the *Number* variable, the trap, if any, is removed. The *Macro* variable value must be one or two ASCII characters.

Initial Value: −

If No Value Specified: −

.da [*Macro*]
Diverts, appending to the specified macro and appends version of the **.di** request. The *Macro* variable must be one or two ASCII characters. Mode or relevant values are associated with the current diversion level.

Initial Value: −

If No Value Specified: End current diversion

.de *Macro1* [*Macro2*]
Defines or redefines the value specified by the *Macro1* variable. The contents of the macro begins on the next input line. Input lines are copied in copy mode until the definition is stopped by a line beginning with .*Macro2*. In the absence of the *Macro2* variable, the definition is stopped by a line beginning with "**..**". A macro can contain **.de** requests, provided the stopping macros differ or the contained definition terminator is concealed. The "**..**" can be concealed as "\\ .", which copies as "**\...**" and is reread as "**..**". The *Macro1* and *Macro2*variables must each be one or two ASCII characters.

Initial Value: −

If No Value Specified: **.***Macro2***=..**

.di [*Macro*]
Diverts output to the specified macro. Normal text processing occurs during diversion except that page offsetting is not performed. The diversion ends when the **.di** or **.da** request is

encountered without a variable. Extraneous requests of this type should not be displayed when nested diversions are being used. The *Macro* variable must be one or two ASCII characters. Mode or relevant values are associated with the current diversion level.

Initial Value: −

If No Value Specified: End

.ds *StringName String*

Defines a string specified by the *StringName* variable to contain the value specified by the *String* variable. Any initial double−quote in *String* is stripped off to permit initial blanks. The *StringName* variable must be one or two ASCII characters.

.ds *StringName* ^A<*SetNumber*> <*MessageNumber*> [^A"<*DefaultMessage*>"] [^A<*Argument*>^B<*Argument*>^B<*Argument*>...]

Provides an alternate **.ds** syntax that allows the use of a message catalog for language−independent string definitions.

Based on the message *SetNumber* and the *MessageNumber* within the locale−specific catalog, the message catalog is read in copy mode and the corresponding message is placed into the *StringName* variable. The initial sequence specifying the message set and message number can be omitted for backward compatibility. The ASCII code Control−A (**^A**) delimits message identification, default message and optional argument list. The ASCII code Control−B (**^B**) delimits an individual optional argument list.

In the following example,

```
.ds {c ^A2 41^A"ERROR: (%1$s) input line \
%2$s" ^A\n(.F^B\n(.c
```

`2` is the message set number.

`41` is the message number.

text within quotes `("...")` is the default message.

`\n(.F` is the name of the current input file.

`\n(.c` is the number of lines read from the input file.

If you assume the **troff** command runs with these conditions:

- The message at set 2 and number 41 matches the default message
- The current input file is `paper.doc`
- The **.ds** directive is on line 124 in the input file.

then the string `{c` would be defined as:

```
ERROR: (paper.doc)input line 123
```

Other examples are:

```
.ds {c ^A2 41
/* Without optional default message */

.ds {c ^A2 41^A"ERROR: (%1$s) input file \
%2$s" /* Without optional arguments */
```

If both the set number and the message number are set to zero, then the current date is returned in the current local's format. A user defined date format string can be defined in the default message field. The user defined format string must conform to the conversion specifications outlined by the **strftime** function in *Technical Reference: Base Operating System and Extensions*.

In the following examples:

```
.ds DT^A0 0
```

If the current date were July 10, 1991, in an English U.S. locale, `DT` would be defined as `7/10/91`.

```
.ds DT^A0 0^A"Today is %B %d, %Y"
```

If the current date were July 10, 1991, in an English U.S. locale, `DT` would be defined as `Today is July 10, 1991`.

The second syntax method is not intended for general use. It is used in the **nroff** and **troff** macro files supplied with the system to facilitate internationalization of internally generated messages.

Initial Value: −

If No Value Specified: Ignored

.dt *Number Macro*

Installs a diversion trap at the position specified by the *Number* variable in the current diversion to start the specified macro. Another **.dt** request redefines the diversion trap. If no variables are given, the diversion trap is removed. The *Macro* variable must be one or two ASCII characters. Mode or relevant values are associated with the current diversion level.

Initial Value: −

If No Value Specified: Off

.em *Macro*

Calls the specified macro when all input has ended. The effect is the same as if the contents of the specified macro had been at the end of the last file processed. The specified macro must be one or two ASCII characters.

Initial Value: None

If No Value Specified: None

.it *Number Macro*

Sets an input−line−count trap to call the specified macro after the number of lines of text input specified by the *Number* variable have been read (control or request lines are not counted). The text can be inline text or text provided by macros called explicitly (through inline calls) or implicitly (through traps). The *Macro* variable must be one or two ASCII characters. Relevant values are part of the current environment.

Initial Value: −

If No Value Specified: Off

.rm *Name*

Removes the specified request, macro, or string. The *Name* variable value is removed from the name list and any related storage space is freed. Subsequent references have no effect. The *Name* variable must be one or two ASCII characters.

Initial Value: −

If No Value Specified: Ignored

.rn *Name1Name2*

Renames the request, macro, or string value specified by the *Name1*variable to the value specified by the *Name2* variable. The *Name1*and *Name2* variable values must each be one or two ASCII characters.

Initial Value: Ignored

If No Value Specified: −

.wh *Number Macro*

Installs a trap to call the specified macro at the page position specified by the *Number* variable. A negative *Number* variable value is interpreted with respect to the page bottom. Any macro previously planted at the page position specified by the *Number* variable is replaced by the *Macro* variable value. A *Number* variable value of 0 refers to the top of a page. In the absence of the *Macro* variable, the first trap found at the page position specified by the *Number* variable, if any, is removed. The *Macro* variable must be one or two ASCII characters.

Initial Value: −

If No Value Specified: −

Number Registers

.af *RegisterIndicator*

Assigns the format as specified by the *Indicator* variable to the specified register. The *Register* variable must be one or two ASCII characters. The available format *Indicator* variable values are as follows:

Indicator	Numbering Sequence
1	0,1,2,3,4,5, . . .
001	000,001,002,003,004,005, . . .

i	0,i,ii,iii,iv,v, . . .
I	0,I,II,III,IV,V, . . .
a	0,a,b,c, . . . ,z,aa,ab, . . . ,zz,aaa, . . .
A	0,A,B,C, . . . ,Z,AA,AB, . . . ,ZZ,AAA, . . .

An Arabic format indicator having *N* digits (for example, 000000001) indicates a field width of *N* digits. The read−only registers and the width function are always Arabic.

Japanese Language Support**:** The following value specifies the character width for formatting Japanese numeric output in kanji:

k The number is formatted as a kanji string. If this is requested when a non−kanji codeset is specified, a warning message is printed and the **1** format is used.

Initial Value: Arabic

If No Value Specified: −

.nr *Register +/−Number1 Number2*

Assigns the specified register the value specified by the *+/−Number* variable with respect to the previous value, if any. The increment for auto−incrementing is set to the *Number2* variable value. The *Register* variable must be one or two ASCII characters.

Initial Value: −

If No Value Specified: −

.rr *Register*

Removes the specified register. If many registers are being created dynamically, it can become necessary to remove registers that are not needed to recapture internal storage space for new registers. The *Register* variable must be one or two ASCII characters.

Initial Value: −

If No Value Specified: −

Tabs, Leaders, and Fields

.fc *Delimiter Indicator*

Sets the field delimiter to the specified delimiter; the padding indicator is set to the space character or to the specified indicator. In the absence of variables, the field mechanism is turned off. The *Delimiter* variable value and the *Indicator* variable value must be ASCII characters.

Initial Value: Off

If No Value Specified: Off

.lc *Character*

Sets the leader repetition character to the specified character, or removes specifying motion. The *Character* variable value must be an ASCII character. Relevant values are part of the current environment.

Initial Value: **.**

If No Value Specified: None

.ta *Stop* [*Type*]... Sets tab stops. Default tab stops are set at every eight characters for the **nroff** command and every half inch for the **troff** command. Multiple *StopType* pairs can be specified by separating them with spaces; a value preceded by **+** (plus sign) is treated as an increment to the previous stop value.

The specified type determines how the text is adjusted at the tab stops. The *Type* variable values are as follows:

Type	Adjustment
R	Right−adjusting
C	Centering
blank	Left−adjusting

Relevant values are part of the current environment.

Initial Value: 8 ens for the **nroff** command and 0.5 inch for the **troff** command

If No Value Specified: None

.tc *Character* Sets the tab repetition character to the specified character, or removes specifying motion. The *Character* variable value must be an ASCII character. Relevant values are part of the current environment.

Initial Value: None

If No Value Specified: None

Input/Output Conventions and Character Translations

.cc *Character* Sets the basic control character to the specified character, or resets to "**.**". The *Character* variable value must be an ASCII character. Relevant values are part of the current environment.

Initial Value: **.**

If No Value Specified: **.**

.cu [*Number*] A variant of the **.ul** request that causes every character to be underlined and causes no line breaks to occur in the affected input lines. That is, each output space following a **.cu** request is similar to an unpaddable space. The **.cu** request is identical to the **.ul**request in the **troff** command. Relevant values are part of the current environment.

Initial Value: Off

If No Value Specified: *Number*=**1**

.c2 *Character* Sets the no−break control character to the specified character or resets to " ' ". The *Character* variable value must be an ASCII character. Relevant values are part of the current environment.

Initial Value: '

If No Value Specified: '

.ec *Character*
Sets the escape character to \ (backslash) or to the value specified by the *Character* variable, if given. The *Character* variable value must be an ASCII character.

Initial Value: \

If No Value Specified: \

.eo
Turns off the escape mechanism.

Initial Value: On

If No Value Specified: −

.lg [*Number*]
Turns on the ligature mode if the *Number* variable value is absent or nonzero; turns off ligature mode if the *Number* variable value is 0. If the *Number* variable value is 2, only the two−character ligatures are automatically called. The ligature mode is inhibited for request, macro, string, register, or file names, and in the copy mode. This request has no effect in the **nroff** command.

Initial Value: On, for the **troff** command

If No Value Specified: On

.tr *Character1Character2Character3Character4*
Translates, among other things, the character value specified by the *Character1* variable into the *Character2* variable value, the character value specified by the *Character3* variable into the *Character4*variable value. If an odd number of characters is given, the last one is mapped into the space character. To be consistent, a particular translation must stay in effect from input to output time. All specified characters must be ASCII characters. To reset the **.tr** request, follow the request with previous variables given in duplicate.

For example, the following **.tr** request:

```
.tr aAbBc<C,>
```

can be reset by entering:

```
.tr aabbcc
```

It must stay in effect until logical output.

Initial Value: None

If No Value Specified: −

.ul [*Number*]
Underlines in the **nroff** command (or italicizes in the **troff**command) the number of input−text lines specified by the *Number*variable. Actually switches to underline font, saving the current font for later restoration. Other font changes within the span of a **.ul** request take effect, but the restoration undoes the last change. Output generated by the **.tl** request is affected by the font change, but does not decrement the *Number* variable value. For more information, refer to the section "Three−Part Titles". If the specified number is greater than 1, there is the risk that a trap−called macro can provide text lines within the span;

environment switching can prevent this.

Relevant values are part of the current environment.

Initial Value: Off

If No Value Specified: *Number*=**1**

.uf *Font* Underlines the font set to the value specified by the *Font* variable. In the **nroff** command, the *Font* variable cannot be on position 1 (initially Times Roman). The *Font* variable value must be an ASCII font name.

Initial Value: Italic

If No Value Specified: Italic

Hyphenation

.hc *Character* Sets the hyphenation indicator character to the value specified by the *Character* variable or to the default. The indicator is not displayed in the output. The *Character* variable value must be an ASCII character. Relevant values are part of the current environment.

Initial Value: **\%**

If No Value Specified: **\%**

.hw *Word1*... Specifies hyphenation points in words with embedded minus signs. Versions of a word with a terminal s are implied; that is, *dig−it* implies *dig−its*. This list is examined initially and after each suffix stripping. The space available is 1024 characters, or about 50 to 100 words.

Initial Value:

If No Value Specified: Ignored

.hy *Number* Turns on automatic hyphenation if the specified number is equal to or greater than 1; turns it off if the specified number is equal to 0 (equal to the **.nh** request). If the specified number is 2, the last lines (ones that cause a trap) are not hyphenated. If the specified number is 4 or 8, the last or first two characters, respectively, of a word are not split off. These values are additive; for example, a value of 14 calls all three restrictions (number equal to 2, number equal to 4, and number equal to 8).

Relevant values are part of the current environment.

Initial Value: No hyphenation

If No Value Specified: Hyphenate

.nh Turns off automatic hyphenation. Relevant values are part of the current environment.

Initial Value: No hyphenation

If No Value Specified: −

Three−Part Titles

.lt [+/−][*Number*] Sets the length of title value specified by the +/−*Number* variable. The line length and the title length are independent. Indents do not apply to titles, although page offsets do. Relevant values are part of the current environment.

Initial Value: 6.5 inches

If No Value Specified: Previous

.pc *Character* Sets the page number character to the specified character or removes it. The page−number register remains %. The *Character* variable value must be an ASCII character.

Initial Value: **%**

If No Value Specified: Off

.tl *'Left'Center'Right'* The strings represented by the *Left*, *Center*, and *Right*variables, respectively, are left−adjusted, centered, and right−adjusted in the current title length. Any of the strings can be empty, and overlapping is permitted. If the page−number character (initially **%**) is found within any of the fields, it is replaced by the current page number having the format assigned to the **%** register. Any ASCII character that is not displayed in the strings can be used as the string delimiter.

Initial Value: −

If No Value Specified: −

Output−Line Numbering

.nm [+/−][*Number*] [*M*] [*S*] [*I*]

Turns on line−number mode. If the *M* variable is specified, only those line numbers that are multiples of the *M* variable value are to be printed. Every line number is printed if the *M* variable is absent (default is *M*=**1**). When line−number mode is in effect, a three−digit Arabic number plus a digit space are prepended to output text lines. The text lines are thus offset by four digit spaces, but otherwise retain their line length. If the *S* variable is given, it specifies the number of digit spaces to be displayed between the line number and the text (default is *S*=**1**). If the *I* variable is given, it specifies the number of digit spaces to indent before the line number (default is *I*=**0**).

Relevant values are part of the current environment.

Initial Value: −

If No Value Specified: Off

.nn *Number* Suspends line numbering. The specified number of lines are not numbered. Relevant values are part of the current environment.

Initial Value: −

If No Value Specified: *Number*=**1**

Conditional Acceptance of Input

The *Condition* variable specifies one of the following one−character names:

o	If the current page number is odd.
e	If the current page number is even.
t	If the formatter is the **troff** command.

n	If the formatter is the **nroff** command.
.if *Condition Anything*	If the value specified by the *Condition* variable is true, accepts the value specified by the *Anything* variable as input; in multiline case, uses **{***Anything***}**.
.if *!Condition Anything*	If the value specified by the *Condition* variable is false, accepts the value specified by the *Anything* variable as input.
.if *Number Anything*	If the expression states that the *Number* variable value is greater than 0, accept the value specified by the *Anything* variable as input.
.if *!Number Anything*	If the expression states that the *Number* variable value is less than or equal to 0, accepts the value specified by the *Anything* variable as input.
.if *'String1'String2' Anything*	If the *String1* variable value is identical to the *String2* variable value, accepts the value specified by the *Anything* variable as input. Any nonblank ASCII character not in the *String1* and *String2* variables can be used as the delimiter.
.if *!'String1'String2' Anything*	If the *String1* variable value is not identical to the *String2*variable value, accepts the value specified by the *Anything* variable as input. Any nonblank ASCII character not in the *String1* and *String2* variables can be used as the delimiter.
.el *Anything*	Specifies the **else** portion of an **if/else** conditional.
.ie *Condition Anything*	Specifies the **if** portion of an **if/else** conditional dependent on the value of the *Condition* variable. Can be used with any of the preceding forms of the **.if** request.

Environment Switching

.ev *Environment* Switches to the specified environment. The value specified by the *Environment*variable must be 0, 1, or 2. Switching is done in push−down fashion so that restoring a previous environment must be performed with the **.ev**request rather than with a specific reference.

Initial Value: *Environment*=0

If No Value Specified: Previous

Insertions from Standard Input

.ex Exits from the **nroff** command or **troff** command. Text processing is stopped exactly as if all input had ended.

Initial Value: −

If No Value Specified: −

.rd *Prompt* Reads insertion from standard input until two newline characters in a row are found. If the standard input is the user's keyboard, the specified prompt (or the ASCII BEL character) is written onto the user's terminal. The **.rd** request behaves like a macro, and additional variables can be placed after the *Prompt* variable.

Initial Value: −

If No Value Specified: *Prompt*=the ASCII BEL character

Input and Output File Switching

.cf *File* Copies the contents of the specified file, uninterrupted, into the **troff** command output file at this point. Problems occur unless the motions in the file restore the current horizontal and

vertical position.

Initial Value: −

If No Value Specified: −

.lf *Number File* Corrects the **troff** command interpretation of the current line number (as specified by the *Number* variable) and the current file (as specified by the *File* variable) for use in error messages.

Initial Value: −

If No Value Specified: −

.nx *File* Uses the specified file as the input file. The current file is considered ended and the input is immediately switched to the specified file.

Initial Value: −

If No Value Specified: End of file

.pi *Program* Pipes output to the specified program. This request must occur before any printing occurs. No variables are transmitted to the specified program.

Initial Value: −

If No Value Specified: −

.so *File* Switches the source file. The top input (file−reading) level is switched to the specified file. When this file ends, input is again taken from the original file. The **.so** request can be nested.

Once a **.so** request is encountered, the processing of the specified file is immediate. Processing of the original file (for example, a macro that is still active) is suspended.

A file should be preprocessed, if necessary, before being called by the **.so** request. The **eqn**, **tbl**, **pic**, and **grap** commands do not reach through a **.so** request to process an object file.

Initial Value: −

If No Value Specified: −

Miscellaneous

.ab *Text* Prints the value specified by the *Text* variable to the diagnostic output (normally the terminal) and ends without further processing. If text is missing, the message `User Abort` is printed and the output buffer is flushed. This request is used in interactive debugging to force output.

.ab ^A<*SetNumber*> <*MessageNumber*> [^A"<*Default*>"] [^A<*Argument*>^B<*Argument*> ^B<*Argument*>...]

Provides alternate syntax to allow use of a message catalog for language−independent abort messages. Prints the appropriate message specified by the parameter on the diagnostic output (normally the terminal) and ends without further processing. If there are no parameters, the message catalog equivalent to the following:

```
troff: User Abort, line no. file filename
```

is output. The output buffer is flushed. This request is used in interactive debugging to force output.

Based on the message *SetNumber* and the *MessageNumber* variables within the locale−specific catalog, the message catalog is read in copy mode and the corresponding message is written to the user's terminal. The initial sequence specifying the message set and message number can be omitted for backward compatibility. The ASCII code Control−A (^A) delimits message identification, default message, and optional argument list. The ASCII code Control−B (^B) delimits individual optional argument list.

In the following example:

```
.ab ^A2 42^A"Processing has been terminated \
 at line %1$s."^A\n(c.
```

2 is the message set number.

42 is the message number.

Text within quotes "..." is the default message.

\n(c. is the number of lines read from the input file.

If you assume the **troff** command runs with the following conditions:

- The message at set 2 and number 42matches the default message.
- The **.ab** directive is on line 124in the input file.

then the following would be displayed on the user's terminal:

```
Processing has been terminated at line 123.
```

Initial Value: −

If No Value Specified: User cancel

.Dt*Parameter*

Defines the format for returning the date within the **nroff** or **troff** request. By default, without the optional *Parameter*, the locale−specific date format specified by the current locale setting for the **LC_TIME** category is used. This corresponds to the "%x" format specifier of **strftime**. *Parameter* is a format string identical to the format string used with the **strftime** function in *Technical Reference: Base Operating System and Extensions*. Reference this function for a complete list of the format specifiers.

For example,

```
.Dt "%A, %B %d, %Y (%T)"
```

provides the following output for an English−speaking locale:

```
Thursday, January 31, 1991 (10:40:00)
```

The %A format is replaced by the locale−specific weekday name. The %B format is replaced by the locale−specific month name. The %d format is replaced by the day of the month in a two−digit format. The %Y format is replaced by the year with the century as a decimal number. The %T format is replaced by the time in hours (24−hour clock), minutes, and seconds in decimal numbers. This format provides for leap seconds and double leap seconds.

.fl Flushes output buffer. This request normally causes a line break similar to the **.br** request. Calling this request with the control character " ' " (instead of "**.**") suppresses that break function.

Initial Value: −

If No Value Specified: −

.ig *Macro* Ignores input lines. The **.ig** request works exactly like the **.de** request, except that the input is discarded. For more information, refer to "Macros, Strings, Diversions, and Position Traps". The input is read in copy mode, and any auto−incremented registers are affected. The *Macro* variable must be one or two ASCII characters.

Initial Value: −

If No Value Specified: **.***Macro*=**..**

.mc [*Character*] [*N*] Uses the specified character as the margin character to display the specified distance (*N*) to the right of the margin after each non−empty text line (except those produced by the **.tl** request). If the output line is too long (as can happen in no−fill mode), the character is appended to the line. If the *N* variable is not given, the previous *N* variable is used. The first *N* variable is 0.2 inches in the **nroff** command and 1 em in the **troff** command.

Relevant values are part of the current environment.

Initial Value: .2 inches in **nroff**; 1 em in **troff**

If No Value Specified: Off

.pm [*Character*] Prints macros. The names and sizes of all of the defined macros and strings are printed on the user's terminal. If any ASCII alphanumeric character is given as a variable, only the total of the sizes is printed. The size is given in blocks of 128 characters.

Initial Value: −

If No Value Specified: All

.sy *Command* [*Flags*] The specified command is run but its output is not captured at

this point. The standard input for the specified command is closed. Output must be explicitly saved in an output file for later processing. Often the **.sy** directive is followed by a subsequent **.so** directive to include the results of the previous command.

For example:

```
.sy date > /tmp/today
Today is
.so /tmp/today
```

Initial Value: −

If No Value Specified: −

.tm *String* The specified string is written to the user's terminal.

.tm ^A<*SetNumber*> <*MessageNumber*> [^A"<*DefaultMessage*>"] [^A<*Argument*> ^B <*Argument*>^B<*Argument*>...]

Based on the message set number and the message number within the locale−specific catalog, the message catalog is read in copy mode and the corresponding message is written to the user's terminal. The initial sequence specifying the message set and message number can be omitted for backward compatibility. The ASCII code Control−A ^A delimits message identification, default message, and optional argument list. The ASCII code Control−B ^B delimits individual optional argument list.

In the following example:

```
.tm ^A2 23^A"The typesetter is %1$s.On line
%2$s."^A\*(.T^B\n(c.
```

2 is the message set number.

23 is the message number.

Text within quotes "..." is the default message.

*(.T is the first argument in troff for value of **−T**.

\n(c. is the number of lines read from the input file.

If you assume the **troff** command runs with the following conditions:

- The message at set 2 and number 23 matches the default message.
- The command line has **troff** using the **−T** option with device PSC.
- The **.tm** directive is on line 539 in the input file.

Then the following would be displayed on the user's terminal:

```
The typesetter is psc. On line 538.
```

The locale−specific message catalog is found in **/usr/lib/nls/msg/$LANG/macros.cat**.

Initial Value: −

If No Value Specified: Newline

Notes

The following notes apply to the **nroff** and **troff** requests. They are referenced by number in the requests where they apply.

1. The **.L** string register contains the current program locale value of all the categories.
2. The **.m** string register contains the locale value of the **LC_MESSAGES** category.
3. The **.t** string register contains the locale value for the **LC_TIME** category.
4. While the **.L**, **.t**, and **.m**string registers provide access to some environment values, a more general technique can be used to access any other environment variable. For example, if the **TED** environment variable is exported, the following **troff** commands:

```
.sy echo .ds z $TED >x
.so x
.sy rm x
```

set the *z* string register to contain the value of **$TED**.

Environment Variables

LC_ALL

Specifies the locale to be used for all the locale categories. It overrides any setting of the other locale environment variables.

LC_MESSAGES

Specifies the locale value for the **LC_MESSAGES** category. This is used if the **LC_ALL** environment variable is not set.

LC_TIME

Specifies the locale value for the **LC_TIME** category. This is used if the **LC_ALL** environment variable is not set.

LANG Specifies the locale value to be used for all the locale categories. This is used if none of the above environment variables are set. This is the most often used environment variable to specify the locale.

Files

/usr/share/lib/tmac/tmac.*

Contains the pointers to standard macro files.

/usr/share/lib/macros/*

Denotes standard macro files.

/usr/share/lib/tmac/tmac.an

Contains the pointer to the **man** macro package.

/usr/share/lib/macros/an

Contains the **man** macro package.

/usr/share/lib/tmac/tmac.e file

	Contains the **me** macro definition file.
/usr/share/lib/me directory	
	Contains the macro definition files.
/usr/share/lib/tmac/tmac.m	
	Contains the pointer to the **mm** macro package.
/usr/share/lib/macros/mmn	
	Contains the **mm** macro package.
/usr/share/lib/macros/mmt	
	Contains the **mm** macro package.
/usr/share/lib/tmac/tmac.ptx	
	Points to the macro package.
/usr/share/lib/macros/ptx	
	Contains the macro package.
/usr/share/lib/tmac/tmac.x	
	Contains the macro definition files.
/usr/share/lib/ms	Contains the **ms** macro definitions.
/usr/share/lib/tmac/tmac.v	
	Contains macro definitions.
/usr/share/lib/macros/vmca	
	Contains macro definitions.
/usr/lib/nls/msg/$LANG/macros.cat	
	Contains locale−specific message catalog for the **mm**, **me**, **ms**, and **mv** macro packages.
/usr/lib/font/dev*/*	
	Contains the font width tables.
/var/tmp/trtmp*	Denotes a temporary file.

Related Information

The **col** command, **eqn** command, **grap** command, **hplj** command, **ibm3812** command, **ibm3816** command, **mm** command, **mmt** command, **mvt** command, **neqn** command, **nroff** command, **pic** command, **ptx** command, **refer** command, **tbl** command, **tc** command, **xpreview** command.

The **nroff** and **troff** Input file format, **troff** file format, **troff** font file format.

The **setlocale** function, **strftime** function.

Message Facility Overview for Programming in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

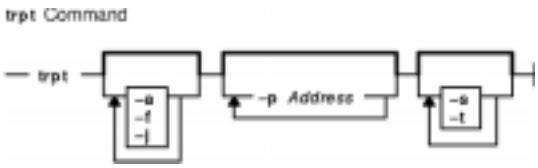
National Language Support Overview for Programming in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

trpt Command

Purpose

Performs protocol tracing on TCP sockets.

Syntax



trpt [**−a**] [**−f**] [**−j**] [**−p***Address*]... [**−s**] [**−t**]

Description

The **trpt** command queries the buffer for Transmission Control Protocol (TCP) trace records. This buffer is created when a socket is marked for debugging with the **setsockopt** subroutine. The **trpt** command then prints a description of these trace records.

> **Note:** You can use the **traceson** command to turn on socket level debugging for daemons.

When you specify no options, the **trpt** command prints all the trace records found in the system and groups them according to their TCP/IP connection protocol control block (PCB).

Before you can use the **trpt** command, you must:

1. Isolate the problem and mark for debugging the socket or sockets involved in the connection.
2. Find the address of the protocol control blocks associated with these sockets by using the **netstat −aA** command.
3. Then you can run the **trpt** command, using the **−p** flag to supply the associated protocol control block addresses. You can specify multiple **−p***Address* flags with a single **trpt** command.

The **−f** flag can be used to follow the trace log once it is located. The **−j** flag can be used to check the presence of trace records for the socket in question.

If the system image does not contain the proper symbols to find the trace buffer, the **trpt** command cannot succeed.

Output Fields

The information put out by the **trpt** command varies with the flag you use. Definitions of the fields contained in the various types of output follow:

Protocol Control Block identifier	Identifies the protocol block to be traced, as shown in the following example: `4c500c:`
Timestamp	Specifies the time at which the connection is attempted, as shown in the following example: `500`

Connection State Specifies the state of the connection with the protocol control block:

CLOSED	Connection is closed.
LISTEN	Listening for a connection.
SYN_SENT	Active; have sent SYN. Represents waiting for a matching conn request after having sent a connection request.
SYN_RCVD	Have sent and received SYN. Represents waiting for a confirmi connection request acknowledgment after having both received sent connection requests.
ESTABLISHED	Connection established.
CLOSE_WAIT	Have received FIN; waiting to receive CLOSE.
LAST_ACK	Have received FIN and CLOSE; awaiting FIN ACK.
FIN_WAIT_1	Have closed; sent FIN.
CLOSING	Closed; exchanged FIN; awaiting FIN.
FIN_WAIT_2	Have closed; FIN is acknowledged; awaiting FIN.
TIME_WAIT	In 2MSL (twice the maximum segment length) quiet wait after

Action Specifies the current status of the packet trace connection. The output of the comm changes depending on the action.

Input Receiving input packets. The syntax of the output is:

```
input (SourceAddress, Port, DestinationAddress,
Port) <Sequence Number of the First Data Octet> @
AcknowledgementNumber
```

as in the following example:

```
input (src=129.353173176,23, dst=129.35.17.140, 1795) fb9f5461@fb
```

Output Transmitting packets. The syntax of the output is:

```
output (SourceAddress, Port, DestinationAddress,
Port) <Sequence Number Of The First Data Octet>..
<Sequence Number of the Last Data Octet>@
AcknowledgementNumber)
```

as in the following example:

```
output (src=129.35.17.140,1795, dst=129.35.17.176, 23) fb9e4c68@fb
```

Window Size Specifies the size of the window sending or rec packets, as shown in the following example:

```
(win=1000)
```

User Specifies user request. The following is an exar a user request:

```
SLOWTIMO<KEEP>
```

Types of user requests and their definitions foll

PRU_ATTACH	Attach protocol to up.
PRU−DETACH	Detach protocol from u
PRU_BIND	Bind socket to address.
PRU_LISTEN	Listen for connection.

PRU_CONNECT	Establish connection to
PRU_ACCEPT	Accept connection from
PRU_DISCONNECT	Disconnect from peer.
PRU_SHUTDOWN	Will not send any more
PRU_RCVD	Have taken data; more now.
PRU_SEND	Send this data.
PRU_ABORT	Abort (fast DISCONNE DETACH).
PRU_CONTROL	Control operations on protocol.
PRU_SENSE	Return status into m.
PRU_RCVOOB	Retrieve out of band da
PRU_SENDOOB	Send out of band data.
PRU_SOCKADDR	Fetch socket's address.
PRU_PEERADDR	Fetch peer's address.
PRU_CONNECT2	Connect two sockets.
PRU_FASTTIMO	200 milliseconds timeo
PRU_SLOTIMO	500 milliseconds timeo
PRU_PROTORCV	Receive from below.
PRU_PROTOSEND	Send to below.

Drop	Specifies that data was in preceding segment; data is dropped.
Window and Sequence Variables	Types of window and sequence variables follow:
	rcv_nxt Next sequence number expected on incoming segments.
	rcv_wnd Size of receive window.
	snd_una Oldest unacknowledged sequence number.
	snd_nxt Next sequence number to be sent.
	snd_max Highest sequence number sent.
	snd_sl1 Window update segment sequence number.
	snd_wl1 Window update segment ack number.
	snd_wnd Send window.

Flags

−a	Prints the values of the source and destination addresses for each packet recorded, in addition to the normal output.
−f	Follows the trace as it occurs, waiting briefly for additional records each time the end of the log is reached.
−j	Lists just the protocol control block addresses for which trace records exist.
−p*Address*	Shows only trace records associated with the protocol control block specified in hexadecimal by the *Address* variable. You must repeat the **−p** flag with each *Address* variable specified.
−s	Prints a detailed description of the packet−sequencing information, in addition to the normal output.
−t	Prints the values for all timers at each point in the trace, in addition to the normal output.

Examples

1. To print trace information as well as the source and destination addresses for each packet recorded, enter:

   ```
   $ trpt -a
   ```

   This might display the following output:

   ```
   124b0c:
   900 ESTABLISHED:input (src=192.9.201.3,4257, dst=192.9.201.2,102
   5)2326e6e5@ad938c02(win=200)<ACK,FIN,PUSH> -> CLOSE_WAIT
   900 CLOSE_WAIT:output (src=192.9.201.2,1025, dst=192.9.201.3,425
   7)ad938c02@2326e6e6(win=4000)<ACK> -> CLOSE_WAIT
   900 LAST_ACK:output (src=192.9.201.2,1025, dst=192.9.201.3,4257)
   ad938c02@2326e6e6(win=4000)<ACK,FIN> -> LAST_ACK
   900 CLOSE_WAIT:user DISCONNECT -> LAST_ACK
   900 LAST_ACK:user DETACH -> LAST_ACK 12500c:
   800 ESTABLISHED:output (src=192.9.201.2,1024, dst=192.9.201.3,51
   2)ad8eaa13@2326e6e5(win=4000)<ACK> -> ESTABLISHED
   800 ESTABLISHED:input (src=192.9.201.3,512, \
   dst=192.9.201.2,1024)
   [2326e6e5..2326e727)@ad8eaa13(win=1ef)<ACK,PUSH> -> ESTABLISHED
   800 ESTABLISHED:user RCVD -> ESTABLISHED
   900 ESTABLISHED:output (src=192.9.201.2,1024, dst=192.9.201.3,51
   2)ad8eaa13@2326e727(win=4000)<ACK> -> ESTABLISHED
   900 ESTABLISHED:input (src=192.9.201.3,512, \
   dst=192.9.201.2,1024)
   [2326e727..2326e82f)@ad8eaa13(win=1ef)<ACK,PUSH> -> ESTABLISHED
   900 ESTABLISHED:user RCVD -> ESTABLISHED
   900 ESTABLISHED:output (src=192.9.201.2,1024, dst=192.9.201.3,51
   2)ad8eaa13@2326e82f(win=4000)<ACK> -> ESTABLISHED
   900 ESTABLISHED:input (src=192.9.201.3,512, \
   dst=192.9.201.2,1024)
   2326e82f@ad8eaa13(win=1ef)<ACK,FIN,PUSH> -> CLOSE_WAIT
   900 CLOSE_WAIT:output (src=192.9.201.2,1024, \
   dst=192.9.201.3,512)
   ad8eaa13@2326e830(win=4000)<ACK> -> CLOSE_WAIT
   900 LAST_ACK:output (src=192.9.201.2,1024, dst=192.9.201.3,512)a
   d8eaa13@2326e830(win=4000)<ACK,FIN> -> LAST_ACK
   900 CLOSE_WAIT:user DISCONNECT -> LAST_ACK
   900 LAST_ACK:user DETACH -> LAST_ACK
   $ _
   ```

2. To list the protocol control blocks that have trace records, enter:

   ```
   trpt -j
   ```

   This might display the following output:

   ```
   124b0c, 12500c
   ```

3. To print the trace records associated with a single protocol control block, enter:

   ```
   trpt -p 12500c
   ```

   This might display the following output:

   ```
   800 ESTABLISHED:output ad8eaa13@2326e6e5(win=4000)<ACK> ->
   ESTABLISHED
   800 ESTABLISHED:input [2326e6e5..2326e727)@ad8eaa13(win=1ef)
   <ACK,PUSH> -> ESTABLISHED
   ```

```
800 ESTABLISHED:user RCVD -> ESTABLISHED
900 ESTABLISHED:output ad8eaa13@2326e727(win=4000)<ACK> -> ESTABLISHED
900 ESTABLISHED:input [2326e727..2326e82f)@ad8eaa13(win=1ef) <ACK,PUSH> -> ESTABLISHED
900 ESTABLISHED:user RCVD -> ESTABLISHED
900 ESTABLISHED:output ad8eaa13@2326e82f(win=4000)<ACK> -> ESTABLISHED
900 ESTABLISHED:input 2326e82f@ad8eaa13(win=1ef)<ACK,FIN,PUSH> -> CLOSE_WAIT
900 CLOSE_WAIT:output ad8eaa13@2326e830(win=4000)<ACK> -> CLOSE_WAIT
900 LAST_ACK:output ad8eaa13@2326e830(win=4000)<ACK,FIN> -> LAST_ACK
900 CLOSE_WAIT:user DISCONNECT -> LAST_ACK
900 LAST_ACK:user DETACH -> LAST_ACK
$ _
```

Related Information

The **netstat** command, **tracesoff** command, **traceson** command.

The **setsockopt** subroutine.

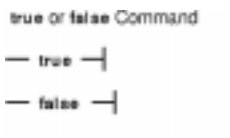
TCP/IP Overview, TCP/IP Protocols, TCP/IP Routing in *AIX Version 4.3 System Management Guide: Communications and Networks*.

true or false Command

Purpose

Returns an exit value of zero (true) or a nonzero exit value (false).

Syntax



true

false

Description

The **true** command returns a zero exit value. The **false** command returns a nonzero exit value. These commands are most often used as part of a shell script.

Examples

To construct a loop that displays the date and time once each minute, use the following code in a shell script:

```
while true
do
   date
   sleep 60
done
```

Related Information

Creating and Running a Shell Script in *AIX Version 4.3 System User's Guide: Operating System and Devices* provides information on creating and executing shell procedures.

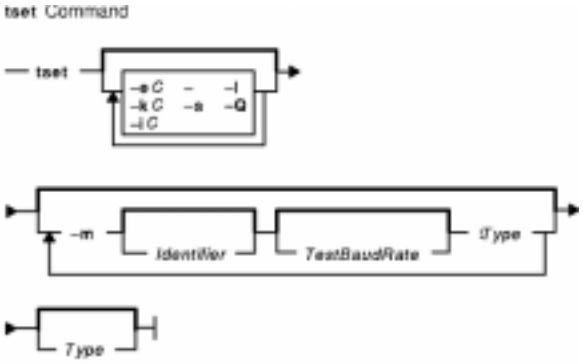
Commands Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

tset Command

Purpose

Initializes terminals.

Syntax



tset [**−e***C*] [**−k***C*] [**−i***C*] [**−**] [**−s**] [**−l**] [**−Q**] [**−m** [*Identifier*] [*TestBaudRate*] **:***Type*] ...
[*Type*]

Description

The **tset** command enables you to set the characteristics of your terminal. It performs terminal−dependent processing, such as setting erase and kill characters, setting or resetting delays, and sending any sequences needed to properly initialize the terminal.

The **tset** command first determines the type of terminal involved (specified by the *Type* flag). It then performs necessary initializations and mode settings. The type of terminal attached to each port is specified in the Object Data Manager (ODM) database. The terminfo database contains possible type names for terminals. If a port is not wired permanently to a specific terminal (that is, it is not hardwired), the **tset** command gives it an appropriate generic identifier, such as `dialup`.

When no flags are specified, the **tset** command reads the terminal type out of the **TERM** environment variable and re−initializes the terminal.

When the **tset** command is used in a startup script (the **.profile** file for **sh** users or the **.login** file for **csh** users), the script should include information about the type of terminal you will usually use on ports that are not hardwired. These ports are identified in the ODM database as `dialup,plugboard,` or `ARPANET`, among others. To specify which terminal type you usually use on these ports, use the **−m** flag (followed by the appropriate port type identifier), an optional baud rate specification, and the terminal type. If more than one mapping is specified, the first applicable mapping prevails. A missing port type identifier matches all identifiers. Any of the alternate generic names given in the **terminfo** database can be used as the identifier.

You can specify the baud rate in the **tset** command as you would with the **stty** command. The baud rate is compared with the speed of the diagnostic output (which should be the control terminal). The baud rate test can be any combination of the following characters:

- `.` (period)
- `@` (at sign)

- < (less than sign)
- ! (exclamation point)

The @ (at sign) stands for the preposition at, and the ! (exclamation point) inverts the sense of the test. To avoid problems with metacharacters, place the **−m** flag argument inside '' (single quotes). Users of the **csh** command must also put a \ (backslash) before any ! (exclamation point).

The following example sets the terminal type to adm3a if the port in use is a dialup at a speed greater than 300 baud. It sets the terminal type to dw2 if the port is a dialup port at a speed of 300 baud or less:

```
tset −m 'dialup>300:adm3a' −m dialup:dw2 −m 'plugboard:?adm3a'
```

If the *Type* parameter begins with a question mark (?), you are prompted to verify the type. To use the specified type, press Enter. To use a different type, enter the type you want. In the example given, you are prompted to verify the adm3 plugboard port type.

If no mapping applies and a final type option (not preceded by an **−m** flag) is given on the command line, that type is used. Otherwise, the default terminal type is the one identified in the ODM database. Hardwired ports should always be identified in the ODM database.

Once the terminal type is known, the **tset** command engages in terminal driver mode setting. This normally involves setting:

- An initialization sequence to the terminal
- The single character erase and optionally the line−kill (full−line erase) characters
- Special character delays

Tab and new−line expansion are turned off during transmission of the terminal initialization sequence.

On terminals that can backspace but not overstrike (such as a CRT), and when the erase character is the default erase character (# on standard systems), the erase character is changed to Backspace (Ctrl−H).

Flags

−e*C*	Sets the erase character to the character specified by the C variable.The default is the backspace character.
−I	Suppresses transmission of terminal initialization strings.
−i*C*	Sets the interrupt character to the character specified by the C variable. The C variable defaults to ^C. The ^ character can also be used for this option.
−k*C*	Sets the line−kill character to the character specified by the C variable. The C variable defaults to ^X. The ^ character can also be used for this option.
−m*IdentifierTestbaudRate*:*Type*	Specifies which terminal type (in the *Type* parameter) is usually used on the port identified in the *Identifier* parameter. A missing identifier matches all identifiers. You can optionally specify the baud rate in the *TestBaudRate* parameter.
−Q	Suppresses printing of the Erase set to and Kill set to messages.
−s	Prints the sequence of **csh** commands that initialize the **TERM** environment variable, based on the name of the terminal decided upon.
−	The name of the terminal decided upon is output to standard output. This is the **TERM** environment variable.

Examples

The following examples all assume the Bourne shell and usage of the − flag. If you use the **csh** command, use the preceding variations. A typical use of the **tset** command in a **.profile** or **.login** file includes the **−e** and **−k** flags, and often the **−n** or **−Q** flags as well. To streamline the examples, these flags have not been included here.

> **Note:** Make sure to enter the **tset** command all on one line regardless of the number of lines used in the example.

1. At the moment, you are a 2621 terminal. Do not use the following example in your **.profile** file, unless you are always a 2621 terminal.

   ```
   export TERM; TERM=\'tset \− 2621\'
   ```

2. You have an h19 terminal at home that you dial up on, but your office terminal is hardwired and specified in the ODM database.

   ```
   export TERM; TERM=\'tset \− \−m dialup:h19"'
   ```

3. You have a switch that connects everything to everything, making it nearly impossible to key on what port you are coming in. You use a vt100 in your office at 9600 baud and dial up from home on a 2621 to switch ports at 1200 baud. Sometimes, you use a different terminal at work. At high speeds, you want to verify your terminal type, but at 1200 baud, you are always on a 2621. Note how the quotation marks protect the greater−than sign and the question mark from interpretation by the shell.

   ```
   export TERM; TERM=\'tset \− \−m 'switch>1200:?vt100' \−m
   'switch<=1200:2621'
   ```

   If none of the conditions hold, the terminal type specified in the ODM database is used.

4. The following entry is appropriate if you always dial up at the same baud rate on many different terminals. Your most common terminal is an `adm3a`. You are always prompted to verify the terminal type, which defaults to `adm3a`.

   ```
   export TERM; TERM=\'tset \− \?adm3a\'
   ```

5. If the ODM database is not properly installed and you want to key entirely on the baud rate, enter:

   ```
   export TERM; TERM=\'tset \− \−m 'switch>1200:?vt100' \−m
   'switch<=1200:2621'
   ```

6. You dial up at 1200 baud or less on a Concept100, sometimes over switch ports and sometimes over regular dialups. You use various terminals at speeds higher than 1200 over switch ports, most often the terminal in your office, which is a vt100. However, sometimes you log in from the university over the ARPANET; in this case, you are on an ALTO emulating a dm2500. You also often log in on various hardwired ports, such as the console, all of which are properly entered in the ODM database. To set your erase character to Ctrl−H and your kill character to Ctrl−U, enter:

   ```
   export TERM
   TERM=\'tset \−e \−k(hat)U \−Q \− "−m 'switch<1200:concept100'
   "−m 'switch:?vt100' \−m dialup:concept100 "1−m arpanet: dm2500"'
   ```

   This also prevents the **tset** command from printing the following line:

   ```
   Erase set to Backspace, Kill set to Ctrl−U
   ```

7. To set the erase character to a control character, enter:

   ```
   tset −e ^Y
   ```

File

/usr/share/lib/terminfo Contains the terminal capability database.

Related Information

The **csh** command, **reset** command, **sh** command, **stty** command.

The **environ** file, **terminfo** file.

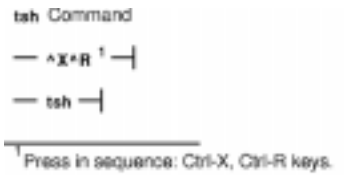
TERM Values for Different Displays and Terminals section of tty Overview for System Managers in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

tsh Command

Purpose

Invokes the trusted shell.

Syntax



Press in sequence: the Ctrl−X, Ctrl−R keys.

tsh Command

Description

The **tsh** command is a command interpreter that provides greater security than the Korn shell (the standard login shell). Generally, a user calls the **tsh** shell by pressing Ctrl−X, Ctrl−R, the secure attention key (SAK) sequence, after a login. The **tsh** shell also can be invoked by defining it as the login shell in the **/etc/passwd** file.

To use the SAK sequence to invoke the trusted shell, the terminal the user is using must have SAK enabled, and the user must be allowed to use the trusted path. See the Trusted Computing Base Overview in *AIX Version 4.3 System Management Guide: Operating System and Devices* for information on enabling SAK on a terminal, and see the **/etc/security/user** file and the **chuser** command for information on allowing a user to access the trusted path.

To exit from the **tsh** shell, use any of the following commands: the **logout** command, **shell** command, **su** command. The **logout** command ends the login session, while the other commands execute the user's initial program and continue the login session.

The trusted shell differs from the Korn shell in the following ways:

- The function and alias definitions are not supported. Alias definitions are only supported in the **/etc/tsh_profile** file.
- The **IFS** and **PATH** environment variables cannot be redefined.
- Only trusted programs can be run from the **tsh** shell.
- The history mechanism is not supported.
- The only profile used is the **/etc/tsh_profile** file.
- The trusted shell has the following built−in commands:

  logout Exits the login session and terminates all processes.

  shell Re−initializes the user's login session. The effect is the same as logging in to the system.

  su Resets the effective ID to the user's identity on the system and executes another trusted shell.

Security

Access Control: This command should be a standard user program and have the **trusted computing base** attribute.

Files Accessed:

Mode	File
r	**/etc/tsh_profile**

Examples

To invoke the trusted shell, press the Ctrl−X, Ctrl−R key sequence, the secure attention key (SAK).

Files

/usr/bin/tsh	Contains the **tsh** command.
/etc/tsh_profile	Contains initialization commands for the trusted shell.
/etc/passwd	Contains basic user attributes.
/etc/security/user	Contains the extended attributes of users.
/etc/security/login.cfg	Contains configuration information.

Related Information

The **chuser** command, **init** command, **ksh** command, **logout** command, **shell** command, **su** command, **tsm** command.

See National Language Support Overview for Programming in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs* for more information about Single−Source Dual Object (SSDO) commands used during installation.

For more information about the identification and authentication of users, discretionary access control, the trusted computing base, and auditing, refer to the Security Administration in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

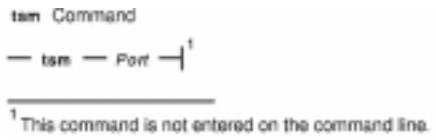
See Trusted Computing Base Overview in *AIX Version 4.3 System Management Guide: Operating System and Devices* for more information about the trusted path and enabling SAK on a terminal.

tsm Command

Purpose

Provides terminal state management.

Syntax



tsm *Port*

Description

The **tsm** command invokes the terminal state manager, which controls the ports used in the trusted path. The functions are:

- Establishing line communication modes and discipline − functions performed by the **getty** command.
- Verifying the user's account and identity, and setting the initial process credentials and environment − functions performed by the **login** command.
- Performing trusted path management if the secure attention key (SAK) is enabled for the port and the system login program is used.

  Note: The **tsm** command is not entered on the command line.

Trusted path management occurs in two phases:

login This phase is in effect if a user has not successfully logged in. If the secure attention key (SAK) signal is detected, the system restarts getty−login type processing. The next login puts the user into the trusted state, if the port and the user support the trusted state.

shell This phase occurs after successful user authentication. The command functions according to the user's **tpath** attribute. The following are valid:

- **on** Provides standard trusted path management. When the secure attention key (SAK) signal is detected, all processes that access the port, except the **tsm** process and its siblings (including the trusted shell), are terminated the next time an attempt is made to access the port. The port is reset to its initial state and is marked as trusted, and the trusted shell command (the **tsh** command) is executed.
- **notsh** The user session terminates when the secure attention key (SAK) signal is detected.
- **always** The user is not allowed off the trusted path. The user's shell will always be the trusted shell, tsh.
- **nosak** The secure attention key (SAK) is disabled for the terminal, and the user's initial program runs.

Security

Access Control: This command should grant execute (x) permission to any user. The command should be setuid to the root user and have the **trusted computing base** attribute.

Files Accessed:

Mode	File
r	**/etc/objrepos/CuAt**
r	**/usr/lib/objrepos/PdAt**
r	**/etc/security/login.cfg**
r	**/etc/security/user**

Examples

To provide terminal state management on `tty0`, add the following line to the **/etc/inittab** file:

```
tty0:2:respawn:/usr/sbin/tsm /dev/tty0
```

This initializes the port `/dev/tty0` and sets up the characteristics of the port.

Files

/usr/sbin/tsm	Contains the **tsm** command.
/etc/security/login.cfg	Contains configuration information.
/etc/security/user	Contains extended user attributes.

Related Information

The **getty** command, **init** command, **login** command, **logout** command, **setgroups** command, **shell** command, **su** command, **tsh** command.

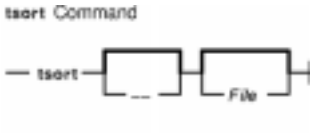
For more information about the identification and authentication of users, discretionary access control, the trusted computing base, and auditing, refer to the Security Administration in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

tsort Command

Purpose

Sorts an unordered list of ordered pairs (a topological sort).

Syntax



tsort [−−] [*File*]

Description

The **tsort** command reads from *File* or standard input an unordered list of ordered pairs, builds a completely ordered list, and writes it to standard output.

The input *File* should contain pairs of non−empty strings separated by blanks. Pairs of different items indicate a relative order. Pairs of identical items indicate presence, but no relative order. You can use the **tsort** command to sort the output of the **lorder** command.

If *File* contains an odd number of fields, an appropriate error message is displayed.

Flag

−− (Double hyphen) Interprets all arguments following the −− flag as file names. If the file is named −−, use tsort −− −−.

Exit Status

This command returns the following exit values:

0 Successful completion.
>0 An error occurred.

Files

/usr/ccs/bin/tsort Contains the **tsort** command.
/usr/ccs/bin/tsort Contains symbolic link to the **tsort** command.

Related Information

The **ar** command, **ld** command, **lorder** command, **xargs** command.

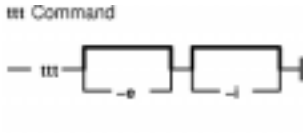
The Commands Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

ttt Command

Purpose

Starts the tic−tac−toe game.

Syntax



ttt [**−e**] [**i**]

Description

The **ttt** command starts the tic−tac−toe game. This is a learning version but it learns slowly. It loses nearly 80 games before completely mastering the game. When you start the game you are prompted `Accumulated knowledge? (Yes or No)`. Entering `y` provides the computer with knowledge gained from previous games.

You are always X and your opponent is always O. You can either make the first move or pass to your opponent. To pass, press the enter key when prompted `Your move?` at the beginning of the game. The first to get three in a row wins the game. For example:

```
new game
123
456
789
Your move?
1
XO3
456
789
Your move?
9
XOO
456
78X
Your move?
5
You win
```

In the example, your first move was to place an X where 1 was located. The computer placed an O where the 2 was located. The game progressed until you had three in a diagonal row (1,5, 9). The game repeats until you quit. To quit the game, press the Interrupt (Ctrl−C) or End Of File (Ctrl−D) key sequence**.**

Flags

−e Increases the speed of the learning.

−i Displays the instructions prior to the start of the game.

Files

$HOME/ttt.a	Specifies the location of the learning file.
/usr/games	Specifies the location of the system's games.

Related Information

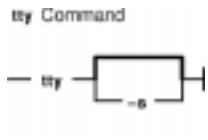
The **arithmetic** command, **back** command, **bj** command, **craps** command, **fish** command, **fortune** command, **hangman** command, **moo** command, **number** command, **quiz** command, **turnoff** command, **turnon** command, **wump** command.

tty Command

Purpose

Writes to standard output the full path name of your terminal.

Syntax



/usr/bin/tty [**−s**]

Description

The **tty** command writes the name of your terminal to standard output.

If your standard input is not a terminal and you do not specify the **−s** flag, you get the message `Standard input is not a tty`.

The following environment variables affect the execution of the **tty** command:

LANG	Determines the locale to use for the locale categories when neither the **LC_ALL** variable nor the corresponding environment variable beginning with **LC_** specifies a locale.
LC_ALL	Determines the locale to be used. This variable overrides any values for locale categories that are specified by any other environment variable beginning with **LC_** or by the **LANG** variable.
LC_CTYPE	Determines the locale for the interpretation of sequences of bytes of text data as characters. For example, this variable may specify multi−byte characters instead of single−byte characters.
LC_MESSAGES	Determines the language for messages.

Flags

−s Suppresses reporting the path name.

Exit Status

This command returns the following exit values:

0 Standard input is a terminal.
1 Standard input is not a terminal.
>1 An error occurred.

Examples

1. To display the full path name of your display:

```
tty
```

2. To test whether or not the standard input is a terminal:

```
if tty -s
then
 echo 'Enter the text to print:' >/dev/tty
fi
qprt
```

If the standard input is a terminal, this displays the message `"Enter the text to print:"` as a prompt and prints the text that the user types. If the standard input is not a terminal, this displays nothing; it merely prints the text read from the standard input.

The `echo . . . >/dev/tty` displays the prompt on the screen even if you redirect the standard output of the shell procedure. This way the prompt is never written into an output file. The special file **/dev/tty** always refers to your terminal, although it also has another name such as **/dev/console** or **/dev/tty2**.

Files

/usr/bin/tty	Contains the **tty** command.
/dev/tty	Specifies the **tty** pseudo device.

Related Information

The National Language Support Overview for Programming in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs* discusses the **LC_** variables.

turnacct Command

Purpose

Provides an interface to the **accton** command to turn process accounting on or off.

Syntax

/usr/sbin/acct/turnacct On | **Off** | **Switch**

Description

The **turnacct** command provides an interface to the **accton** command to turn process accounting on or off. You must specify whether you want process accounting on or off, because there is no default.

The **switch** flag turns off accounting and moves the current active data file (**/var/adm/pacct**) to the next free name in the **/var/adm/pacct***incr* file, where *incr* is a number starting at 1 and increased by one for each additional **pacct** file. After moving the **pacct** file, the **turnacct** command again turns on accounting.

The **turnacct switch** command is usually called by the **ckpacct** command, running under the **cron** daemon, to keep the active **pacct** data file a manageable size.

Security

Access Control: This command should grant execute (x) access only to members of the adm group.

Files

/usr/sbin/acct	Contains the path to the accounting commands.
/var/adm/pacct	Contains the current file for process accounting.
/var/adm/pacct*	Used if the **pacct** file gets too large.

Related Information

The **accton** command, **ckpacct** command.

The **cron** daemon.

Setting Up an Accounting System in *AIX Version 4.3 System Management Guide: Operating System and Devices* explains the steps you must take to establish an accounting system.

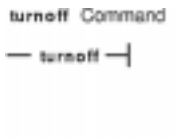
For more information about the Accounting System, the preparation of daily and monthly reports, and the accounting files, see the Accounting Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

turnoff Command

Purpose

Sets the permission codes off for files in the **/usr/games** directory.

Syntax



turnoff

Description

The **turnoff** command sets the permission codes of files in the **/usr/games** directory. Root user authority is required to run this command.

The **turnoff** command looks for files in **/usr/games** whose permissions are set to 111 and sets these permissions to 000. If you install any new games in the **/usr/games** directory, set these permissions to 111.

Files

/usr/games Contains the location of the system's games.

Related Information

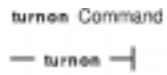
The **arithmetic** command, **back** command, **bj** command, **craps** command, **fish** command, **fortune** command, **hangman** command, **moo** command, **number** command, **quiz** command, **ttt** command, **turnon** command, **wump** command.

turnon Command

Purpose

Sets permission codes on for files in the games directory.

Syntax



turnon

Description

The **turnon** command sets the permission codes of files in the **/usr/games** directory. Root user authority is required to run this command.

The **turnon** command looks for files with permissions set to 000 and sets them to 111 (execute permission for all users). If you install any new games in the **/usr/games** directory, set these permissions to 111.

File

/usr/games Contains the location of the system's games.

Related Information

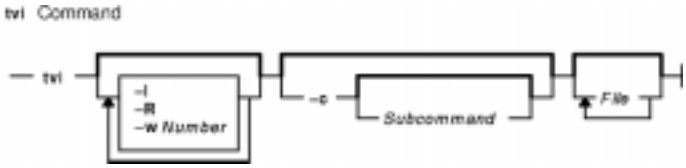
The **arithmetic** command, **back** command, **bj** command, **craps** command, **fish** command, **fortune** command, **hangman** command, **moo** command, **number** command, **quiz** command, **ttt** command, **turnoff** command, **wump** command.

tvi Command

Purpose

Provides a trusted editor with a full screen display.

Syntax



tvi [**−l**] [**−R**] [**−w***Number*] [**−c** [*Subcommand*]] [*File ...*]

Description

The **tvi** command calls the **tvi** editor, a trusted version of the **vi** editor, to edit the file or files specified by the *File* parameter. Files are edited in the order specified. If you do not provide a file name, the command opens a new file in which you can create text, but if you try to save the text to a file, you are prompted to add a file name to the save command, such as **:w** *File*. See the Examples section for more information.

You enter and leave the **tvi** editor in command mode, but to add or change text, you must enter text input mode. See the description of text input mode for information about the subcommands that initiate text input mode. To leave text input mode, press the **Esc** key. This returns you to command mode where you can save the text to a file with one of the **:w** commands, and exit the **tvi** editor, for example, with the **:q** command.

Since the full−screen display editor started by the **tvi** command is based on the **ex** editor, you can use the **ex** subcommands within the **tvi** editor. Subcommands function at the cursor position on the display screen.

The **tvi** editor makes a copy of the file you are editing in an edit buffer. The contents of the file are not changed until you save the changes.

> **Note:** Several functions of the **vi** editor are not supported by the **tvi** editor. If you refer to information on the **vi** editor, be aware that the **−r** flag, the **−t** flag, shell escapes, user−defined macros, key mapping, and setting **vi** options permanently are not supported by the **tvi** editor.

tvi Editor Limitations

The maximum limits of the **tvi** editor assume single−byte characters. The limits are as follows:

- 2048 characters per line
- 256 characters per global command list
- 128 characters in the previously inserted and deleted text
- 128 characters in a shell escape command
- 128 characters in a string−valued option
- 30 characters in a tag name
- 524,230 lines of 2048 characters per line silently enforced
- 128 map macros with 2048 characters total

Editing Modes

The **tvi** editor operates in the following modes:

command mode The **tvi** editor starts in command mode. Any subcommand can be called except those that only correct text during text input mode. To see a description of the subcommands, refer to the topics in Subcommands for the tvi Editor. To identify the subcommands that cannot be called from command mode, refer to Changing Text While in Input Mode. The **tvi** editor returns to command mode when subcommands and other modes end. Press the Esc key to cancel a partial subcommand.

text input mode The **tvi** editor enters text input mode when you use a permitted command that adds or changes text. To see a list of subcommands that initiate text input mode, refer to Adding Text to a File and the subcommands that change text from command mode, the **C** subcommand and the **c***x* subcommands. After entering one of these subcommands, you can edit text with any of the subcommands that function in text input mode. To see a description of the subcommands, refer to the topics in Subcommands for the tvi Editor. To return to command mode from text input mode, press the Esc key for a normal exit or press the Ctrl−C keys to create an **INTERRUPT** signal.

last line mode Some subcommands read input on a line displayed at the bottom of the screen. These subcommands include those with the prefix : (colon), / (slash), and ? (question mark). When you enter the initial character, the **tvi** editor places the cursor at the bottom of the screen so you can enter the remaining command characters. To run the subcommand, press the Enter key. To cancel the subcommand, press the Ctrl−C keys to create an **INTERRUPT** signal. When you use the : (colon) to enter last line mode, the following characters have special meaning when used before commands that specify counts:

% All lines regardless of cursor position

$ Last line

. Current line

Customizing the tvi Editor

You can customize the **tvi** editor on a temporary basis by following the directions in "Setting vi Editor Options". The section on "Setting vi Options Permanently" is *not* applicable to the **tvi** editor.

Subcommands for the tvi Editor

Information on **vi** editor subcommands that are applicable to the **tvi** editor is summarized in the following list (refer to the **vi** command documentation and *AIX Version 4.3 INed Editor User's Guide* for this information):

- vi General Subcommand Syntax.
- vi Subcommands for Adjusting the Screen.
- Editing Text with the **vi** Editor.
- Entering Shell Commands within the **vi** Editor is *not* supported by the **tvi** editor.
- Manipulating Files with the **vi** Editor.
- Subcommands for Interrupting and Ending the **vi** Editor.

Flags

−c [*Subcommand***]** Carries out the **ex** editor subcommand before editing begins. This provides a line−oriented text editor. When a null operand is entered for the *Subcommand* parameter, as in `−c ''`, the editor places the cursor on the last line of the file.

−l Enters the editor in LISP mode. In this mode, the editor indents appropriately for LISP code, and the **(**, **)**, **{**, **}**, **[[**, and **]]** subcommands are modified to act appropriately for LISP. These subcommands place the cursor at the specified LISP function. For more

	information on the LISP subcommands, refer to Moving to Sentences, Paragraphs, and Sections.
−R	Sets the **readonly** option to protect the file against overwriting.
−w *Number*	Sets the default window size to the value specified by the *Number* parameter. This is useful when you use the editor over a low−speed line.
+ [*Subcommand***]**	Same as the **−c** Subcommand.

Security

Access Control: This command should grant execute (x) access to all users and have the **trusted computing base** attribute.

Auditing Events:

Event	Information
TVI	filename

Examples

1. To call a trusted editor to edit the `plans` file, enter:

   ```
   tvi plans
   ```

   This command puts the **tvi** editor into command mode. To add or change text, you must enter text input mode or use a command accepted in command mode. For more information, refer to the description of text input mode.

2. To save the text you create with the **tvi** editor, leave text input mode by pressing the Esc key, and then enter one of the save commands **:w**, **:w** *File*, or **:w!***File*, for example:

   ```
   :w plans
   ```

   In this example, a file name, such as `plans`, is needed if you gave the **tvi** command without specifying a file name. If the file is already named, the **:w** command would not need the *File* parameter. If you want to overwrite an existing file, use the **:w!***File* command, specifying the file you want to overwrite with the *File* parameter.

   If you try to save an unnamed file without supplying a file name, the following message appears:

   ```
   No current filename
   ```

   If this happens, repeat the **:w** command with a file name.

3. To exit the **tvi** editor from text input mode, press the Esc key to enter command mode, and then enter:

   ```
   :q!
   ```

If the editor already is in command mode, you do not need to press the Esc key before giving the quit (**q!**) command.

File

/usr/bin/tvi Contains the **tvi** command.

Related Information

The **ex** command, **vi** command.

Editors Overview in *AIX Version 4.3 INed Editor User's Guide* gives an overview of various editors, including the **vi** editor.

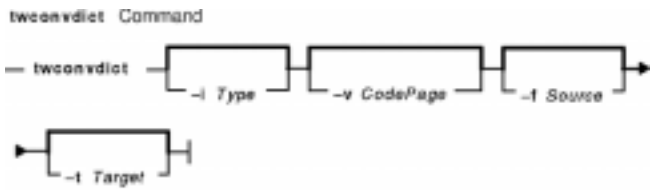
For more information about the identification and authentication of users, discretionary access control, the trusted computing base, and auditing, refer to the Security Administration in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

twconvdict Command

Purpose

Converts other user dictionary to the AIX user dictionary. This command only applies to AIX Version 4.2 or higher.

Syntax



twconvdict [**−i** *Type*] [**−v** *CodePage*] [**−f** *Source*] [**−t** *Target*]

Description

The **twconvdict** command converts a dictionary to an AIX user dictionary. The supported code pages are SOPS, PS55 and ET. The dictionary type include both Tseng_Jye and Phonetic user dictionaries.

Flags

−f *Source*	Specifies the name of font file to convert.
−i *Type*	Specifies the type of dictionary to convert to. *Type* can be: **TJ** Tseng_Jye, or **PH** Phonetic.
−t *Target*	Specifies the name of the converted font file.
−v *CodePage*	Specifies the type of code page to convert to. *CodePage* can be: **SOPS** **PS55**, or **ET**.

Exit Status

This command returns the following exit values:

0 Successful completion.
>0 An error occurred.

Security

Access Control: You must have root authority to run this command.

Auditing Events: N/A

Examples

To convert the dictionary `USRFONT.C12` to an AIX dictionary of code page of type SOPS and dictioanry type of Tseng_Jye with the name `aix`, enter:

```
twconvdict −i TJ −v SOPS −f USRFONT.C12 −t aix
```

Files

/usr/lpp/tls/bin/twconvdict Contains the **twconvdict** command.

Related Information

None.

twconvfont Command

Purpose

Converts other font files to a BDF font file. This command only applies to AIX Version 4.2 or higher.

Syntax

twconvfont [**−v** *CodePage*] [**−f** *Source*] [**−t** *Target*]

Description

The twconvfont command converts one font file type to the BDF font file. The supported code pages are SOPS, PS55 and ET.

Flags

−f *Source* Specifies the name of font file to convert.

−t *Target* Specifies the name of the converted font file.

−v *CodePage* Specifies the type of code page to convert to. *CodePage* can be:

SOPS

PS55, or

ET.

Exit Status

This command returns the following exit values:

0 Successful completion.

>0 An error occurred.

Security

Access Control: You must have root authority to run this command.

Auditing Events: N/A

Examples

To convert the font file `USRFONT.C12` to a BDF font file of code page of type SOPS with the name `user.bdf`, enter:

```
twconvfont −v SOPS −f USRFONT.C12 −t user.bdf
```

Files

/usr/lpp/tls/bin/twconvfont Contains the **twconvfont** command.

Related Information

None.

type Command

Purpose

Writes a description of the command type.

Syntax

type*CommandName* ...

Description

The standard output of the **type** command contains information about the specified command and identifies whether this is a shell built−in command, subroutine, alias, or keyword. The **type** command indicates how the specified command would be interpreted if used. Where applicable, the **type** command displays the related path name.

Since the **type** command must know the contents of the current shell environment, it is provided as a Korn shell or POSIX shell regular built−in command. If the **type** command is called in a separate command execution environment, the command may not produce accurate results. This would be the case in the following examples:

```
nohup type writer
```

```
find . −type f | xargs type
```

Exit Status

The following exit values are returned:

0 Successful completion.
>0 An error occurred.

Examples

1. To learn whether the **cd** command is a base command or an alias or some other command type, enter:

   ```
   type cd
   ```

   The screen displays the following information:

   ```
   cd is a shell builtin
   ```

2. To see the location of the **find** command, enter:

   ```
   type find
   ```

The screen displays the following information:

```
find is /usr/bin/find
```

Files

/usr/bin/ksh Contains the Korn shell **type** built−in command.

Related Information

The **bsh** command, **command** command, the **ksh** command.

ucfgif Method

Purpose

Unloads an interface instance from the kernel.

Syntax

ucfgif [**−l** *InterfaceInstance*]

Description

The **ucfgif** method removes an interface instance from the kernel. To remove the interface instance, the **ucfgif** method does the following:

1. Unloads the interface software by calling the **/usr/sbin/ifconfig** interface detach.
2. Sets the status flag of the interface instance to **defined**.

   Note: The **ucfgif** method is a programming tool and should not be executed from the command line.

Flags

−l *InterfaceInstance* Specifies the interface instance to be unconfigured. If no interface name is specified, all configured interface instances are unconfigured.

Example

To remove an interface instance from the kernel, enter the method in the following format:

```
ucfgif −l tr0
```

In this example, the name of the interface instance is `tr0`.

Related Information

The **ifconfig** command.

The **odm_run_method** subroutine.

TCP/IP Network Interfaces in *AIX Version 4.3 System Management Guide: Communications and Networks*.

Writing a Device Method in *Kernel Extensions and Device Support Programming Concepts*.

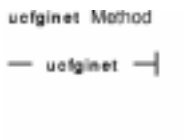
Object Data Manager (ODM) Overview for Programmers in *General Programming Concepts*.

ucfginet Method

Purpose

Unloads the Internet instance and all related interface instances from the kernel.

Syntax



ucfginet

Description

The **ucfginet** method unloads the Internet instance from the kernel. This subroutine also deletes the appropriate entries in the Address Family Domain switch table and in the Network Input Interface switch table. The **ucfginet** method also sets the status flag of the instance to **defined**. The **ucfginet** method is called by the **rmdev** high−level command.

> **Note:** The **ucfginet** method is a programming tool and should not be executed from the command line.

Related Information

The **cfginet** method, **rmdev** command, **ucfgif** method.

The **odm_run_method** subroutine.

TCP/IP Network Interfaces in *AIX Version 4.3 System Management Guide: Communications and Networks*.

Writing a Device Method in *Kernel Extensions and Device Support Programming Concepts*.

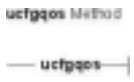
Object Data Manager (ODM) Overview for Programmers in *General Programming Concepts*.

ucfgqos Method

Purpose

Unconfigures and unloads the Quality of Service (QoS) instance from the kernel.

Syntax



ucfgqos

Description

The **ucfgqos** method disables Quality of Service (QoS) for the TCP/IP protocol suite on an AIX host. This method detaches the QoS instance from the TCP/IP instance and unloads it from the kernel.

> **Note:** The **ucfgqos** method is a programming tool and is not intended to be invoked from the command line.

Example

To configure QoS on a host, use the following format:

```
ucfgqos
```

Related Information

The **cfgqos** method, and **ucfginet** method.

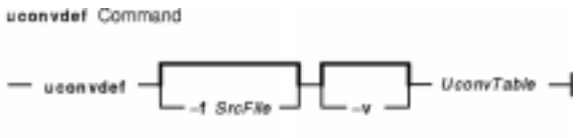
TCP/IP Quality of Service (QoS) in the *AIX Version 4.3 System Management Guide: Communications and Networks*.

uconvdef Command

Purpose

Compiles or generates a UCS−2 (Unicode) conversion table for use by the **iconv** library.

Syntax



uconvdef [**−f***SrcFile*] [**−v**] *UconvTable*

Description

The **uconvdef** command reads *SrcFile* and creates a compiled conversion table in *UconvTable*. The *SrcFile* defines a mapping between UCS−2 and multibyte code sets (one or more bytes per character). The *UconvTable* is in a format that can be loaded by the UCSTBL conversion method located in the **/usr/lib/nls/loc/uconv** directory. This method uses the table to support UCS−2 conversions in both directions.

Flags

−f *SrcFile*	Specifies the conversion table source file. If this flag is not used, standard input is read.
−v	Causes output of the processed file statements.
UconvTable	Specifies the path name of the compiled table created by the **uconvdef** command. This should be the name of the code set that defines conversions into and out of UCS−2.

Exit Status

The following exit values are returned:

0 Successful completion.
>0 An error occurred.

Examples

To access the compiled UCS−2 conversion table:

1. Create the compiled *UconvTable* using the name of the multibyte code set. For example, the conversion table between IBM−850 and UCS−2 can be compiled as follows:

   ```
   uconvdef −f IBM−850.ucmap IBM−850
   ```

2. Place the table in a directory called **uconvTable**. The default system directory is **/usr/lib/nls/loc/uconvTable**. If another directory is used, the **LOCPATH** environment variable needs to be set to include the parent directory (for example, **/usr/lib/nls/loc**).

   ```
   mv IBM−850 /usr/lib/nls/loc/uconvTable
   ```

3. Create symbolic links for conversions in each direction in a directory called **iconv**. The names for these links should be formed by concatenating the "From" code set and the "To" code set, separated by an underscore. The links should be set to point to the **/usr/lib/nls/loc/uconv/UCSTBL** conversion method. The default directory for these links is **/usr/lib/nls/loc/iconv**. If another directory is used, the **LOCPATH** environment variable needs to be set to include the parent directory (for example, **/usr/lib/nls/loc**).

```
ln -s /usr/lib/nls/loc/uconv/UCSTBL \
/usr/lib/nls/loc/iconv/IBM-850_UCS-2


ln -s /usr/lib/nls/loc/uconv/UCSTBL \
/usr/lib/nls/loc/iconv/UCS-2_IBM-850
```

> **Note:** The \ (backslash) is a line continuation character that is only needed if the command is broken into two lines.

Related Information

The **iconv** command.

The **iconv** subroutine, **iconv_close** subroutine, **iconv_open** subroutine.

The **unconvdef** source file format.

Code Set Overview in *AIX Version 4.3 Kernel Extensions and Device Support Programming Concepts*.

Converters Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

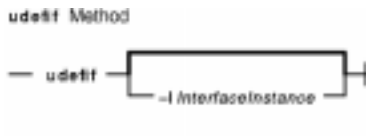
List of UCS−2 Interchange Converters in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

udefif Method

Purpose

Removes an interface object from the system configuration database.

Syntax



udefif [**−l** *InterfaceInstance*]

Description

The **udefif** method deletes the specified interface instance from the system configuration database by:

1. Removing the database object associated with the interface instance.
2. Removing the connection and attribute information associated with the interface instance.

Flags

−l *InterfaceInstance* Specifies the interface instance to be undefined. If no interface instances are specified, the **udefif** method undefines all defined interface instances.

Example

To remove an interface instance from the database, enter a method similar to the following:

```
udefif −l tr0
```

In this example, the interface instance to be removed is `tr0`.

Related Information

The **rmdev** command, **udefinet** method.

The **odm_run_method** subroutine.

TCP/IP Network Interfaces in *AIX Version 4.3 System Management Guide: Communications and Networks*.

Writing a Device Method in *Kernel Extensions and Device Support Programming Concepts*.

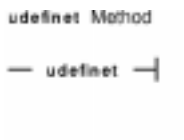
Object Data Manager (ODM) Overview for Programmers in *General Programming Concepts*.

udefinet Method

Purpose

Undefines the Internet instance in the configuration database.

Syntax



udefinet

Description

The **udefinet** method removes the database information associated with the Internet instance, including attribute information associated with the Internet instance.

> **Note:** The **udefinet** method is a programming tool and should not be executed from the command line.

Related Information

The **rmdev** command.

The **odm_run_method** subroutine.

TCP/IP Network Interfaces in *AIX Version 4.3 System Management Guide: Communications and Networks*.

Writing a Device Method in *Kernel Extensions and Device Support Programming Concepts*.

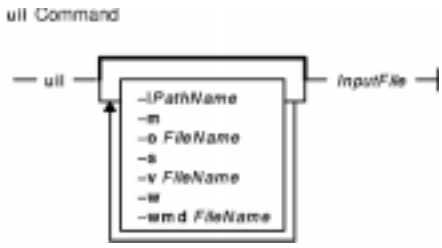
Object Data Manager (ODM) Overview for Programmers in *General Programming Concepts*.

uil Command

Purpose

Starts the User Interface Language (UIL) compiler for the AIXwindows system.

Syntax



uil [**−I***PathName*] *InputFile* [**−m**] [**−o** *FileName*] [**−s**] [**−v** *FileName*] [**−w**]
[**−wmd** *FileName*]

Description

The **uil** command calls the UIL compiler. The UIL is a specification language for describing the initial state of a user interface for an AIXwindows application. The specification describes the objects (menus, dialog boxes, labels, push buttons, and so on) used in the interface and specifies the functions to be called when the interface changes state as a result of user interaction.

Flags

−I*PathName*	Specifies **I**nclude*PathName* with no spaces. Causes the compiler to look for include files in a specified directory if include files were not found in the default paths. (uppercase i)
−m	Specifies that machine code is listed. This directs the compiler to place a description of the records that it added to the User Interface Definition (UID) in the listing file. This helps you isolate errors. The default is no machine code.
−o *FileName*	Directs the compiler to produce a UID. By default, UIL creates a UID with the name **a.uid**. The file specifies the file name for the UID. No UID is produced if the compiler issues any diagnostics categorized as error or severe.
−s	Directs the compiler to set the locale before compiling any files. The locale is set in an implementation−dependent manner. On ANSI C−based systems, the locale is usually set by calling the **setlocale (LC_ALL, "")** function. If this option is not specified, the compiler does not set the locale.
−v *FileName*	Directs the compiler to generate a listing. The file specifies the file name for the listing. If the **−v** option is not present, no listing is generated by the compiler. The default is no listing.
−w	Specifies that the compiler suppress all warning and informational messages. If this option is not present, all messages are generated, regardless of the severity.
−wmd *FileName*	Specifies a binary widget meta−language (WML) description file to be used instead of the default WML description.

Example

To start the UIL compiler, enter:

```
uil −I. −o ex.uid ex.uil
```

Exit Status

This command returns the following exit values:

0 Indicates successful completion.
>0 Indicates an error occurred.

Related Information

The **X** command.

uimx Command

Purpose

Starts the UIM/X user−interface management system for the X Window System.

Syntax

uimx [−dir *Path* **][−file** *FileName*] **[−workspace** *Name* **][−xrm** *Options* **]**

Description

The **uimx** command starts the UIM/X user−interface management system for the X Window System. It supports Motif 1.2 and provides a complete programming environment for developing graphical user interfaces (GUIs). UIM/X supports object−oriented programming in both C and C++.

UIM/X saves and loads text files that use the Xt resource syntax to describe interfaces and projects. It can also load UIL files. It generates C, C++, and UIL code. It can also generate makefiles, message catalogs, and resource files for an application.

UIM/X includes a built−in C Interpreter and the following tools and editors:

- Palette of Motif widgets
- Widget Browser for browsing complex widget hierarchies
- WYSIWYG layout editor for drawing interfaces
- Property Editor for setting initial values of widget properties; initial values can be literal values or C expressions
- Callback Editors for entering callback code
- Event, Action, and Translation Editors
- Menu and Main Window Editors
- Declarations Editor for editing the generated code for an interface
- Program Layout Editor for editing the generated main program and makefile; this editor gives you direct access to the main event loop

UIM/X supports two operating modes: Design and Test. In Test mode, the built−in C Interpreter allows you to test the behavior of your application. In Design mode, the C Interpreter validates the code you enter into the various UIM/X editors.

UIM/X provides a convenience library of functions that simplify the task of programming with X and Motif.

Flags

dir*Path*	Sets UIM/X's current directory to path.
file*FileName*	Loads an existing project, interface, or palette file called *FileName*. *FileName* can include an absolute path name, a path name relative to the current directory, or a path name

	relative to the **−dir** value.
workspace*Name*	Loads UIM/X into the corresponding CDE workspace called *name*.
xrm*Options*	Enables you to enter any resource specifications (*options*) that you would otherwise put in a resource file.

Security

Access Control: Any User

Files Accessed: None

Example

To start UIM/X, enter:

```
uimx
```

Files

/usr/uimx2.8/bin/uimx Contains the **uimx** command.

Related Information

UIM/X 2.8 for AIX: Getting Started with UIM/X, SC23−2557

UIM/X 2.8 for AIX: Developer's Guide, SC23−2558

UIM/X 2.8 for AIX: Extending and Customizing UIM/X, SC23−2559

UIM/X 2.8 for AIX: Installing and Configuring UIM/X, SC23−2570

ul Command

Purpose

Performs underlining.

Syntax

ul [**−i**] [**−t** *Terminal*] [*File ...*]

Description

The **ul** command reads the named files specified by the *File* parameter (or standard input if no file is given) and translates occurrences of underscores to the sequence that indicates underlining for the terminal in use, as specified by the **TERM** environment variable.

Flags

−i Causes the **ul** command to indicate underlining by a separate line containing appropriate _ (underline characters). Use this to see the underlining present in an **nroff** command output stream on a CRT terminal.

−t *Terminal* Overrides the terminal type specified in the environment. The **terminfo** file is read to determine the appropriate sequences for underlining. If the terminal is incapable of underlining, but is capable of a standout mode, then that mode is used instead. If the terminal can overstrike or automatically underline, the **ul** command acts like the **cat** command and displays on the screen. If the terminal cannot underline and no alternatives are available, underlining is ignored.

If the **−t** flag is not specified, the **ul** command translates for the terminal type specified by the **TERM** environment variable. If the value of the *Terminal* variable is not a valid terminal type, the **ul** command translates for a dumb terminal.

Files

/usr/share/lib/terminfo/* Contains the terminal capabilities database.

Related Information

The **cat** command, **colcrt** command, **man** command, **nroff** command.

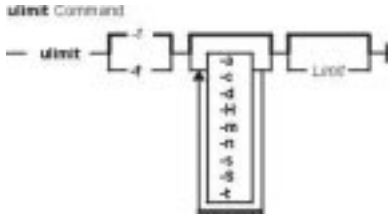
The **terminfo** file.

ulimit Command

Purpose

Sets or reports user resource limits.

Syntax



ulimit [**−H**] [**−S**] [**−a**] [**−c**] [**−d**] [**−f**] [**−m**] [**−n**] [**−s**] [**−t**] [*Limit*]

Description

The **ulimit** command sets or reports user process resource limits, as defined in the **/etc/security/limits** file. This file contains these default limits:

```
fsize = 2097151
core = 2097151
cpu = −1
data = 262144
rss = 65536
stack = 65536
nofiles = 2000
```

These values are used as default settings when a new user is added to the system. The values are set with the **mkuser** command when the user is added to the system, or changed with the **chuser** command.

Limits are categorized as either soft or hard. With the **ulimit** command, you can change your soft limits, up to the maximum set by the hard limits. You must have root user authority to change resource hard limits.

Many systems do not contain one or more of these limits. The limit for a specified resource is set when the *Limit* parameter is specified. The value of the *Limit* parameter can be a number in the unit specified with each resource, or the value `unlimited`. To set the specific ulimit to `unlimited`, use the word `unlimited`

> **Note:** Setting the default limits in the **/etc/security/limits** file sets system wide limits, not just limits taken on by a user when that user is created.

The current resource limit is printed when you omit the *Limit* parameter. The soft limit is printed unless you specify the **−H** flag. When you specify more than one resource, the limit name and unit is printed before the value. If no option is given, the **−f** flag is assumed.

Since the **ulimit** command affects the current shell environment, it is provided as a shell regular built−in command. If this command is called in a separate command execution environment, it does not affect the file size limit of the caller's environment. This would be the case in the following examples:

```
nohup ulimit −f 10000
env ulimit 10000
```

Once a hard limit has been decreased by a process, it cannot be increased without root privilege, even to revert to the original limit.

For more information about user and system resource limits, refer to the **getrlimit**, **setrlimit**, or **vlimit** subroutine in *AIX Version 4.3 Technical Reference: Base Operating System and Extensions Volume 1*.

Flags

−a Lists all of the current resource limits.
−c Specifies the size of core dumps, in number of 512−byte blocks.
−d Specifies the size of the data area, in number of K bytes.
−f Sets the file size limit in blocks when the *Limit* parameter is used, or reports the file size limit if no parameter is specified. The **−f** flag is the default.
−H Specifies that the hard limit for the given resource is set. If you have root user authority, you can increase the hard limit. Anyone can decrease it.
−m Specifies the size of physical memory, in number of K bytes.
−n Specifies the limit on the number of file descriptors a process may have.
−s Specifies the stack size, in number of K bytes.
−S Specifies that the soft limit for the given resource is set. A soft limit can be increased up to the value of the hard limit. If neither the **−H** nor **−S** flags are specified, the limit applies to both.
−t Specifies the number of seconds to be used by each process.

Exit Status

The following exit values are returned:

0 Successful completion.
>0 A request for a higher limit was rejected or an error occurred.

Example

To set the file size limit to 51,200 bytes, enter:

```
ulimit −f 100
```

Files

/usr/bin/ksh Contains the **ulimit** built−in command.

Related Information

The **ksh** command.

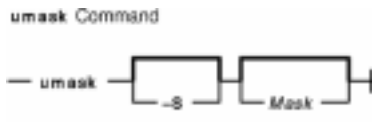
The **ulimit** subroutine, **getrlimit**, **setrlimit**, or **vlimit** subroutine in *AIX Version 4.3 Technical Reference: Base Operating System and Extensions Volume 1*.

umask Command

Purpose

Displays or sets the file mode creation mask.

Syntax



umask [**−S**] [*Mask*]

Description

If the *Mask* parameter is not specified, the **umask** command displays to standard output the file mode creation mask of the current shell environment. If you specify the *Mask* parameter using a three−digit octal number or symbolic code, the **umask** command sets the file creation mask of the current shell execution environment. The bits set in the file creation mask are used to clear the corresponding bits requested by an application or command when creating a file.

The **chmod** command describes how to use the symbolic and numeric codes to set permissions.

The **−S** flag produces symbolic output. If the flag is not specified, the default output format is octal.

If the **/usr/bin/umask** command is called in a subshell or separate command execution environment, it does not affect the file mode creation mask of the caller's environment. This would be the case in the following example:

```
(umask 002)

nohup umask ...

find . −exec umask ... \;
```

Flags

−S Produces symbolic output.

Exit Status

The following exit values are returned:

0 The file mode creation mask was successfully changed, or no *Mask* parameter was supplied.
>0 An error occurred.

Examples

1. To set the mode mask so that subsequently created files have their **S_IWOTH** bit cleared, enter either:

   ```
   umask a=rx,ug+w
   ```

   OR

   ```
   umask 002
   ```

   After setting the mode mask, display the current values of the mode mask by entering:

   ```
   umask
   ```

   The screen displays the following value:

   ```
   02
   ```

2. To produce symbolic output, enter:

   ```
   umask -S
   ```

   The screen displays the following values:

   ```
   u=rwx,g=rwx,o=rx
   ```

3. Either numeric or symbol output can be used as the *Mask* parameter to a subsequent invocation of the **umask** command. Assume the mode mask is set as shown in example 2. To set the mode mask so that subsequently created files have their **S_IWGRP** and **S_IWOTH** bits cleared, enter:

   ```
   umask g-w
   ```

4. To set the mode mask so that subsequently created files have all their write bits cleared, enter:

   ```
   umask -- -w
   ```

   > **Note:** The **-r**, **-w**, and **-x***Mask* parameter values (or anything beginning with a hyphen) must be preceded by -- (double hyphen, no space between) to keep it from being interpreted as an option.

Files

/usr/bin/ksh Contains the Korn shell **umask** built-in command.
/usr/bin/umask Contains the **umask** command.

Related Information

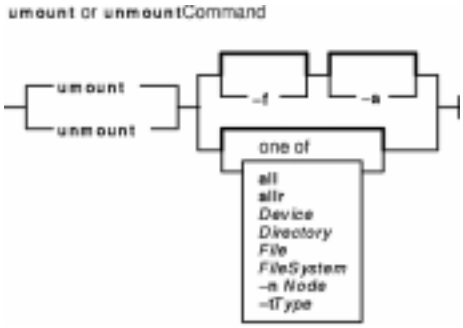
The **bsh** command, **chmod** command, **csh** command, **ksh** command.

umount or unmount Command

Purpose

Unmounts a previously mounted file system, directory, or file.

Syntax



{ **unmount** | **umount** }
[**−f**] [**−a**] | [**all** | **allr** | *Device* | *Directory* | *File* | *FileSystem* | **−n** *Node* | **−t***Type*]

Description

Another name for the **umount** command is the **unmount** command. Either name can be used. You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit umount
```

The **umount** command unmounts a previously mounted device, directory, file, or file system. Processing on the file system, directory, or file completes and it is unmounted. Members of the system group and users operating with root user authority can issue any **umount** command. Only users with root authority or are members of the system group can unmount a directory or file.

> **Note:** SMIT will not unmount the **/usr/lpp/info/$LANG** directory, the directory on which SMIT helps are located. Typically, this is the CD−ROM.

To unmount local mounts you can specify the device, directory, file, or file system on which it is mounted.

Flags

−a Unmounts all mounted file systems.

all Unmounts all mounted file systems.

allr Unmounts all remotely mounted file systems.

> **Note:** For remote mounts, specify the device, directory, file, or file system parameters. If you specify the **allr** flag, the **umount** command unmounts all remote mounts.

−f Forces an unmount in a remote environment. Use to free a client when the server is down and server path names cannot be resolved. The **−f** flag is not supported for journaled file systems.

> **Note:** This forced unmount works only when there is no hold on the directory, that is, when the directory is not some user's working directory and there is no process

started and still running there.

−n *Node* Specifies the node holding the mounted directory you want to unmount. The **umount −n***Node* command unmounts all remote mounts made from the *Node* parameter.

−t *Type* Unmounts all stanzas in the **/etc/filesystems** file that contain the **type=***Type* flag and are mounted. The *Type* parameter is a string value, such as the remote value that specifies the name of the group.

Note: You cannot use the **umount** command on a device in use. A device is in use if any file is open for any reason or if a user's current directory is on that device.

Examples

1. To unmount all mounts from remote node `Node A`, enter:

   ```
   umount −n nodeA
   ```

2. To unmount files and directories of a specific type, enter:

   ```
   umount −t test
   ```

This unmounts all files or directories that have a stanza in the **/etc/filesystems** file that contains the **type=test** attribute.

Files

/etc/filesystems Lists the known file systems and defines their characteristics.

Related Information

The **mount** command, **fuser** command.

The **mount** subroutine, **umount** subroutine, **vmount** subroutine, **uvmount** subroutine, **mntctl** subroutine.

The System Management Interface Tool (SMIT): Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains the structure, main menus, and tasks that are done with SMIT.

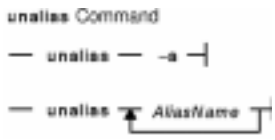
The Mounting Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains mounting files and directories, mount points, and automatic mounts.

unalias Command

Purpose

Removes alias definitions.

Syntax



unalias −a

unalias *AliasName ...*

Description

The **unalias** command removes the definition for each alias name specified, or removes all alias definitions if the **−a** flag is used. Alias definitions are removed from the current shell environment.

Since the **unalias** command affects the current shell execution environment, it is provided as a Korn shell or POSIX shell regular built−in command.

Flags

−a Removes all alias definitions from the current shell environment.

Exit Status

The following exit values are returned:

0 Successful completion.
>0 One of the alias names specified did not represent a valid alias definition, or an error occurred.

Files

/usr/bin/ksh Contains the Korn shell **unalias** built−in command.
/usr/bin/unalias Contains the **unalias** command.

Related Information

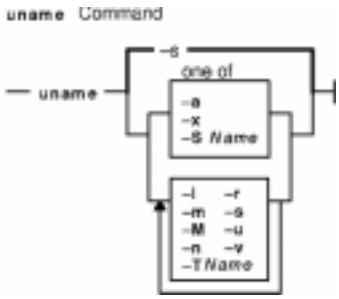
The **alias** command, **csh** command, **ksh** command.

uname Command

Purpose

Displays the name of the current operating system.

Syntax



uname [**−a** | **−x** | **−S***Name*] | [**−l**] [**−m**] [**−M**] [**−n**] [**−r**] [**−s**] [**−T***Name*] [**−u**] [**−v**]

Description

The **uname** command writes to standard output the name of the operating system that you are using.

The machine ID number contains 12 characters in the following digit format: *xxyyyyyymmss*. The *xx* positions indicate the system and is always 00. The *yyyyyy* positions contain the unique ID number for the entire system. The *mm* position represents the model ID. The *ss* position is the submodel number and is always 00. The model ID describes the ID of the CPU Planar, not the model of the System as a whole.

You can use the **uname −m** command sometimes to determine which model you are using. The following list is not complete. Refer to hardware vendor supplied documentation for values in the range E0 − FF. Also note that not all machine types have a machine ID. Many new machines share a common machine ID of 4C. Hexadecimal codes for the system models (*mm*) are:

Machine Type	Machine Model	Machine ID
7006	410	42
7007	N40	F0
7008	M20	43
7008	M20A	43
7009	C10	48

```
7011          220             41
7011          230             47
7011          250             46


7012          320             31
7012          320H            35
7012          340             37
7012          350          38 or 77
7012          355             77
7012          360             76
7012          365             76
7012          370             75
7012          375             75
7012          380             58
7012          390             57
7012          G30             A6
7012          G40             A7


7013          520             30
7013          520H            34
7013          530             10
7013          530H            18
7013          540          14 or 11
7013          550             1C
7013          550L            77
7013          560             5C
7013          570             67
7013          580             66
7013          58H             71
7013          590             70
7013          590H            72
7013          J30             A0
7013          J40             A1


7015          930          20 or 02
7015          950             2E
7015          970             63
7015          970B            63
7015          980             64
7015          980B            64
7015          990             80
7015          R10             67
7015          R20             72
7015          R24             81
7015          R30             A3
7015          R40             A4

7016          730             10


7018          740             30
7018          770             67

7024          E20             C0

7025          F30             C4
```

```
7030            3AT             58
7030            3BT             57


7043            140             4C
7043            240             4C

7428            43P             4C
```

The machine identifier value returned by the **uname** command may change when new operating system software levels are installed. This change affects applications using this value to access licensed programs. To view this identifier, enter the **uname −m** command.

Contact the appropriate support organization if your application is affected.

Flags

−a Displays all information specified with the **−m**, **−n**, **−r**, **−s**, and **−v** flags. Cannot be used with the **−x** or **−S***Name* flag. If the **−x** flag is specified with the **−a** flag, the **−x** flag overrides it.

−l Displays the LAN network number.

−m Displays the machine ID number of the hardware running the system.

−M Displays the system model name. If the model name attribute does not exist, a null string is displayed.

−n Displays the name of the node. This may be a name the system is known by to a UUCP communications network.

−r Displays the release number of the operating system.

−s Displays the system name. This flag is on by default.

−S *Name* Sets the name of the node. This can be the UUCP communications network name for the system.

−T *Name* Sets the system name. This can be the UUCP communications network name for the system.

−u Displays the system ID number. If this attribute is not defined, the output is the same as the output displayed by **uname−m**.

−v Displays the operating system version.

−x Displays the information specified with the **−a** flag as well as the LAN network number, as specified by the **−l** flag.

If you enter a flag that is not valid, the **uname** command exits with an error message, an error return status, and no output.

> **Note:** The uname command does not preserve the new system name and node name values across system reboot.

Exit Status

This command returns the following exit values:

0 The requested information was successfully written.

>0 An error occurred.

Example

To display the complete system name and version banner, enter:

```
uname -a
```

Files

/usr/bin/uname Contains the **uname** command.

Related Information

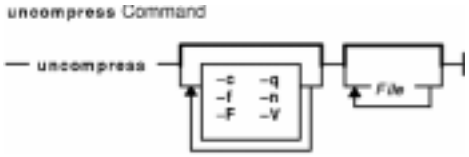
The **uname** or **unamex** subroutine.

uncompress Command

Purpose

Restores compressed files.

Syntax



uncompress [**−c**] [**−F**] [**−f**] [**−n**] [**−q**] [**−V**] [*File* ...]

Description

The **uncompress** command restores original files that were compressed by the **compress** command. Each compressed file specified by the *File* parameter is removed and replaced by an expanded copy. The expanded file has the same name as the compressed version, but without the **.Z** extension. If the user has root authority, the expanded file retains the same owner, group, modes, and modification time as the original file. If the user does not have root authority, the file retains the same modes and modification time, but acquires a new owner and group. If no files are specified, standard input is expanded to standard output.

Flags

−c	Write to standard output. No files are changed.
−f or **−F**	Forces expansion. The **−f** and **−F** flags are interchangeable. Overwrites the file if it already exists. The system does not prompt the user that an existing file will be overwritten. File size may not actually shrink.
−n	Omits the compressed file header from the compressed file.
−q	Suppresses the display of compression statistics generated by the **−v** flag. If several **−v** and **−q** flags are on the same command line, the last one specified controls the display of the statistics.
−V	Writes the current version and compile options to standard error.

Parameters

File ...	Specifies the compressed files to restore.

Return Values

The **uncompress** command detects an error and exit with a status of 1 if any of the following events occur:

- The input file was not produced by the **compress** command.
- An input file cannot be read or an output file cannot be written.

If no error occurs, the exit status is 0.

Exit Status

0 Successful completion.

>0 An error occurred.

Example

To uncompress the `foo.Z` file, enter:

```
uncompress foo.Z
```

The `foo.Z` file is uncompressed and renamed `foo`.

Related Information

The **compress** command uses the modified Lempel−Zev algorithm described in "A Technique for High Performance Data Compression," Welch, Terry A. *IEEE Computer*, vol. 17, no. 6 (June 1984), pp. 8−19.

The **compress** command, **pack** command, **unpack** command, **zcat** command.

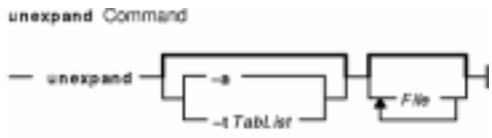
Commands Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

unexpand Command

Purpose

Writes to standard output with tabs restored.

Syntax



unexpand [**−a** | **−t** *TabList*] [*File* ...]

Description

The **unexpand** command puts tabs back into the data from the standard input or the named files and writes the result to standard output. By default, only leading spaces and tabs are reconverted to maximal strings of tabs.

Note: The *File* parameter must be a text file.

Flags

−a Inserts tabs wherever their presence compresses the resultant file by replacing two or more characters.

−t*TabList* Specifies the position of the tab stops. The default value of a tab stop is 8 column positions.

The *TabList* variable must consist of a single positive−decimal integer or multiple positive−decimal integers. The multiple integers must be in ascending order and must be separated by commas or by blank characters with quotation marks around the integers. The single *TabList* variable sets the tab stops an equal number of column positions apart. The multiple *TabList* variable sets the tab stop at column positions that correspond to the integers in the *TabList* variable.

A space−to−tab conversion does not occur for characters at positions beyond the last one specified in a multiple *TabList* variable.

Note: When the **−t** flag is specified, the **−a** flag is ignored and conversion is not limited to processing leading blank characters.

Exit Status

This command returns the following exit values:

0 The command ran successfully.

>0 An error occurred.

Example

To replace space characters with tab characters in the **xyz** file, enter:

```
unexpand xyz
```

Files

/usr/bin/unexpand Contains the **unexpand** command.

Related Information

The **expand** command, **newform** command, **tab** command, **untab** command.

Files Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

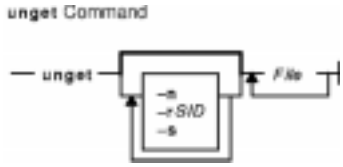
Input and Output Redirection Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

unget Command (SCCS)

Purpose

Cancels a previous **get** command.

Syntax



unget [−**r***SID*] [−**s**] [−**n**] *File* ...

Description

The **unget** command allows you to restore a g−file created with **get −e** before the new delta is created. Any changes are therefore discarded. If you specify a − (dash) for the value of *File*, standard input is read, and each line of standard input is interpreted as the name of an SCCS file. An end−of−file character terminates input.

If you specify a directory for the *File* value, the **unget** command performs the requested actions on all SCCS files that are currently in the process of being edited (those files with the **s.** prefix).

Once you have run an **unget** command on a file, you must reissue a **get −e** command to make changes to the file. The **unget** command automatically deletes the g−file.

Flags

Each flag or group of flags applies independently to each named file.

−n Prevents the automatic deletion of the g−file. This flag allows you to retain the edited version of the file without making a delta.

−**r***SID* Specifies the new delta that would have been created by the next use of the **delta** command. You must use this flag if you have two or more pending deltas to the file under the same login name. You can look at the p−file to see if you have more than one delta pending to a particular SID under the same login name. The *SID* specification must unambiguously specify only one SID to discard, or the **unget** command displays an error message and stops running.

−s Suppresses displaying the deleted SID.

Exit Status

This command returns the following exit values:

0 Successful completion.

>0 An error occurred.

Example

To discard the changes you have made to an SCCS file after running a **get −e** command, enter:

```
unget s.prog.c
```

Files

/usr/bin/unget Contains the path to the SCCS **unget** command.

Related Information

The **delta** command, **get** command, **sact** command.

The **sccsfile** file format.

List of SCCS Commands in *AIX General Programming Concepts: Writing and Debugging Programs*.

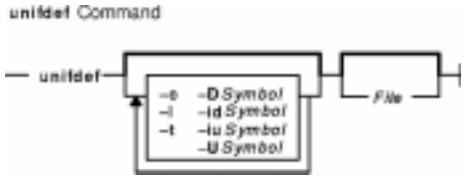
Source Code Control System (SCCS) Overview in *AIX General Programming Concepts: Writing and Debugging Programs*.

unifdef Command

Purpose

Removes ifdef lines from a file.

Syntax



unifdef [**−t**] [**−l**] [**−c**] [**−D***Symbol*] [**−U***Symbo*l] [**−id***Symbol*] [**−iu***Symbol*] [*File*]

Description

The **unifdef** command is useful for removing ifdef lines from a file while otherwise leaving the file alone. The **unifdef** command recognizes nested ifdefs, comments, and single and double quotes of C syntax in order to function correctly, but does not include files or interpret macros. The **unifdef** command recognizes but does not remove comments.

The **unifdef** command takes its input from standard input if no *File* is specified and copies its output to standard output.

Once a *Symbol* is specified, the lines inside those ifdefs are copied to the output or removed, as appropriate. The ifdef, ifndef, else, elif, and endif lines associated with the symbol are also removed. Ifdefs that involve unspecified symbols are untouched and copied out along with their associated ifdef, else, elif, and endif lines. If the same symbol appears in more than one argument, only the first occurrence is significant. For instance, if an ifdef X occurs nested inside another ifdef X, the inside ifdef is considered an unrecognized symbol.

When using ifdefs to delimit non−C lines such as comments or unfinished code, it is necessary to specify which symbols are to be used for that purpose. Otherwise, the **unifdef** command will try to parse for quotes and comments in those ifdef lines.

The **unifdef** command cannot process **cpp** constructs such as:

```
#if defined(X) || defined(Y)
```

OR

```
#elif X
```

OR

```
#elif defined(X) || defined(Y)
```

Keywords

The following keywords are recognized by the **unifdef** command:

- **ifdef**
- **ifndef**
- **else**
- **endif**
- **elif**

Flags

−c	Complements the operation of the **unifdef** command. That is, the lines which would have been removed are retained and vice versa.
−D *Symbol*	Specifies the symbol to be defined.
File	Specifies the input source.
−id *Symbol*	The **unifdef** command will not try to recognize comments, single quotes, or double quotes inside specified **ifdefs**, but these lines will be copied out.
−iu *Symbol*	The **unifdef** command will not try to recognize comments, single quotes, or double quotes inside specified **ifdefs**. These lines will not be copied out.
−l	Causes removed lines to be replaced with blank lines instead of being deleted.
−t	Allows the **unifdef** command to be used for plain text (instead of C code): the **unifdef** command will not try to recognize comments, single quotes and double quotes.
−U *Symbol*	Specifies the symbol to be undefined.

Exit Status

This command returns the following exit values:

0 The output is an exact copy of the input.

1 The output is not an exact copy of the input.

2 The command failed due to a premature EOF, or to an inappropriate **else**, **elif**, or **endif**.

Examples

1. The following example:

   ```
   unifdef −DA original.c > modified.c
   ```

   causes the **unifdef** command to read the file `original.c`, and remove the #ifdef A lines. It then removes everything following an #elif/#else associated with the #ifdef A, down to the #endif. The output is placed in the `modified.c` file.

2. The following example:

   ```
   unifdef −UA original.c > modified.c
   ```

causes the **unifdef** command to read the file `original.c`, and remove the #ifdef A down to either its associated #elif//#else, or its associated #endif. In the case of the #elif, the #elif is replaced with #if. In the case of #else, the #else is deleted along with its associated #endif. The output is placed in the `modified.c` file.

Files

/usr/bin/unifdef Contains the **unifdef** command.

Related Information

The **cpp** command.

The Commands Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

uniq Command

Purpose

Reports or deletes repeated lines in a file.

Syntax

uniq [**−c** | **−d** | **−u**] [**−f***Fields*] [**−s** *Characters*] [*−Fields*] [*+Characters*]
[*InFile* [*OutFile*]]

Description

The **uniq** command deletes repeated lines in a file. The **uniq** command reads either standard input or a file specified by the *InFile* parameter. The command first compares adjacent lines and then removes the second and succeeding duplications of a line. Duplicated lines must be adjacent. (Before issuing the **uniq** command, use the **sort** command to make all duplicate lines adjacent.) Finally, the **uniq** command writes the resultant unique lines either to standard output or to the file specified by the *OutFile* parameter. The *InFile* and *OutFile* parameters must specify different files.

The input file must be a text file. A *text* file is a file that contains characters organized into one or more lines. The lines can neither exceed 2048 bytes in length (including any newline characters) nor contain null characters.

The **uniq** command compares entire lines by default. If the **−f** *Fields* or *−Fields* flag is specified, the **uniq** command ignores the number of fields specified by the *Fields* variable. A *field* is a string of characters separated from other character strings by one or more <blank> characters. If the **−s***Characters* or *−Characters* flag is specified, the **uniq** command ignores the number of characters specified by the *Characters* variable. Values specified for the *Fields* and *Characters* variables must be positive decimal integers.

The current national language environment determines the <blank> characters used by the **−f** flag as well as how the **−s** flag interprets bytes as a character.

The **uniq** command exits with a value of 0 if successful. Otherwise, it exits with a value greater than 0.

Flags

−c	Precedes each output line with a count of the number of times each line appeared in the input file.
−d	Displays only the repeated lines.
−f *Fields*	Ignores the number of fields specified by the *Fields* variable. If the value of the *Fields* variable exceeds the number of fields on a line of input, the **uniq** command uses a null string for comparison. This flag is equivalent to the *−Fields* flag.
−u	Displays only the unrepeated lines.

−s *Characters* Ignores the number of characters specified by the *Characters* variable. If the value of the *Characters* variable exceeds the number of characters on a line of input, the **uniq** command uses a null string for comparison. If both the **−f** and **−s** flags are specified, the **uniq** command ignores the number of characters specified by the **−s***Characters* flag starting in the field following the fields specified by the **−f***Fields* flag. This flag is equivalent to the +*Characters* flag.

−*Fields* Ignores the number of fields specified by the *Fields* variable. This flag is equivalent to the **−f** *Fields* flag.

+*Characters* Ignores the number of characters specified by the *Characters* variable. If both the −*Fields* and +*Characters* flags are specified, the **uniq** command ignores the number of characters specified by the +*Characters* flag starting in the field following the fields specified by the −*Fields* flag. This flag is equivalent to the **−s** *Characters* flag.

Exit Status

This command returns the following exit values:

0 The command ran successfully.

>0 An error occurred.

Example

To delete repeated lines in a file named `fruit` and save it to a file named `newfruit,` enter:

```
uniq fruit newfruit
```

If the `fruit` file contains the following lines:

```
apples
apples
peaches
pears
bananas
cherries
cherries
```

then the `newfruit` file will contain the following lines after you run the **uniq** command:

```
apples
peaches
pears
bananas
cherries
```

Files

/usr/bin/uniq Contains the **uniq** command.

Related Information

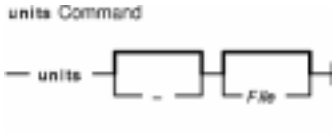
The **comm** command, **sort** command.

units Command

Purpose

Converts units in one measure to equivalent units in another measure.

Syntax



units [−] [*File*]

Description

The **units** command converts quantities expressed in one measurement to their equivalents in another. The **units** command is an interactive command. It prompts you for the unit you want to convert *from* and the unit you want to convert *to*. This command only does multiplicative scale changes. That is, it can convert from one value to another only when the conversion involves a multiplication. For example, it cannot convert between degrees Fahrenheit and degrees Celsius because the value of 32 must be added or subtracted in the conversion.

You can specify a quantity as a multiplicative combination of units, optionally preceded by a numeric multiplier.

Indicate powers by entering suffixed positive integers, and indicate division with a / (slash).

The **units** command recognizes `lb` as a unit of mass, but considers `pound` to be the British pound sterling. Compound names are run together (such as `lightyear`). Prefix British units differing from their American counterparts with `br` (`brgallon`, for instance).

The **/usr/share/lib/unittab** file contains a complete list of the units that the **units** command uses. You can also define new units in this file. The *File* parameter may be used to override the values of the standard conversion factors listed in the **/usr/share/lib/unittab** file. The specified file must follow the same format as the **unittab** file.

Most familiar units, abbreviations, and metric prefixes are recognized by the **units** command, as well as the following:

pi	Ratio of circumference to diameter
c	Speed of light
e	Charge on an electron
g	Acceleration of gravity
force	Same as **g**
mole	Avogadro's number
water	Pressure head per unit height of water
au	Astronomical unit

Flags

− Lists the conversion factors contained in the **/usr/share/lib/unittab** file before you are prompted to enter your conversion.

Examples

1. To display conversion factors for inches to centimeters, enter:

   ```
   units
   you have: in
   you want: cm
   ```

   The **units** command returns the following values:

   ```
   * 2.540000e+00
   / 3.937008e−01
   ```

   The output tells you to multiply the number of inches by `2.540000e+00` to get centimeters, and to multiply the number of centimeters by `3.937008e−01` to get inches.

   These numbers are in standard exponential notation, so `3.937008e−01` means `3.937008 x 10−1`, which is the same as `0.3937008`.

   > **Note:** The second number is always the reciprocal of the first; for example, `2.54` equals `1/0.3937008`.

2. To convert a measurement to different units, enter:

   ```
   units
   you have: 5 years
   you want: microsec
   ```

   The **units** command returns the following values:

   ```
   * 1.577846e+14
   / 6.337753e−15
   ```

   The output shows that `5 years` equals `1.577846 x 1014` microseconds, and that one microsecond equals `6.337753 x 10−15` years.

3. To give fractions in measurements, enter:

   ```
   units
   you have: 1|3 mi
   you want: km
   ```

   The **units** command returns the following values:

   ```
   * 5.364480e−01
   / 1.864114e+00
   ```

   The | (vertical bar) indicates division, so `1|3` means one−third. This shows that one−third mile is the same as `0.536448` kilometers.

4. To include exponents in measurements, enter:

   ```
   units
   ```

```
you have: 1.2-5 gal
you want: floz
```

The **units** command returns the following values:

```
* 1.536000e-03
/ 6.510417e+02
```

The expression `1.2-5 gal` is the equivalent of `1.2 x 10-5`. Do *not* type an `e` before the exponent (that is, `1.2e-5 gal` is not valid). This example shows that `1.2 x 10-5 (0.000012)` gallons equal `1.536 x 10-3 (0.001536)` fluid ounces.

5. To specify complex units, enter:

```
units
you have: gram centimeter/second2
you want: kg-m/sec2
```

The **units** command returns the following values:

```
* 1.000000e-05
/ 1.000000e+05
```

The units `gram centimeter/second2` mean "grams x centimeters/second2." Similarly, `kg-m/sec2` means "kilograms x meters/sec2," which is often read as "kilogram-meters per seconds squared."

6. If the units you specify after `you have:` and `you want:` are incompatible:

```
you have: ft
you want: lb
```

The **units** command returns the following message and values:

```
conformability
3.048000e-01 m
4.535924e-01 kg
```

The `conformability` message means the units you specified cannot be converted. Feet measure length, and pounds measure mass, so converting from one to the other does not make sense. Therefore, the **units** command displays the equivalent of each value in standard units.

In other words, this example shows that one foot equals `0.3048` meters and that one pound equals `0.4535924` kilograms. The **units** command shows the equivalents in meters and kilograms because the command considers these units to be the standard measures of length and mass.

Files

/usr/bin/units	Contains the **units** command.
/usr/share/lib/unittab	Lists units that the **units** command creates as well as units defined by the user.

Related Information

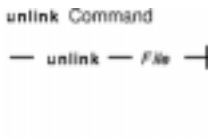
The **bc** command, **dc** command.

unlink Command

Purpose

Performs an **unlink** subroutine.

Syntax



unlink*File*

Description

The **unlink** command performs the **unlink** subroutine on a specified file.

The **unlink** command does not issue error messages when the associated subroutine is unsuccessful; you must check the exit value to determine if the command completed normally. It returns a value of 0 if it succeeds, a value of 1 if too few or too many parameters are specified, and a value of 2 if its system call is unsuccessful.

> **Attention:** The **unlink** command allows a user with root user authority to deal with unusual problems, such as moving an entire directory to a different part of the directory tree. It also permits you to create directories that cannot be reached or escaped from. Be careful to preserve the directory structure by observing the following rules:

- Be certain every directory has a . (dot) link to itself.
- Be certain every directory has a .. (dot dot) link to its parent directory.
- Be certain every directory has no more than one link to itself or its parent directory.
- Be certain every directory is accessible from the root of its file system.

Example

To remove a directory entry pointed by `file2`, enter:

```
unlink file2
```

Files

/usr/sbin/unlink Contains the **unlink** command.

Related Information

The **fsck** command, **link** command, **ln** command.

The **link** subroutine, **unlink** subroutine.

The File Systems Overview for System Management in *AIX Version 4.3 System Management Concepts:*

Operating System and Devices explains file system types, management, structure, and maintenance.

The Files Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* provides information on working with files.

The Directory Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* explains working with directories and path names.

unloadipsec Command

Purpose

Unloads a crypto module from the IP Security subsystem.

Syntax

unloadipsec **−c***crypto_mod_name*

Description

The **unloadipsec** command unloads a crypto module from the IP Security subsystem. The **unloadipsec** command can be used when a crypto module is no longer being used or when a crypto module is to be replaced with a newer version.

A crypto module can only be unloaded after the IP Security device is stopped. The steps for replacing a crypto module are: change the IP Security device to the defined state; unload the old crypto module using this command; uninstall the old module and install the new module, and bring the IP Security device back to the available state.

Flags

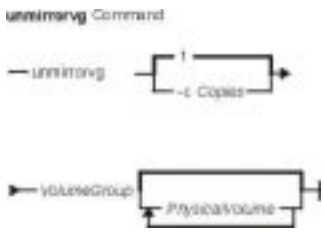
−c*crypto_mod_name* Specifies the name of the crypto module to be unloaded. When used without any flag, the command lists all the crypto modules installed (but not necessarily loaded).

unmirrorvg Command

Purpose

Removes the mirrors that exist on volume groups or specified disks. This command only applies to AIX Version 4.2.1 or later.

Syntax



unmirrorvg [**−c** *Copies*] *VolumeGroup* [*PhysicalVolume .. .*]

Description

The **unmirrorvg** command unmirrors all the logical volumes detected on a given volume group. This same functionality may also be accomplished manually if you execute the **rmlvcopy** command for each individual logical volume in a volume group.

By default, **unmirrorvg** will pick the set of mirrors to remove from a mirrored volume group. If you wish to control which drives no longer are to contain mirrors, you must include the list of disks in the input parameters, *PhysicalVolume.*

When the *PhysicalVolume* parameter is listed in the command, this indicates that only the mirrors that exist on this disk should be unmirrored. Mirrors that exist on other drives in the volume group, but not listed in a user−provided disk list are left alone and remain mirrored.

> **Note:** If a logical volume copy spans more than one disk, the portion of the logical volume copy that resides on a disk not listed by the user is also removed.

When **unmirrorvg** is executed, the default `COPIES` value for each logical volume becomes 1. If you wish to convert your volume group from triply mirrored to doubly mirrored, use the **−c** option.

> **Note:** To use this command, you must either have root user authority or be a member of the **system** group.
>
> **Attention:** The **unmirrorvg** command may take a significant amount of time to complete because of complex error checking and the number of logical volumes to unmirror in a volume group.

Y ou can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit unmirrorvg
```

Flag

−c *Copies* Specifies the minimum number of copies that each logical volume must have after the **unmirrorvg** command has finished executing. If you do not want all logical volumes to have the same number of copies, then reduce the mirrors manually with the **rmlvcopy** command. If this option is not used, the copies will default to 1.

Examples

1. To unmirror a triply mirrored volume group and leave two copies, enter:

   ```
   unmirrorvg −c 2 workvg
   ```

   The logical partitions in the logical volumes held on `workvg` now have 2 copies.

2. To get default unmirroring of rootvg, enter:

   ```
   unmirrorvg rootvg
   ```

   `rootvg` now has only 1 copy.

3. To replace a bad disk drive in a mirrored volume group, enter:

   ```
   unmirrorvg workvg hdisk7
   reducevg workvg hdisk7
   rmdev −l hdisk7 −d
   replace the disk drive, let the drive be renamed hdisk7
   extendvg workvg hdisk7
   mirrorvg workvg
   ```

   Note: By default in this example, **mirrorvg** will try to create 2 copies for logical volumes in `workvg`. It will try to create the new mirrors onto the replaced disk drive. However, if the original system had been triply mirrored, there may be no new mirrors created onto `hdisk7`, as other copies may already exist for the logical volumes. This follows the default behavior of **unmirrorvg** to reduce the mirror copy count to 1.

Implementation Specifics

Software Product/Option: Base Operating System/ AIX 3.2 to 4.1 Compatibility Links
Standards Compliance: NONE

Files

/usr/sbin Directory where the **unmirrorvg** command resides.

Related Information

The **mklvcopy** command, **mirrorvg** command, **syncvg** command, **reducevg** command, **extendvg** command.

The Logical Volume Storage Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

unpack Command

Purpose

Expands files.

Syntax

unpack *File* ...

Description

The **unpack** command expands files created by the **pack** command. For each file specified, the **unpack** command searches for a file called *File***.z**. If this file is a packed file, the **unpack** command replaces it by its expanded version. The **unpack** command names the new file name by removing the **.z** suffix from *File*. If the user has root authority, the new file has the same access modes, access and modification times, owner, and group as the original file. If the user does not have root authority, the file retains the same access modes, access time, and modification time, but acquires a new owner and group.

The **unpack** command operates only on files ending in **.z**. As a result, when you specify a file name that does not end in **.z**, the **unpack** command adds the suffix and searches the directory for a file name with that suffix.

The exit value is the number of files the **unpack** command was unable to unpack. A file cannot be unpacked if any of the following occurs:

- The file name (exclusive of **.z**) has more than 253 bytes.
- The file cannot be opened.
- The file is not a packed file.
- A file with the unpacked file name already exists.
- The unpacked file cannot be created.

Note: The **unpack** command writes a warning to standard error if the file it is unpacking has links. The new unpacked file has a different i−node than the packed file from which it was created. However, any other files linked to the original i−node of the packed file still exist and are still packed.

Exit Status

This command returns the following exit values:

0 The command ran successfully.

>0 An error occurred.

Example

To unpack packed files:

```
unpack chap1.z chap2
```

This expands the packed files `chap1.z` and `chap2.z`, and replaces them with files named `chap1` and `chap2`. Note that you can give the **unpack** command file names either with or without the **.z** suffix.

Files

/usr/bin/unpack Contains the **unpack** command.

Related Information

The **cat** command, **compress** command, **pack** command.

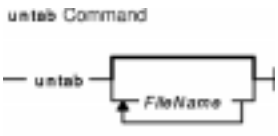
Files Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

untab Command

Purpose

Changes tabs into spaces.

Syntax



untab [*FileName* ...]

Description

The **untab** command reads the file specified by the *FileName* parameter or standard input, and replaces tabs in the input with space characters. If you specify a file with the *FileName* parameter, the **untab** command writes the resulting file back to the original file. If the input is standard input, the **untab** command writes to standard output. The **untab** command assumes that tab stops are set every eight columns, starting with column nine. The file name specified for the *FileName* parameter cannot exceed **PATH_MAX**−9 bytes in length.

Example

To replace tab characters in the `File` file with space characters, enter:

```
untab File
```

Files

/usr/bin/untab Contains the **untab** command.

Related Information

The **expand** command, **newform** command, **tab** command, **unexpand** command.

Files Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

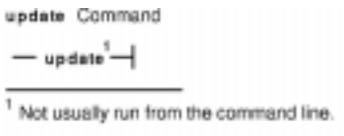
Input and Output Redirection Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

update Command

Purpose

Periodically updates the super block.

Syntax



update

Description

The **update** command executes a **sync** subroutine every 30 seconds. This action ensures the file system is up−to−date in the event of a system crash.

Files

/usr/sbin/update Contains the **update** command.

Related Information

The **init** command, **rc** command, **sync** command.

The **cron** daemon.

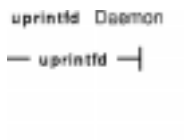
The **sync** subroutine.

uprintfd Daemon

Purpose

Constructs and writes kernel messages.

Syntax



uprintfd

Description

The **uprintfd** daemon retrieves, converts, formats, and writes kernel messages to processes' controlling terminals. Kernel messages are submitted through the **NLuprintf** and **uprintf** kernel services. Because the **uprintfd** daemon never exits, it should be run only once.

Related Information

The **NLuprintf** kernel service and **uprintf** kernel service.

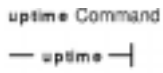
The Input and Output Handling Programmer's Overview in *AIX General Programming Concepts: Writing and Debugging Programs*.

uptime Command

Purpose

Shows how long the system has been up.

Syntax



uptime

Description

The **uptime** command prints the current time, the length of time the system has been up, the number of users online, and the load average. The load average is the number of runnable processes over the preceding 5−, 10−, 15−minute intervals. The output of the **uptime** command is, essentially, the heading line provided by the **w** command.

Related Information

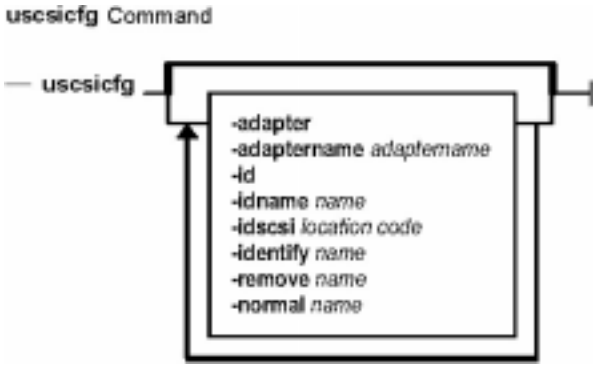
The **ruptime** command, **w** command.

uscsicfg Command

Purpose

List SCSI devices attached to an adapter.

Syntax



uscsicfg [**−adapter**] [**−adaptername***adaptername*] [**−id**] [**−idname***name*] [**−idscsi***location code*] [**−identify***name*] [**−remove***name*] [**−normal***name*]

Description

The **uscsicfg** command allows the user to list all SCSI devices attached to a particular adapter by providing the *name* or *location code* of the SCSI device or the name of the adapter to which it is attached.

This command also allows the user to put a disk that is connected to an SES backplane into the identify, remove, or normal state. To put the disk into one of these states, the user needs to provide the state and the *name*, or *location code* of the device.

Flags

−adapter	Lists the names of all the SCSI adapters on the system.
−adaptername*adaptername*	Lists the devices attached to *adaptername* where *adaptername* is the name of a SCSI adapter (for example, `scsi1`).
−id	Lists all the SCSI devices that are not adapters on the system.
−idname*name*	Lists all the SCSI devices that are siblings of *name* where *name* is the name of a SCSI device that is not an adapter (for example, `hdisk0`).
−identify*name*	Sets the *name* in identify mode where *name* is a disk attached to a backplane (for example, `hdisk0`).
−idscsi*location code*	Lists all the SCSI devices that have the same parent as the device for which the *location code* was provided. The device is not an adapter.
−normal*name*	Sets the *name* to normal from identify, or remove where *name* is a disk attached to a backplane (for example, `hdisk0`).
−remove*name*	Sets the *name* in remove mode where *name* is a disk attached to a backplane (for example, `hdisk0`).

Examples

1. **/usr/lpp/diagnostics/bin/uscsicfg** −adaptername scsi0

```
backplane    ses0 10-60-00-15,0        bank      slot      box      power
    hdisk0            10-60-00-8,0       U        1     SYSTEM      on
cd0               10-60-00-3,0
```

2. **/usr/lpp/diagnostics/bin/uscsicfg** −idname cd0

```
backplane    ses0 10-60-00-15,0        bank      slot      box      power
    hdisk0            10-60-00-8,0       U        1     SYSTEM      on
cd0               10-60-00-3,0
```

3. **/usr/lpp/diagnostics/bin/uscsicfg** −identify hdisk0
4. **/usr/lpp/diagnostics/bin/uscsicfg** −identify hdisk0

Files

/usr/lpp/diagnostics/bin Contains the uscsicfg command.

Related Information

The **diag** command.

users Command

Purpose

Displays a compact list of the users currently on the system.

Syntax

users [*File*]

Description

The **users** command lists the login names of the users currently on the system to standard output (**stdout**) in a compact, one−line list format. If an argument is given, it is used as an alternate file instead of **/etc/utmp**.

Files

/etc/utmp	Contains list of current users.
/usr/bin/users	Contains the **users** command.

Related Information

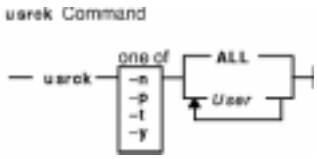
The **who** command.

usrck Command

Purpose

Verifies the correctness of a user definition.

Syntax



usrck { **−n** | **−p** | **−t** | **−y** } { **ALL** | *User* ... }

Description

The **usrck** command verifies the correctness of the user definitions in the user database files, by checking the definitions for **ALL** the users or for the users specified by the *User* parameter. If more than one user is specified, there must be a space between the names. You must select a flag to indicate whether the system should try to fix erroneous attributes.

The command first checks the entries in the **/etc/passwd** file. If you indicate that the system should fix errors, duplicate user names are reported and removed. Duplicate IDs are reported only, since there is no system fix. If an entry has fewer than six colon−separated fields, the entry is reported, but not fixed. The **usrck** command next checks specific user attributes in other files.

The **usrck** command verifies that each user name listed in the **/etc/passwd** file has a stanza in the **/etc/security/user**, **/etc/security/limits** and **/etc/security/passwd** files. The **usrck** command also verifies that each group name listed in the **/etc/group** file has a stanza in the **/etc/security/group** file. The **usrck** command using the **−y** flag creates stanzas in the security files for the missing user and group names.

> **Note:** This command writes its messages to **stderr**.

A list of all the user attributes follows, with notations stating which attributes are checked:

account_locked	No check. The **usrck** command sets this attribute to True and disables accounts.
admgroups	Checks to see if the **admgroups** are defined in the user database and, if you indicate that the system should fix errors, the command removes any groups that are not in the database.
auditclasses	Checks to see if the **auditclasses** are defined for the user in the **/etc/security/audit/config** file. If you indicate that the system should fix errors, the command deletes all the auditclasses that are not defined in the **/etc/security/audit/config** file.
auth1	Checks the primary authentication method. Unless the method is NONE or SYSTEM, it must be defined in the **/etc/security/login.cfg** file and the program attribute must exist and be executable by the root user. If you indicate that the system should fix errors, it will disable the user account if an error is found.
auth2	Checks the secondary authentication method. Unless the method is NONE or SYSTEM, it must be defined in the **/etc/security/login.cfg** file and the program attribute must exist and be executable by the root user. There is no system fix.

core Ensures that the values are sensible. If not, the command resets the values to 200 blocks, the minimum value.

core_hard Ensures that the values are sensible. If not, the command resets the values to 200 blocks, the minimum value. This attribute applies to AIX Version 4.2 or later.

cpu Ensures that the values are sensible. If not, the command resets the values to 120 seconds, the minimum value.

cpu_hard Ensures that the values are sensible. If not, the command resets the values to 120 seconds, the minimum value. This attribute applies to AIX Version 4.2 or later.

data Ensures that the values are sensible. If not, the command resets the values to 128 blocks (64K) and for AIX Version 4.1.5 and later to 1272 blocks (636K), the minimum value.

data_hard Ensures that the values are sensible. If not, the command resets the values to 1 272 blocks (636K), the minimum value. This attribute applies to AIX Version 4.2 or later.

dictionlist Checks the list of dictionary files. If you indicate that the system should fix errors, all dictionary files that do not exist are deleted from the user database.

expires No check.

fsize Ensures that the values are sensible. If not, the command resets the values to 200 blocks, the minimum value.

fsize_hard Ensures that the values are sensible. If not, the command resets the values to 200 blocks, the minimum value. This attribute applies to AIX Version 4.2 or later.

gecos No check.

histexpire Ensures that the values are sensible. If you indicate that the system should fix errors, values that are too large are set to the largest possible value and values that are too small are set to the smallest possible value.

histsize Ensures that the values are sensible. If you indicate that the system should fix errors, values that are too large are set to the largest possible value and values that are too small are set to the smallest possible value.

home Checks the existence and accessibility of the home directory by read mode and search mode. If you indicate that the system should fix errors, it will disable the user account if an error is found.

id Checks the uniqueness of the user ID. If you indicate that the system should fix errors, the command deletes any invalid entry in the **/etc/passwd** file.

login No check.

loginretries Checks if the user attempted unsuccessful logins more than the allowable amount. If so, the system disables the user account.

logintimes Ensures that the string of time specifiers is valid. If you indicate that the system should fix errors, the system disables the user account if an error is found.

maxage Ensures that the values are sensible. If you indicate that the system should fix errors, values that are too large are set to the largest possible value and values that are too small are set to the smallest possible value.

maxexpired Ensures that the values are sensible. If you indicate that the system should fix errors, values that are too large are set to the largest possible value and values that are too small are set to the smallest possible value.

maxrepeats Ensures that the values are sensible. If you indicate that the system should fix errors, values that are too large are set to the largest possible value and values that are too small are set to the smallest possible value.

minage Ensures that the values are sensible. If you indicate that the system should fix errors, values that are too large are set to the largest possible value and values that are too small are set to the

smallest possible value. The system also indicates if the **minage** attribute is larger than the **maxage** attribute.

minalpha Ensures that the values are sensible. If you indicate that the system should fix errors, values that are too large are set to the largest possible value and values that are too small are set to the smallest possible value.

mindiff Ensures that the values are sensible. If you indicate that the system should fix errors, values that are too large are set to the largest possible value and values that are too small are set to the smallest possible value.

minlen Ensures that the values are sensible. If you indicate that the system should fix errors, values that are too large are set to the largest possible value and values that are too small are set to the smallest possible value.

minother Ensures that the values are sensible. If you indicate that the system should fix errors, values that are too large are set to the largest possible value and values that are too small are set to the smallest possible value. The system also indicates if the **minage** attribute plus the **maxage** attribute is greater than the maximum password size.

name Checks the uniqueness and composition of the user name. The name must be a unique string of eight bytes or less. It cannot begin with a + (plus sign), a : (colon), a − (minus sign), or a ~ (tilde). Names beginning with a + (plus sign) or with a − (minus sign) are assumed to be names in the NIS (Network Information Service) domain, and no further processing is performed. It cannot contain a colon (:) in the string and cannot be the **ALL** or **default** keywords. If you indicate that the system should fix errors, the command disables the user account if an error is found and deletes any invalid entry in the **/etc/passwd** file.

The **usrck** command verifies that, for each user name listed in the **/etc/passwd** file, there is a stanza in the **/etc/security/user**, **/etc/security/limits**, and **/etc/security/passwd** files. The command adds stanzas for each one identified as missing. The **usrck** command additionally verifies that each group name listed in the **/etc/group** file has a stanza in the **/etc/security/group** file.

nofiles Ensures that the value is sensible. If not, resets the value to 200, the minimum value.

nofiles_hard Ensures that the value is sensible. If not, resets the value to 200, the minimum value.

pgrp Checks for the existence of the primary group in the user database. If you indicate that the system should fix errors, it will disable the user account if an error is found.

pwdchecks Checks the list of external password restriction methods. If you indicate that the system should fix errors, all methods that do not exist are deleted from the user database.

pwdwarntime Ensures that the value is sensible. If not, the system resets the value to the difference between the **maxage** and **minage** values.

rlogin No check.

rss Checks to ensure that the values are sensible. If not, the command resets the values to 128 blocks (64K), the minimum value.

rss_hard Checks to ensure that the values are sensible. If not, the command resets the values to 128 blocks (64K), the minimum value. This attribute applies to AIX Version 4.2 or later.

shell Checks the existence and accessibility of the shell by execute mode. If you indicate that the system should fix errors, it will disable the user account if an error is found.

stack Checks to ensure that the values are sensible. If not, the command resets the values to 128 blocks (64K), the minimum value.

stack_hard Checks to ensure that the values are sensible. If not, the command resets the values to 128 blocks (64K), the minimum value. This attribute applies to AIX Version 4.2 or later.

su No check.

sugroups	Checks for the existence of the **sugroups** in the user database files. If you indicate that the system should fix errors, it will delete all the groups that are not in the database.
sysenv	No check.
tpath	Checks to ensure that the **shell** attribute is tagged as a trusted process if **tpath=always**. If you indicate that the system should fix errors, it will disable the user account if an error is found.
ttys	Checks for the existence of the ttys in the user database files. If you indicate that the system should fix errors, it will delete all the ttys that do not exist from the user database.
usrenv	No check.

If the fix involves disabling a user account, use the **chuser** command to reset the value of the **account_locked** attribute to False. You can use the System Management Interface Tool (SMIT) to run the **chuser** command by entering:

```
smit chuser
```

The root user or a member of the security group can enable a user account again by removing the **account_locked** attribute or setting the **account_locked** attribute to False. The root user's account is not disabled by the **usrck** command.

Generally, the **sysck** command calls the **usrck** command as part of the verification of a trusted−system installation. If the **usrck** command finds any errors in the user database, the root user or a member of the security group should execute both the **grpck** command and the **pwdck** command.

The **usrck** command checks to see if the database management security files (**/etc/passwd.nm.idx**, **/etc/passwd.id.idx**, **/etc/security/passwd.idx**, and **/etc/security/lastlog.idx**) files are up−to−date or newer than the corresponding system security files. Please note, it is alright for the **/etc/security/lastlog.idx** to be not newer than **/etc/security/lastlog**. If the database management security files are out−of−date, a warning message appears indicating that the root user should run the **mkpasswd** command.

The **usrck** command checks if the specified user can log in. If the user cannot log in because of too many unsuccessful login attempts or because the password is expired, the **usrck** command issues a warning message indicating why the user cannot log in. If you indicate that the system should fix errors, the system disables the user account if the user cannot log in for the above reasons.

Flags

−n Reports errors but does not fix them.

−p Fixes errors but does not report them.

−t Reports errors and asks if they should be fixed.

−y Fixes errors and reports them.

Security

Access Control: This command should grant execute (x) access to the root user and members of the security group. The command should be **setuid** to the root user and have the **trusted computing base** attribute.

Files Accessed:

Mode	File
r	**/etc/passwd**
r	**/etc/security/user**

rw	**/etc/security/group**
rw	**/etc/group**
rw	**/etc/security/lastlog**
rw	**/etc/security/limits**
rw	**/etc/security/audit/config**
rw	**/etc/security/login.cfg**

Auditing Events:

Event	Information
USER_Check	user, attribute−error, status

Examples

1. To verify that all the users exist in the user database, and have any errors reported (but not fixed), enter:

   ```
   usrck −n ALL
   ```

2. To delete from the user definitions those users who are not in the user database files, and have any errors reported, enter:

   ```
   usrck −y ALL
   ```

Files

/usr/bin/usrck	Specifies the path of the **usrck** command.
/etc/passwd	Contains basic user attributes.
/etc/security/user	Contains the extended attributes of users.
/etc/group	Contains basic group attributes.
/etc/security/group	Contains the extended attributes of groups.
/etc/security/lastlog	Contains the last login attributes for users.
/etc/security/limits	Contains the process resource limits of users.
/etc/security/audit/config	Contains audit system configuration information.
/etc/security/login.cfg	Contains configuration information.

Related Information

The **grpck** command, **pwdck** command, **sysck** command.

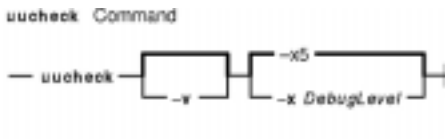
Security Administration in *AIX Version 4.3 System Management Concepts: Operating System and Devices* describes the identification and authentication of users, discretionary access control, the trusted computing base, and auditing.

uucheck Command

Purpose

Checks for files and directories required by BNU.

Syntax



uucheck [**−v**] [**−x** *DebugLevel*]

Description

The **uucheck** command verifies the presence of the files and directories required by the Basic Networking Utilities (BNU) facility. The command also checks for some errors in the **/etc/uucp/Permissions** file.

> **Note:** The **uucheck** command does not check for correct file and directory modes or for errors in the **/etc/uucp/Permissions** file, such as duplicate login or machine names.

Issue the **uucheck** command from the command line after installing the BNU program, configuring the BNU facility for your site, or making changes in part of the BNU facility, such as the **/etc/uucp/Permissions** file.

> **Note:** Only someone with root user authority can use the **uucheck** command at the command line.

Flags

−v	Displays a detailed explanation of how BNU interprets the **/etc/uucp/Permissions** file.
−x*DebugLevel*	Displays debugging information. The valid range for the *DebugLevel* variable is 0 to 9, with a default of 5. The higher the number, the more detailed the information.

Examples

1. To find out how the BNU programs interpret the **/etc/uucp/Permissions** file, enter:

   ```
   uucheck -v
   ```

   The **−v** flag instructs the **uucheck** command to verify that the BNU files exist and displays a detailed explanation of how the BNU programs interpret the **/etc/uucp/Permissions** file. The output is similar to the following:

   ```
   *** uucheck: Check Required Files and Directories
   *** uucheck: Directories Check Complete

   *** uucheck:  Check /etc/uucp/Permissions file
   ** LOGNAME PHASE (when they call us)

   When a system logs in as: (unostro)
   ```

```
   We DO allow them to request files.
   We WILL send files queued for them on this call.
   They can send files to
      /
   They can request files from
      /
   Myname for the conversation will be plague.austin..
   PUBDIR for the conversation will be
   /var/spool/uucppublic.

** MACHINE PHASE (when we call or execute their uux requests)

When we call system(s): (nostromo)
   We DO allow them to request files.
   They can send files to
      /
   They can request files from
      /
   Myname for the conversation will be plague.austin..
   PUBDIR for the conversation will be
   /var/spool/uucppublic.

Machine(s): (nostromo)
CAN execute the following commands:
command (ALL), fullname (ALL)

*** uucheck:  /etc/uucp/Permissions Check Complete
```

For an explanation of these permissions, see the **/etc/uucp/Permissions** file.

2. To debug with the **uucheck** command, enter:

```
uucheck -x8
```

The `-x8` flag produces extensive debugging output.

Files

/etc/uucp/Permissions	Describes access permissions for remote systems.
/etc/uucp/Systems	Describes accessible remote systems.

Related Information

The **uucp** command, **uustat** command, **uux** command.

The **uucico** daemon, **uusched** daemon.

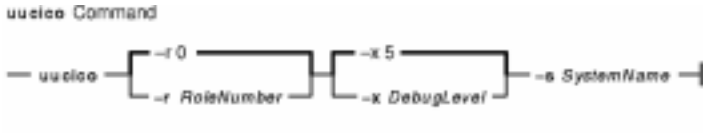
How to Configure BNU in *AIX Version 4.3 System Management Guide: Communications and Networks*.

uucico Daemon

Purpose

Transfers Basic Networking Utilities (BNU) command, data, and execute files to remote systems.

Syntax



uucico [**−r***RoleNumber*] [**−x***DebugLevel*] **−s***SystemName*

Description

The **uucico** daemon transfers Basic Networking Utilities (BNU) command (**C.***), data (**D.***), and execute (**E.***) files, created by the **uucp** and **uux** commands, to a specified remote system. Both the local and remote systems run the **uucico** daemon, and the two daemons communicate with each other to complete transfer requests.

The **uucico** daemon performs the following actions:

1. Scans the spooling directory (**/var/spool/uucp/***SystemName*) on the local system for transfer requests.
2. Selects the device used for the communications connection after checking the **/etc/uucp/Devices** file and the lock files in the **/etc/locks** directory.
3. Places a call to the specified remote system using information in the **Systems**, **Dialers**, and **Dialcodes** files located in the **/etc/uucp** directory.
4. Performs the required login sequence specified in the **Systems** file.
5. Checks permissions listed in the **/etc/uucp/Permissions** file.
6. Checks scheduling limits in the **Maxuuscheds** and **Maxuuxqts** files located in the **/etc/uucp** directory.
7. Runs all transfer requests from both the local and the remote system, placing the transferred files in the public directories (**/var/spool/uucppublic/***).
8. Logs transfer requests and completions in files in the **/var/spool/uucp/.Log/uucico** directory.
9. Notifies specified users of transfer requests.

Usually the **uucico** daemon is called by the **uucp** and **uux** commands when needed and is started periodically by the BNU scheduling daemon, **uusched**, which is started by the **cron** daemon.

The **uucico** daemon can be started from the command line for debugging. The BNU **uutry**, **Uutry**, and **uukick** commands also start the **uucico** daemon with debugging turned on.

Notes:

1. Either you must be in the **/usr/sbin/uucp** directory when you call the **uucico** daemon, or you must call the daemon with the full path name, **/usr/sbin/uucp/uucico**.
2. In the case of a **uux** command request for the execution of a command on a remote system, the **uucico** daemon transfers the files and the **uuxqt** daemon executes the command on the remote system.

Flags

−r *RoleNumber* Specifies the server and client relationship. The role numbers are 1 for server mode and 0 for client mode. If the **−r** flag is not used, the **uucico** daemon is started in client mode (**−r** 0), because the **uucico** daemon is generally started automatically by a BNU command or daemon. When the **uucico** daemon is started manually, this flag should be set to 1.

−x *DebugLevel* Displays debugging information on the screen of the local terminal. The valid range for the *DebugLevel* variable is 0 to 9, with a default of 5. Higher numbers cause the information to be more detailed. This flag is useful for diagnosing problems with the expect−send sequence in the **/etc/uucp/Systems** file.

−s *SystemName* Specifies the name of the remote system. This flag is required when starting the **uucico** daemon from the command line. The *SystemName* variable is supplied internally when the **uucico** daemon is started automatically.

Note: System names must contain only ASCII characters.

Example

To call the **uucico** daemon from the command line, enter:

```
/usr/sbin/uucp/uucico −r 1 −s hera &
```

to start the daemon as a background process and contact remote system `hera`.

Files

/etc/locks/*	Contains lock files which prevent multiple uses of devices and multiple calls to systems.
/usr/sbin/uucp/*	Contains the **uucico** daemon and the configuration files for BNU.
/etc/uucp/Devices	Contains information about available devices.
/etc/uucp/Dialcodes	Contains dialing code abbreviations.
/etc/uucp/Dialers	Specifies initial handshaking on a connection.
/etc/uucp/Maxuuscheds	Limits scheduled jobs.
/etc/uucp/Maxuuxqts	Limits remote command executions.
/etc/uucp/Permissions	Describes access permissions for remote systems.
/etc/uucp/Systems	Describes accessible remote systems.
/var/spool/uucp/.Admin/errors	Lists **uucico** daemon errors that BNU cannot correct.
/var/spool/uucp/.Log/uucico /*	Contains **uucico** daemon log files.
/var/spool/uucp/.Status/*SystemName*	Lists the last time a remote system was contacted and the minimum time until the next retry.
/var/spool/uucp/*SystemName* **/***	Contains **C.***, **D.***, and **X.*** files to be transferred by the **uucico** daemon.
/var/spool/uucp/*SystemName***/C.***	Contains command files.
/var/spool/uucp/*SystemName***/D.***	Contains data files.
/var/spool/uucp/*SystemName***/X.***	Contains execute files.
/var/spool/uucppublic/*	Contain files after transfer by the **uucico** daemon.

Related Information

The **cron** daemon, **uusched** daemon.

The **uucp** command, **uukick** command, **uuq** command, **uustat** command, **uusnap** command, **uutry** command, **Uutry** command, **uux** command.

How to Monitor a BNU Remote Connection, How to Monitor a BNU File Transfer, How to Use the uucico Daemon to Debug BNU Login Failures in *AIX Version 4.3 System Management Guide: Communications and Networks*.

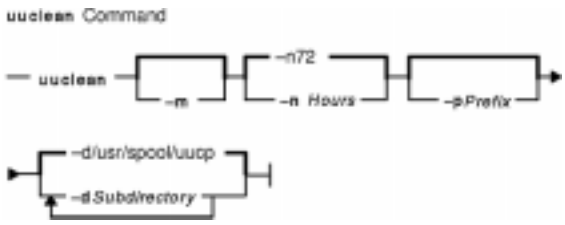
Understanding the BNU Daemons, Understanding the BNU File and Directory Structure in *AIX Version 4.3 System Management Guide: Communications and Networks*.

uuclean Command

Purpose

Removes files from the BNU spool directory.

Syntax



/usr/sbin/uucp/uuclean [**−m**] [**−n***Hours*] [**−p***Prefix*] [**−d***Subdirectory*]

Description

The **uuclean** command checks the Basic Networking Utilities (BNU) spool directory (**/var/spool/uucp**) for files with the specified prefixes and deletes those that are older than the given number of hours. If the **−n***Hours* flag is not included, the **uuclean** command deletes files that are older than 72 hours.

If the **−p** flag is not included, the **uuclean** command deletes all files in the specified subdirectories of the spool directory that meet the age requirement. If the **−d** flag is not included, the command deletes all the files (that meet the age and prefix requirements) in all the subdirectories of the spool directory. Thus if neither the **−d** or the **−p** flag is included, the **uuclean** command deletes *all* files in *all* subdirectories of the **/var/spool/uucp** directory that meet the age requirement.

If the **−m** flag is not specified, the **uuclean** command sends mail to owners of all command (**C.***) files that it deletes. If the **−m** flag is specified, the command sends mail to the owner of each file it deletes, including data (**D.***) and execute (**X.***) files. The mail message includes the name of the deleted file.

The **uuclean** command is usually run by the **cron** daemon.

> **Note:** Only someone with root user authority or who is logged in as **uucp** can issue the **uuclean** command.

Flags

−d*Subdirectory*	Deletes files from the specified subdirectory of the **/var/spool/uucp** directory if they match specifications given with the **−n** and **−p** flags. If the **−d** flag is not specified, the **uuclean** command checks all subdirectories of the **/var/spool/uucp** directory. Up to 10 subdirectories can be specified with the **−d** flag.
−m	Instructs the **uuclean** command to send mail to the owner of each file when it is deleted.
−n*Hours*	Deletes files whose ages are more than the number of hours specified by the *Hours* variable, if they match specifications given with the **−d** and **−p** flags. The default is 72 hours.
−p*Prefix*	Deletes files with the prefix given by the *Prefix* variable, if they match specifications given with the **−n** and **−d** flags. Up to 10 prefixes can be specified with the **−p** flag.

Examples

1. To delete all old command files, enter:

   ```
   /usr/sbin/uucp/uuclean -pC
   ```

   This command deletes all files in all subdirectories of the **/var/spool/uucp** directory whose names begin with `C` and that are older than 72 hours (the default). The system sends mail to the original owner of each file, stating that the file has been deleted.

2. To delete all old files from the spool directory for systems `venus` and `nostromo`, enter:

   ```
   /usr/sbin/uucp/uuclean -n84 -dvenus -dnostromo
   ```

   This command deletes all files in the **/var/spool/uucp/venus** and **/var/spool/uucp/nostromo** directories that are older than 84 hours. By default, the system notifies owners of **C.*** files that the files have been deleted; however, it does not notify owners of other files it deletes.

3. To delete all old files from all spool directories and notify users that they have been deleted, enter:

   ```
   /usr/sbin/uucp/uuclean -m
   ```

   This command deletes all files in all subdirectories of the spool directory, if the files are older than 72 hours (the default). It sends mail to the owner of each file it deletes.

4. To schedule the **uuclean** command to be started periodically by the **cron** daemon, add an entry similar to the following to your **/var/spool/cron/crontabs/uucp** file:

   ```
   15 22 * * * /usr/sbin/uucp/uuclean -n96 -pC -pD -pX
   ```

This entry will cause the **cron** daemon to start the **uuclean** command at 22:15 (10:15 p.m.) daily. The **uuclean** command will delete all command (**C.***), data (**D.***), and execute (**X.***) files that are older than 96 hours from all subdirectories of the spool directory.

Files

/usr/sbin/uucp/uuclean	Contains the **uuclean** command.
/var/spool/uucp /*	Contains spooling files removed by the **uuclean** command.
/var/spool/cron/crontabs/uucp	Schedules **uucp** jobs for the **cron** daemon.

Related Information

The **uucp** command, **uux** command, **uucleanup** command, **uudemon.cleanu** command.

The **uucico** daemon, **cron** daemon.

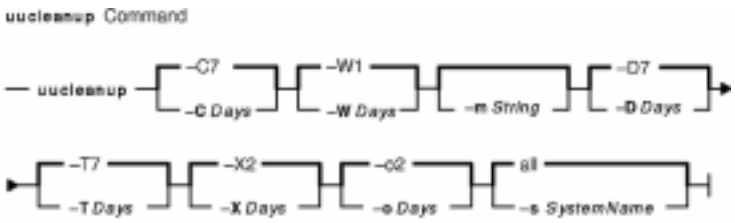
Understanding the BNU File and Directory Structure, Using BNU Maintenance Commands in *AIX Version 4.3 System Management Guide: Communications and Networks*.

uucleanup Command

Purpose

Deletes selected files from the Basic Networking Utilities (BNU) spooling directory.

Syntax



uucleanup [**−C***Days*] [**−W***Days*] [**−m***String*] [**−D***Days*] [**−T***Days*] [**−X***Days*] [**−o** *Days*] [**−s***SystemName*]

Description

The Basic Networking Utilities (BNU) **uucleanup** command scans the spooling directory (**/var/spool/uucp**) for files that are older than a specified number of days and removes them. The **uucleanup** command performs the following tasks:

- Informs the requester of send and receive requests for systems that cannot be reached.
- Warns users about requests that have been waiting for a given number of days. The default is 1 day.
- Returns to the sender mail that cannot be delivered.
- Removes from the spool directory all other files older than a specified number of days.

Notes:

1. The **uucleanup** command is not usually entered on the command line but is executed by the **uudemon.cleanu** command, a shell procedure. Only someone with root user privileges can issue the **uucleanup** command from the command line.
2. When BNU is installed, automatic cleanup is not enabled. Edit the **/var/spool/cron/crontabs/uucp** file and remove the comment character (#) from the beginning of the **uudemon.cleanu** line to instruct the **cron** daemon to start the **uudemon.cleanu** command.

Flags

−C*Days*	Removes **C.*** (command) files as old as, or older than, the number of days specified by the *Days* variable, and notifies the requester that the files have been deleted. The default time is 7 days.
−D*Days*	Removes **D.*** (data) files as old as, or older than, the number of days specified by the *Days* variable. Also attempts to deliver any remaining mail messages. The default time is 7 days.
−m*String*	Includes a specified line of text in the warning message generated by the **−W***Days* option. The default line is `See your local administrator to locate the problem`.
−o*Days*	Removes other files as old as, or older than, the number of days specified by the *Days* variable. The default time is 2 days.

−s*SystemName* Executes the **uucleanup** command only on the spooling directory specified by the *System* variable. The default is to clean up all BNU spooling directories.

Note: System names can contain only ASCII characters.

−T*Days* Removes **TM.*** (temporary) files as old as, or older than, the number of days specified by the *Days* variable. Also attempts to deliver any remaining mail messages. The default time is 7 days.

−W*Days* Sends an electronic mail message to the requester warning that **C.*** (command) files as old as, or older than, the number of days specified by the *Days* variable are still in the spooling directory. The message includes the job ID and, if the request included mail, the mail message. The administrator can use the **−m** option to include a message line telling whom to call to check the problem. The default time is 1 day.

−X*Days* Removes any **X.*** (execute) files as old as, or older than, the number of days specified by the *Days* variable. The default time is 2 days.

Examples

Warning Users That Their Command Files Have Not Been Sent

1. To send a warning for **C.*** (command) files 2 or more days old, enter:

   ```
   uucleanup -W2
   ```

   This warns the requester that the files have not been sent.

2. To send a message with the warning, enter:

   ```
   uucleanup -m"Check these files waiting in the BNU job queue."
   ```

This locates **C.*** (command) files 1 or more days old (default), warns requesters that their files have not been sent, and gives the message: `Check these files waiting in the BNU job queue.`

Cleaning Up Command, Data, Execute, and Miscellaneous Files

1. To clean up command files 5 or more days old, enter:

   ```
   uucleanup -C5
   ```

   This removes all **C.*** (command) files 5 or more days old and sends an appropriate message to the requesters.

2. To clean up data and execute files 3 or more days old, enter:

   ```
   uucleanup -D3 -X3
   ```

   This removes all **D.*** (data) files and all **X.*** (execute) files 3 or more days old.

3. To clean up all files at once using defaults, enter:

   ```
   uucleanup
   ```

   This removes all **C.***, **D.***, **T.***, and **X.*** files, and all other files older than the default times.

> **Note:** Whenever the **−C** and **−W** flags are used together, make sure the value specified for the **−W** flag is less than that for the **−C** flag. Otherwise, the **−C** flag will delete all the **C.*** (command) files before any warnings can be printed.

Cleaning Up Files for a Specific System

To delete files for one system, enter:

```
uucleanup -shera
```

This removes all files using defaults for system `hera`, but does not remove any files for any other systems.

Files

/usr/sbin/uucp/*	Contains the **uudemon.cleanu** shell procedure and all the configuration files for BNU.
/var/spool/cron/crontabs/uucp	Schedules BNU jobs for the **cron** daemon, including the **uudemon.cleanu** shell procedure.
/var/spool/uucp/*	Contain files removed by the **uucleanup** command.

Related Information

The **cron** daemon.

The **uucp** command, **uudemon.cleanu** command, **uustat** command, **uux** command, **uuclean** command.

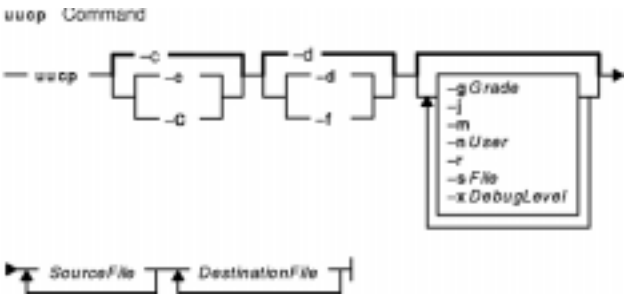
Maintaining BNU in *AIX Version 4.3 System Management Guide: Communications and Networks*.

uucp Command

Purpose

Copies files from one system to another.

Syntax



uucp [**−c** | **−C**] [**−d** | **−f**] [**−g***Grade*] [**−j**] [**−m**] [**−n***User*] [**−r**] [**−s***File*] [**−x***DebugLevel*] *SourceFile ...DestinationFile ...*

Description

The **uucp** command is a Basic Networking Utilities (BNU) command that copies one or more source files from one system to one or more destination files on another UNIX system. Files can be copied within a local system, between a local and a remote system, and between two remote systems.

The **uucp** command accomplishes the file transfer in two steps: first, by creating a command (**C.***) file in the spooling directory on the local computer and then by calling the **uucico** daemon to send the request to the specified computer. Command files include information such as the full path name of the source and destination files and the sender's login name. The full path name of a command file is a form of the following:

/var/spool/uucp/*SystemName***/C.***SystemNameNxxxx*

where *N* is the grade of the request and *xxxx* is the hexadecimal sequence number used by BNU.

If the **uucp** command is used with the **−C** flag to copy the files to the spool directory for transfer, the **uucp** command creates not only a command file, but also a data (**D.***) file that contains the actual source file. The full path name of a data file is a form of the following:

/var/spool/uucp/*SystemName***/D.***SystemNamexxxx###*

Once the command files (and data files, if necessary) are created, the **uucp** command then calls the **uucico** daemon, which in turn attempts to contact the remote computer to deliver the files.

It is useful to issue the **uuname** command to determine the exact name of the remote system before issuing the **uucp** command. The **uulog** command provides information about **uucp** activities with another system.

Source and Destination File Names

File names and system names can contain only ASCII characters. Each can either be a path name on the local system or have the following form:

SystemName!*PathName*

where *SystemName* is taken from a list of system names that BNU knows about.

The destination *SystemName* can also be a list of names, such as the following:

SystemName!*SystemName*! . . . ! *SystemName*!*PathName*

In this case, an attempt is made to send the file using the specified route to the destination. Make sure that intermediate nodes in this route are willing to forward information, and that they actually talk to the next system.

The shell pattern−matching characters ? (question mark), * (asterisk), and [. . .] (brackets and ellipsis) can be used in the path names of the source file; the appropriate system expands them. The shell pattern−matching characters should not be used in the path name of the destination file.

If the *DestinationFile* is a directory rather than a file, the **uucp** command uses the last part of the *SourceFile* name to name the transferred file on the remote system.

Path Names

Path names for the *SourceFile* and *DestinationFile* parameters contain only ASCII characters. Paths for the source file can be one of the following:

- A full path name
- A relative path name

Paths for the *DestinationFile* parameter can be in the forms for the *SourceFile* parameter, or can be one of the following:

- A path name preceded by **~***User* (for example, `~jkimble`) where *User* is a login name on the remote system. The specified user's login directory is then considered the destination of the transfer. If the user specifies an invalid login name, the files are transferred to the public directory, **/var/spool/uucppublic**, which is the default.
- A path name preceded by ~/*Destination*, where *Destination* is appended to **/var/spool/uucppublic**. The destination is treated as a file name unless more than one file is being transferred by the request, the destination already exists as a directory on the remote system, or the destination is specified as a directory.

  To specify the destination as a directory, follow the destination name with a / (slash). For example, `~/amy/` as the destination creates the directory `/var/spool/uucppublic/amy`, if it does not already exist, and puts the requested files in that directory.

Permissions

- The system administrator should restrict the access to local files by users on other systems.
- When transmitting files, the **uucp** command preserves execute permissions and grants read and write permissions to the owner, the group, and all others. (The **uucp** command owns the file.)
- Sending files to arbitrary *DestinationFile* path names on other systems or getting files from arbitrary *SourceFile* path names on other systems often fails because of security restrictions. The files specified in the path name must give read or write permission not only for the same group of users but also for any group.
- Protected files and files in protected directories owned by the requestor can be sent by the **uucp** command.

Flags

−c	Prevents files from being copied. This flag is the default and should not be used with the **−C** flag. If both flags are specified, the **−c** flag is overridden.
−C	Copies local files to the spool directory for transfer. Depending on the configuration of the **Poll** and **Systems** files and on how often the **uusched** daemon is run, the files may be transferred immediately on demand polling or in the future. Occasionally, problems occur while transferring a source file; for example, the remote computer may not be working or the login attempt may fail. In such a case, the file remains in the spool directory until it is either transferred successfully or removed by a cleanup command. This flag counteracts the **−c** flag.
−d	Creates any intermediate directories needed to copy the source files to the destination files on a remote system. Instead of first creating a directory and then copying files to it, the **uucp** command can be entered with the destination path name, and the BNU Program will create the required directory. This flag is the default and cannot be used with the **−f** flag.
−f	Does not create intermediate directories during the file transfer. This flag is used if the destination directory already exists and you do not want BNU to write over it. This command counteracts the **−d** flag.
−g*Grade*	Specifies when the files are to be transmitted during a particular connection. The *Grade* variable is a single number (0 to 9) or letter (A to Z, a to z); lower ASCII−sequence characters cause the files to be transmitted earlier than do higher sequence characters. The number 0 is the highest (earliest) grade; z is the lowest (latest) grade. The default is **N**.
−j	Displays the job identification number of the transfer operation on standard output. This job ID can be used with the **uustat** or **uuq** command to obtain the status of a particular job or with the **uustat −k** command or **uuq −d** command to terminate the transfer before it is completed.
−m	Sends a mail message to the requester when the source file is successfully copied to the destination file on a remote system. The message is sent to the requester's mailbox, **/var/spool/mail/***User*. The **mail** command does not send a message for a local transfer. The **−m** flag works only when sending files or receiving a single file. It does not work when forwarding files.
−n*User*	Notifies the recipient on the remote system identified by the *User* entry that a file has been sent. The mail system does not send a message for a local transfer. User names can contain only ASCII characters. Receiving multiple files specified by the shell pattern−matching characters ? (question mark), * (asterisk), and [. . .] (brackets and ellipses) does not activate the **−n** option.
−r	Prevents the starting of the **uucico** file transfer daemon, even if the command was issued at a time when calls to the remote system are permitted. (By default, a call to the remote system is attempted if the command is issued during a time period specified in the **Poll** and **Systems** files.) The **−r** option is useful for debugging.
−s*File*	Reports the status of the transfer to the specified file. In this case, the *File* variable must designate a full path name.
−x*DebugLevel*	Displays debugging information on the screen of the local system. The *DebugLevel* variable is a number from 0 to 9. The higher the number, the more detailed the report.

Examples

1. To copy a file from the local system to a remote system, enter:

```
uucp /home/geo/f1 hera!/home/geo/f1
```

In this example, the `f1` file from the local system is copied to remote system `hera`.

2. To copy a file from the remote system and place it in the public directory, enter:

```
uucp hera!geo/f2 /var/spool/uucppublic/f2
```

In this example, the `f2` file from remote system `hera` is copied and placed in the public directory.

3. To copy a file from the remote system and place it in a directory other than the public directory, enter:

```
uucp hera!geo/f2 /home/geo/f2
```

In this example, the `f2` file from the remote system `hera` is copied to the `/home/geo/f2` directory. The `geo` login directory must allow write permission to members of the other group, for example, with mode 777.

Files

/usr/bin/uucp	Contains the **uucp** command.
/etc/uucp/Poll	File listing times when remote systems are automatically called (polled).
/etc/uucp/Systems	File describing accessible remote systems.
/etc/uucp/Sysfiles	Specifies alternate files to be used as **Systems** files.
/var/spool/uucp	Spooling directory containing BNU status information.
/var/spool/uucppublic	Public directory containing files awaiting transfer by the **uucico** daemon.
/var/spool/uucppublic/*SystemName***/C.***	Contains command files.
/var/spool/uucppublic/*SystemName***/D.***	Contains data files.

Related Information

The **ct** command, **cu** command, **mail** command, **uuclean** command, **uucleanup** command, **uulog** command, **uuname** command, **uupick** command, **uuq** command, **uustat** command, **uuto** command, **uux** command.

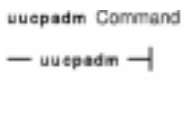
The **uucico** daemon, **uusched** daemon.

uucpadm Command

Purpose

Enters basic BNU configuration information.

Syntax



uucpadm

Description

The **uucpadm** command provides interactive entry and modification of basic BNU configuration information in the **Devices**, **Systems**, **Permissions**, **Poll**, and **Dialcodes** files in the **/etc/uucp** directory. You can use the **uucpadm** command repeatedly to adjust the same file.

When you enter the **uucpadm** command at the command line, the command displays a list of the files you can change. After you choose a file to modify, the command displays a vertical list of the names of the fields in that file. You can enter the appropriate entry in each field. When you press the Enter key, the cursor moves to the next field in the list.

The command uses a copy of a file to record changes. The original file remains unchanged until you press the Ctrl−U or Ctrl−X key sequence at the appropriate menu. You can exit to the main **uucpadm** menu at any time, without saving your changes, by using the Ctrl−D key sequence.

The help routine provides instructions for each data field. Enter a ? (question mark) in any menu field to access the help routine for that field.

Enter a **~** (tilde) in any field to enter an ASCII editor and edit the appropriate file for that field. The **uucpadm** command invokes the editor designated by the **EDITOR** environmental variable. If the **EDITOR** variable is not defined, the command invokes the **vi** editor.

If your entry for the first menu item matches an existing record, the **uucpadm** command retrieves that record for update. The command also tells you how many records have that first entry. If your entry for the first menu item does not match any existing record, the **uucpadm** command displays the word ADD at the top of the screen.

The **uucpadm** command checks the data as you enter it. If an inconsistency among the files is found, the command displays a warning message.

If the **uucpadm** command recognizes the entry you make for the first menu item, it fills in the default values for the remaining fields. For example, if you enter TCP as the Type in the **Devices** file menu, the command places a − (hyphen) in each remaining field for you. It also checks for consistency with other files and for processes that should be running on the system. For example, when you enter TCP as the Type in the **Devices** file menu, the **uucpadm** command checks to see if the **uucpd** daemon is running. If the daemon is not running, the command displays a note after the Type field, as follows:

```
Type: TCP
       <Note: Make certain uucpd is enabled.>
Line1: –
```

Note: The **uucpadm** command does not edit the **/etc/uucp/Dialers** file. Use an ASCII editor to edit this file.

Mode	File
rw	**/etc/uucp/Devices**
rw	**/etc/uucp/Dialcodes**
rw	**/etc/uucp/Permissions**
rw	**/etc/uucp/Poll**
rw	**/etc/uucp/Systems**

Examples

1. To start the **uucpadm** command, enter the following:

   ```
   /usr/sbin/uucp/uucpadm
   ```

   A menu listing the files you can change is displayed.

2. To make an entry to the **/etc/uucp/Devices** file, choose the `Add/Change Uucp Devices` option at the **uucpadm** menu. The following is a sample **uucpadm** screen defining a `direct 9600 baud` connection to system `merlin` over the `tty3` device:

   ```
   Type: merlin
   line1: tty3
   line2: –
   class: 9600
   dialers: direct
   ```

3. To make an entry to the **/etc/uucp/Systems** file, choose the `Add/Change Uucp Systems` option at the **uucpadm** menu. The following is a sample **uucpadm** screen defining the `nostromo.aus.ibm.com` system connected to an `ACU` device in `class 2400`:

   ```
   Name: nostromo.aus.ibm.com
   Time: Any
   Type: ACU
   Class: 2400
   Phone: 997–7942
   Login: nuucp
   Password: gotcha
   ```

4. To change the **/etc/uucp/Permissions** file, choose the `Add/Change Uucp Permissions File` option at the **uucpadm** menu.
   a. Following is a sample **uucpadm** screen defining a LOGNAME entry in the **Permissions** file:

      ```
      L/M: LOGNAME=uucpz
      Request: yes
      Sendfiles: yes
      Read: /
      Write: NOWRITE=/etc
      Callback:
      Commands:
      Validate: merlin:nostromo
      ```

If the remote machine is `merlin` or `nostromo`, the login ID must be `uucpz` (VALIDATE option). Remote hosts using this ID can `Request` to send files, and the local host can `Sendfiles` as requested. Users with this ID can read all files with permissions granted to the others group, and can write to all files, except those in the **/etc** directory, with permissions granted to the others group.

5. Following is a sample **uucpadm** screen defining a MACHINE entry in the **Permissions** file:

```
L/M: MACHINE=merlin
Request: yes
Sendfiles:
Read: NOREAD=/etc
Write: NOWRITE=/etc
Callback:
Commands: ALL
Validate:
```

The machine ID is `merlin`. Requests for file transfers can be made. The user can read all files and can write to all files except those in the **/etc** directory. The execution of all commands is permitted.

- To make an entry in the **/etc/uucp/Poll** file, choose the `Add/Change Uucp Poll File` option at the **uucpadm** menu. Following is a sample **uucpadm** screen defining an entry in the **Poll** file:

```
System: merlin
Hours: 0 7 13 19
```

This entry instructs BNU to poll the `merlin.aus.ibm.com` system at 2400 hours (midnight), 700 hours (7 a.m.), 1300 hours (1 p.m.), and 1900 hours (7 p.m.).

- To make an entry in the **/etc/uucp/Dialcodes** file, choose the `Add/Change Uucp Dialcodes` option at the **uucpadm** menu. Following is a sample **uucpadm** screen defining an entry in the **Dialcodes** file:

```
Abr: LA
Dialcode: 1-213-
```

This entry assigns `LA` as the abbreviation for the Los Angeles area code.

Files

/usr/sbin/uucp/uucpadm	Contains the **uucpadm** command.
/etc/uucp/Devices	Contains information about available devices.
/etc/uucp/Dialcodes	Contains dialing code abbreviations.
/etc/uucp/Dialers	Specifies initial handshaking on a connection.
/etc/uucp/Permissions	Describes access permissions for remote systems.
/etc/uucp/Poll	Specifies when BNU polls remote systems to initiate tasks.
/etc/uucp/Systems	Describes accessible remote systems.

Related Information

The **uucp** command, **uuname** command.

The **uucheck** command checks the **/etc/uucp/Permissions** file for correct configuration.

Example of a BNU Configuration for a Telephone Connection, Example of a BNU Configuration for a TCP/IP Connection, and Example of a BNU Configuration for a Direct Connection in *AIX Version 4.3 System Management Guide: Communications and Networks*.

Understanding the BNU File and Directory Structure in *AIX Version 4.3 System Management Guide: Communications and Networks*.

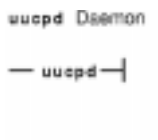
How to Configure BNU in *AIX Version 4.3 System Management Guide: Communications and Networks*.

uucpd Daemon

Purpose

Handles communications between BNU and TCP/IP.

Syntax



The **uucpd** daemon cannot be started from the command line. It is started by the **inetd** daemon.

uucpd

Description

The **uucpd** daemon is an internal program that enables users of systems linked by the Basic Networking Utilities (BNU) program to establish a TCP/IP connection to other systems linked over a Token−Ring, Ethernet, or other network.

The **uucpd** daemon is a subserver of the **inetd** daemon. The **uucpd** daemon must be running as a background process on all the networked systems before the BNU program can use TCP/IP system to communicate. If the **uucpd** daemon is not running, reconfigure the **inetd** daemon to start the **uucpd** daemon. Use the **netstat** command to find out if the **uucpd** daemon is running.

Files

/etc/hosts	Contains the host name table used by TCP/IP.
/etc/inetd.conf	Contains the configuration of the **inetd** daemon.
/etc/services file	Defines socket assignments used by TCP/IP.
/usr/sbin /uucpd	Contains the **uucpd** daemon.
/etc/uucp/Devices	Contains information about available devices.
/etc/uucp/Permissions	Describes access permissions for remote systems.
/etc/uucp/Systems	Describes accessible remote systems.

Related Information

The **inetd** daemon.

How to Configure the inetd Daemon in *AIX Version 4.3 System Management Guide: Communications and Networks*.

TCP/IP Overview for System Management in *AIX Version 4.3 System Management Guide: Communications and Networks*.

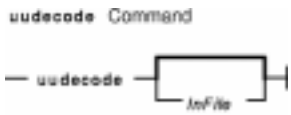
Understanding the BNU Daemons, How to Configure BNU in *AIX Version 4.3 System Management Guide: Communications and Networks*.

uudecode Command

Purpose

Decodes a binary file that was used for transmission using electronic mail.

Syntax



uudecode [*InFile*]

Description

The **uudecode** command reads an encoded file, strips off leading and trailing lines added by mailers, and recreates the original file with the specified mode and name. Decoding a file causes the result to be automatically saved to a file. The file name is identical to the remote file argument originally supplied to the **uuencode** command.

Parameters

InFile Specifies the name of the file to decode.

Example

To decode the file `/tmp/con` on a local system that was encoded with the follwing command:

```
uuencode /usr/lib/boot/unix pigmy.goat > /tmp/con
```

enter:

```
uudecode /tmp/con
```

The file `pigmy.goat` will be identical to the originally encoded file `/usr/lib/boot/unix.`

Files

/usr/bin/uudecode Contains the **uudecode** command.

Related Information

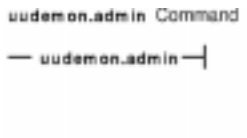
The **mail** command, **rmail** command, **sendmail** command, **uucp** command, **uuencode** command, **uusend** command, **uux** command.

uudemon.admin Command

Purpose

Provides periodic information on the status of BNU file transfers.

Syntax



uudemon.admin

Description

The **/usr/sbin/uucp/uudemon.admin** command is a shell procedure that mails status information about the Basic Networking Utilities (BNU) activities to the **uucp** login ID at intervals specified in the **/var/spool/cron/crontabs/uucp** file. The command executes both the **uustat −p** and the **uustat −q** commands:

- The **−p** flag instructs the **uustat** command to run the **ps −flp** command (process status, which generates a full, long list of specified process IDs) for all process ID (PID) numbers in the lock files.
- The **−q** flag lists the jobs currently queued to run on each system. These jobs either are waiting to execute or are in the process of executing. If a status file exists for the system, its date, time, and status information are reported.

Execute the **uudemon.admin** command at least once a day. The **uudemon.admin** command is not enabled when you install the BNU program. To run this command automatically, edit the **/var/spool/cron/crontabs/uucp** file, removing the comment character (#) from the beginning of the line that governs running the **uudemon.admin** command.

Example

To run the **uudemon.admin** command automatically, edit the **/var/spool/cron/crontabs/uucp** file and remove the comment character (#) from the beginning of the **uudemon.admin** command line. Change:

```
#48 8,12,16 * * * /usr/bin/sh −c
 "/usr/sbin/uucp/uudemon.admin > /dev/null"
```

to:

```
48 8, 12, 16 * * * /usr/bin/sh −c "/usr/sbin/uucp/uudemon.admin > /dev/null"
```

The `48` notation represents minutes, the `8,12,16` notation represents hours based on the 24−hour clock, and the three asterisks (`* * *`) are placeholders representing the day of the month, the month of the year, and the day of the week, respectively. This line therefore instructs the **cron** daemon to run the **uudemon.admin** command daily at 48 minutes past the hours 0800, 1200, and 1600−that is, at 8:48 a.m., 12:48 p.m., and 4:48 p.m. respectively.

Note: These run intervals are defaults. By altering them, you can change the times at which the **cron** daemon executes the **uudemon.admin** command to fit the needs of your site.

Files

/usr/sbin/uucp/uudemon.admin	Contains the **uudemon.admin** command and the configuration files for BNU.
/etc/locks/*	Contains lock files which prevent multiple uses of devices and multiple calls to systems.
/var/spool/cron/crontabs/uucp	Schedules BNU jobs, including the **uudemon.admin** command, for the **cron** daemon.

Related Information

The **uustat** command.

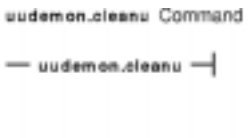
Using BNU Maintenance Commands in *AIX Version 4.3 System Management Guide: Communications and Networks*.

uudemon.cleanu Command

Purpose

Cleans up BNU spooling directories and log files.

Syntax



uudemon.cleanu

Description

The **/usr/sbin/uucp/uudemon.cleanu** command is a shell script that cleans up the Basic Networking Utilities (BNU) spooling directories and log files. The command deletes files in the spooling directories that are as old as, or older than, a specified number of days, and then removes empty spooling directories.

The **uudemon.cleanu** command also updates archived log files by removing log information more than three days old. The command removes log files for individual computers from the **var/spool/uucp/.Log** directory, merges them, and places them in the **var/spool/uucp/.Old** directory, which contains old log information.

After performing the cleanup operations, the **uudemon.cleanu** command mails the **uucp** login ID a summary of the status information gathered during the current day.

Instruct the **cron** daemon to run the **uudemon.cleanu** command daily, weekly, or at longer intervals, depending on the amount of transactions the **uucico** and **uuxqt** daemons perform on the local system.

To run this command automatically, remove the comment character (#) at the beginning of the **uudemon.cleanu** command line in the **/var/spool/cron/crontabs/uucp** file.

> **Note:** The **uudemon.cleanu** command is not usually entered on the command line but is instead executed by the **cron** daemon.

Example

To run the **uudemon.cleanu** procedure automatically, edit the **/var/spool/cron/crontabs/uucp** file and uncomment the **uudemon.cleanu** line. Change:

```
# 45 23 * * * /usr/bin/sh −c
 "/usr/sbin/uucp/uudemon.cleanu > /dev/null"
```

to:

```
45 23 * * * /usr/bin/sh −c "/usr/sbin/uucp/uudemon.cleanu > /dev/null"
```

The 45 notation represents minutes, the 23 notation represents hours based on the 24−hour clock, and the three asterisks (* * *) are placeholders representing the day of the month, the month of the year, and the

day of the week, respectively. This line therefore instructs the **cron** daemon to run the **uudemon.cleanu** shell procedure at 45 minutes after hour 2300−that is, at 11:45 p.m.

Notes:

1. These run intervals are defaults. By altering them, you can change the times at which the **cron** daemon executes the **uudemon.cleanu** command so that they fit the needs of your site.
2. The system allots the BNU program a specified amount of storage space for any one particular log file; the number of blocks is determined by the default **ulimit** value. If the **uudemon.cleanu** command fails to execute because the **ulimit** value is set too low for the requirements of the local system, delete the **uudemon.cleanu** command line (shown previously) from the **/var/spool/cron/crontabs/uucp** file and add the following entry to the **root** crontabs file, **/var/spool/cron/crontabs/root**:

```
45 23 * * * ulimit 5000; /usr/bin/su uucp
 -c "/usr/sbin/uucp/uudemon.cleanu > /dev/null"
```

Put the text on one line when entering it in the **root** crontabs file.

Files

/usr/sbin/uucp/uudemon.cleanu	Contains the **uudemon.cleanu** command.
/var/spool/cron/crontabs/uucp	Schedules BNU jobs, including the **uudemon.cleanu** command, for the **cron** daemon.
/var/spool/cron/crontabs/root	Schedules root user jobs for the **cron** daemon.
/var/spool/uucp/.Log/*	Contains the BNU program log files.

Related Information

The **uustat** command, **uux** command.

The **cron** daemon, **uucico** daemon, **uuxqt** daemon.

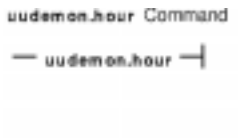
Working with BNU Log Files in *AIX Version 4.3 System Management Guide: Communications and Networks*.

uudemon.hour Command

Purpose

Initiates file transport calls to remote systems using the BNU program.

Syntax



uudemon.hour

Description

The **/usr/sbin/uucp/uudemon.hour** command is a shell procedure used by the Basic Networking Utilities (BNU). In conjunction with the **Poll** file, the **uudemon.poll** command, and the **/var/spool/cron/crontabs/uucp** file, the **uudemon.hour** command initiates calls to remote systems.

The **uudemon.hour** command calls the following programs, which are involved in transferring files between systems at specified hourly intervals:

- The **uusched** daemon first searches the spooling directory on the local system for command files that have not been transferred to the specified remote system, and then schedules the transfer of those files.
- The **uuxqt** daemon searches the spooling directory for execute files that have been transferred to the local system but have not yet been processed on that system.

Instruct the **cron** daemon to run the **uudemon.hour** command at specified hourly intervals. The frequency at which you run the **uudemon.hour** command depends on the amount of file−transfer activity originating from the local computer. If users on the local system initiate a large number of file transfers, you may need to specify that the **cron** daemon should start the **uudemon.hour** command several times an hour. If the number of file transfers originating from the local system is low, you can probably specify a start time once every 4 hours, for example.

To run the **uudemon.hour** command automatically, remove the comment character (#) from the beginning of the **uudemon.hour** command line in the **/var/spool/cron/crontabs/uucp** file.

> **Note:** The **uudemon.hour** command is not usually entered on the command line, but is executed by the **cron** daemon.

Example

To run the **uudemon.hour** command automatically, edit the **/var/spool/cron/crontabs/uucp** file and remove the comment character (#) at beginning of the **uudemon.hour** command line. Change:

```
#25,55 * * * * /usr/bin/sh −c "/usr/sbin/uucp/uudemon.hour >  /dev/null"
```

to:

```
25,55 * * * * /usr/bin/sh −c "/usr/sbin/uucp/uudemon.hour >  /dev/null"
```

The `25,55` notation represents minutes, and the four asterisks (`* * * *`) are placeholders representing the hour of the day, the day of the month, the month of the year, and the day of the week, respectively. Therefore, this line instructs the **cron** daemon to run the **uudemon.hour** command at 25 minutes past the hour and again at 55 minutes past the hour; for example, at 8:25 and 8:55 a.m., again at 9:25 and 9:55 a.m., and again every hour of every day.

Notes:

1. These run intervals are defaults. By altering them, you can change the times at which the **cron** daemon executes the **uudemon.hour** command to fit the needs of your site. For example, to run the **uudemon.hour** command once every 4 hours, enter the numeral 4 in the time−interval field.
2. If you change the run times for the **uudemon.hour** command, you should also change the run times for the **uudemon.poll** command so that it polls remote systems 5 to 10 minutes before the **uudemon.hour** command is run.

Files

/usr/sbin/uucp/uudemon.hour	Contains the **uudemon.hour** command.
/etc/uucp/Poll	Specifies when the BNU program should poll remote systems to initiate tasks.
/var/spool/cron/crontabs/uucp	Schedules BNU jobs, including the **uudemon.hour** and **uudemon.poll** commands, for the **cron** daemon.

Related Information

The **uudemon.poll** command.

The **cron** daemon, **uusched** daemon, **uuxqt** daemon.

How to Set Up BNU Polling of Remote Systems in *AIX Version 4.3 System Management Guide: Communications and Networks*.

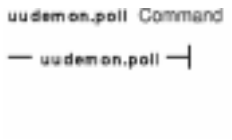
Using BNU Maintenance Commands, Understanding the BNU Daemons in *AIX Version 4.3 System Management Guide: Communications and Networks*.

uudemon.poll Command

Purpose

Polls the systems listed in the BNU **Poll** file.

Syntax



uudemon.poll

Description

The **/usr/sbin/uucp/uudemon.poll** command is a shell procedure used by the Basic Networking Utilities (BNU). In conjunction with the **/etc/uucp/Poll** file, the **uudemon.hour** command, and the **/var/spool/cron/crontabs/uucp** file, the **uudemon.poll** command initiates calls to remote systems.

The **uudemon.poll** command performs the following actions:

- Polls (contacts) the systems listed in the **Poll** file (**/etc/uucp/Poll**).
- Creates command (**C.***) files for the systems listed in the **Poll** file.

The time at which you run the **uudemon.poll** command depends on the time at which you run the **uudemon.hour** command. In general, schedule the polling shell procedure before the hourly procedure. This schedule enables the **uudemon.poll** command to create any required command files before the **cron** daemon runs the **uudemon.hour** command.

Instruct the **cron** daemon to run the **uudemon.poll** command about 5 to 10 minutes before running the **uudemon.hour** command. To run this procedure automatically, remove the comment character (#) from the beginning of the **uudemon.poll** command line in the **/var/spool/cron/crontabs/uucp** file.

> **Note:** The **uudemon.poll** command is not usually entered on the command line, but is executed by the **cron** daemon.

Example

To run the **uudemon.poll** shell procedure automatically, edit the **/var/spool/cron/crontabs/uucp** file and remove the # (comment character) at the beginning of the line which starts the **uudemon.poll** command. Change:

```
#20,50 * * * * /usr/bin/sh −c "/usr/sbin/uucp/uudemon.poll >  /dev/null"
```

to:

```
20,50 * * * * /usr/bin/sh −c "/usr/sbin/uucp/uudemon.poll >  /dev/null"
```

The `20,50` notation represents minutes, and the four asterisks (`* * * *`) are placeholders representing the

hour of the day, the day of the month, the month of the year, and the day of the week, respectively. This line therefore instructs the **cron** daemon to run the **uudemon.poll** command at 20 minutes past the hour and again at 50 minutes past the hour−for example, at 8:20 and 8:50 a.m., and at 9:20 and 9:50 a.m.−every hour of every day.

> **Note:** Change the times at which the **cron** daemon executes the **uudemon.poll** command to correspond to the times you set up for the **uudemon.hour** command. The defaults specified in the **/var/spool/cron/crontabs/uucp** file instruct the **cron** daemon to run the **uudemon.poll** command 5 minutes before running the **uudemon.hour** command.

Files

/usr/sbin/uucp/*	Contains the **uudemon.poll** and **uudemon.hour** commands and all the configuration files for BNU.
/etc/uucp/Poll	Specifies when the BNU program should poll remote systems to initiate tasks.
/var/spool/cron/crontabs/uucp	Schedules BNU jobs, including the **uudemon.poll** command, for the **cron** daemon.

Related Information

The **uudemon.hour** command.

The **cron** daemon.

How to Set Up BNU Polling of Remote Systems in *AIX Version 4.3 System Management Guide: Communications and Networks*.

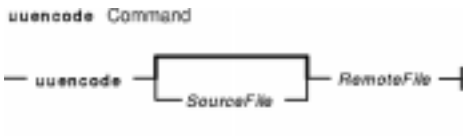
Using BNU Maintenance Commands, Understanding the BNU Daemons in *AIX Version 4.3 System Management Guide: Communications and Networks*.

uuencode Command

Purpose

Encodes a binary file for transmission using electronic mail.

Syntax



uuencode [*SourceFile*] *RemoteFile*

Description

The **uuencode** command converts a binary file to ASCII data before using BNU (or uucp) mail to send the file to a remote system. The **uudecode** command converts ASCII data created by the **uuencode** command back into its original binary form.

The **uuencode** command takes the named *SourceFile* (default standard input) and produces an encoded version on the standard output. The encoding uses only printable ASCII characters, and includes the mode of the file and the *RemoteFile* filename used for recreation of the binary image on the remote system.

Use the **uudecode** command to decode the file.

Parameters

RemoteFile Specifies the name the decoded file will have.

SourceFile Specifies the name of the binary file to convert. Default is standard input.

Examples

1. To encode the file `unix` on the local system and mail it to the user `jsmith` on another system called `mysys`, enter:

   ```
   uuencode unix unix | mail jsmith@mysys
   ```

2. To encode the file `/usr/lib/boot/unix` on your local system with the name `pigmy.goat` in the file `/tmp/con` , enter:

   ```
   uuencode /usr/lib/boot/unix pigmy.goat > /tmp/con
   ```

Files

/usr/bin/uuencode Contains the **uuencode** command.

Related Information

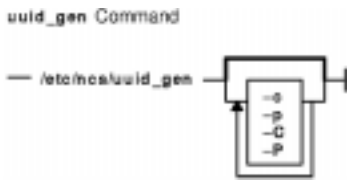
The **mail** command, **rmail** command, **sendmail** command, **uucp** command, **uudecode** command, **uusend** command, **uux** command.

uuid_gen Command (NCS)

Purpose

Generates Universal Unique Identifiers (UUIDs) for objects, types, and interfaces.

Syntax



/etc/ncs/uuid_gen [**−c**] [**−p**] [**−C**] [**−P**]

Description

The **uuid_gen** program generates Universal Unique Identifiers (UUIDs). By default, it generates a character−string representation of a UUID. The options for **uuid_gen** enable you to generate source−code representations of UUIDs, suitable for initializing variables of type **uuid_$t**.

Flags

−C Generates a C source−code representation of a UUID.
−c Generates a template, including a UUID attribute, for an interface definition in the C syntax.
−P Generates a Pascal source−code representation of a UUID.
−p Generates a template, including a UUID attribute, for an interface definition in the Pascal syntax.

Examples

1. To generate a character−string representation of a UUID, enter the following:

   ```
   /etc/ncs/uuid_gen
   ```

   This produces the following output:

   ```
   34dc23469000.0d.00.00.7c.5f.00.00.00
   ```

2. To generate a template for an interface definition in the C syntax, enter:

   ```
   /etc/ncs/uuid_gen −c
   ```

   This produces the following output:

   ```
   %c
   [
   uuid(34dc239ec000.0d.00.00.7c.5f.00.00.00),
   version(1)
   ]
   interface INTERFACENAME {
   ```

```
}
```

3. To generate a C source−code representation of a UUID, enter the following:

   ```
   /etc/ncs/uuid_gen −C
   ```

   This produces the following output:

   ```
   = { 0x34dc23af,
       0xf000,
       0x0000,
       0x0d,
       {0x00, 0x00, 0x7c, 0x5f, 0x00, 0x00, 0x00} };
   ```

4. To generate a template for an interface definition in the Pascal syntax, enter the following:

   ```
   /etc/ncs/uuid_gen −p
   ```

   This produces the following output:

   ```
   %pascal
   [
   uuid (458487b55160.02.c0.64.02.03.00.00.00),
   version (1)
   ]
   interface INTERFACENAME;

   end;
   ```

5. To generate a Pascal source−code representation of a UUID, enter the following:

   ```
   /etc/ncs/uuid_gen −P
   ```

   This produces the following output:

   ```
   := [
       time_high := 16#458487df,
       time_low := 16#9fb2,
       reserved := 16#000,
       family := chr(16#02),
       host := [chr(16#c0), chr(16#64), chr(16#02), chr(16#03),
                chr(16#00), chr(16#00), chr(16#00)]
       ]
   ```

Related Information

List of NCS Commands.

uukick Command

Purpose

Uses debugging mode to contact a specified remote system.

Syntax

uukick [**−x***DebugLevel*] *SystemName*

Description

The **uukick** command contacts a remote system, named by the *SystemName* parameter, using debugging mode. The debugging mode provides a means of monitoring Basic Networking Utilities (BNU) file transfers and connections to remote computers.

The **uukick** command starts the **uucico** daemon, which actually contacts the specified remote system. The **uucico** daemon produces debugging output that enables you to monitor its progress as it establishes the connection to the remote system, performs the remote login, and transfers a file.

The debugging output is scrolled on the screen of the local system. Once the system has finished displaying this information, press the Interrupt key to return to the prompt.

Notes:

1. Either you must be in the **/usr/lib/uucp** directory when you issue the **uukick** command, or you must issue the command with the full path name, **/usr/sbin/uucp/uukick**.
2. The **uukick** command is a shell script stored in the **/usr/lib/uucp** directory.

Flags

−x*DebugLevel* Overrides the default amount of detail in the debugging information the command displays on the screen. The valid range for the *DebugLevel* variable is 0 to 9, with a default of 5. Higher numbers cause the final report to be more detailed. If the **−x** flag is not used, the **uucico** daemon is started with the default level, which produces a moderate amount of information.

Example

To change the amount of detail in the information about the progress of the operation of the **uucico** daemon, use the **−x** flag to specify a higher or lower debugging level. For example, enter:

```
uukick −x9 hera
```

This instructs the **uukick** command to generate as much information as possible about the way in which the

uucico daemon is working while trying to connect to system `hera`. Or, enter:

```
uukick −x3 hera
```

This instructs the command to generate less than the default amount of information about the connection.

Files

/usr/sbin/uucp/uukick	Contains the **uukick** shell script.
/etc/uucp	Contains the configuration files for BNU.
/etc/uucp/Devices	Contains information about available devices.
/etc/uucp/Dialcodes	Contains dialing code abbreviations.
/etc/uucp/Dialers	Specifies initial handshaking on a connection.
/etc/uucp/Permissions	Describes access permissions for remote systems.
/etc/uucp/Systems	Describes accessible remote systems.
/var/spool/uucp/*	Contain files to be transferred and files recording transfer statistics.
/var/spool/uucppublic/*	Contain files that have been transferred.

Related Information

The **tail** command, **uucp** command, **uutry** command, **Uutry** command, **uux** command.

The **uucico** daemon, **uucpd** daemon.

How to Monitor a BNU Remote Connection and How to Monitor a BNU File Transfer in *AIX Version 4.3 System Management Guide: Communications and Networks*.

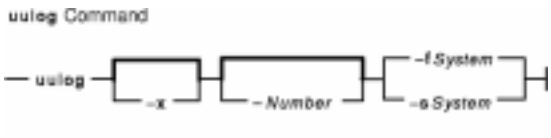
Maintaining BNU in *AIX Version 4.3 System Management Guide: Communications and Networks*.

uulog Command

Purpose

Provides information about BNU file−transfer activities on a system.

Syntax



uulog [**−x**] [*−Number*] [**−f***System* | **−s***System*]

Description

The Basic Networking Utilities (BNU) **uulog** command displays the contents of the log files containing the activities of the **uucico** and **uuxqt** daemons. Individual log files are created for each remote system with which the local system uses the **uucp**, **uuto**, and **uux** commands to communicate.

Use the **uulog** command to display a summary of **uucp**, **uuto**, and **uux** command requests by the user or by the system. All of these transactions are logged in files in the **/var/spool/uucp/.Log** directory. The files are named *DaemonName*/*SystemName* where the *DaemonName* directory is named for the daemon involved and the *SystemName* file is named for the remote system the daemon is contacting.

The **uucp** and **uuto** commands call the **uucico** daemon. The **uucico** daemon's activities are logged in the *SystemName* file in the **/var/spool/uucp/.Log/uucico** directory.

The **uux** command calls the **uuxqt** daemon. The **uuxqt** activities are logged in the *SystemName* file in the **/var/spool/uucp/.Log/uuxqt** directory.

You can examine these individual log files by issuing the **uulog** command directly. However, you can also have the BNU program automatically append these temporary log files to a primary log file that you can then examine. This is called *compacting the log files* and is handled by the **uudemon.cleanu** command, a shell script.

Flags

−f*System* Issues a **tail** command with the **−f** flag on the file transfer log for the specified *System* variable, displaying the end of the log file. Press the Interrupt key to leave the file and return to the prompt.

−s*System* Displays a summary of copy (**uucico** daemon) requests involving the specified system.

Notes:

1. System names can contain only ASCII characters.
2. The **−f** and **−s** flags cannot be combined.

−x Displays the **uuxqt** daemon log file for the given system.

−Number Displays the last lines of the file. The number of lines is determined by the *Number* variable. (To display the lines, the **uulog** command issues a **tail** command with the **−f** flag for the specified number of lines.)

Examples

1. To display the **uucico** log file for system `hera`, enter:

   ```
   uulog -shera
   ```

   The output from the command is similar to the following:

   ```
   uucp hera (10/30-10:18:38,3833,0) SUCCEEDED (call to  hera)
   uucp hera (10/30-10:18:39,3833,0) OK (startup)
   jim hera heraN661d (10/30-10:18:39,3833,0) REQUEST
   (nostromo!D.hera661e6c9 --> hera!X.heraN661d (jim))
   jim hera heraN661d (10/30-10:18:40,3833,0) FAILED (CAN'T
   READ /var/spool/uucp/hera/D.hera661e6c9 13)
   uucp hera (10/30-10:18:41,3833,0) OK (conversation
   complete -8)
   ```

   The preceding lines log a conversation between the local system (`nostromo`) and remote system `hera`. The conversation began at `10:18:38` (a.m.) on October 30th, and ended at `10:18:41`. User `jim` attempted to transfer a data file, `D.hera661e6c9`, to system `hera`. The connection to `hera` was successful, but the file could not be transferred because BNU could not read it.

2. To display the **uuxqt** log file, enter:

   ```
   uulog -x
   ```

3. To display the last forty lines of the file transfer log for system `zeus`, enter:

   ```
   uulog -fzeus -40
   ```

Files

/usr/bin/uulog	Contains the **uulog** command.
/var/spool/uucp/.Log	Contain the BNU log files.

Related Information

The **tail** command, **uucp** command, **uudemon.cleanu** command, **uuto** command, **uux** command.

The **uucico** daemon, **uuxqt** daemon.

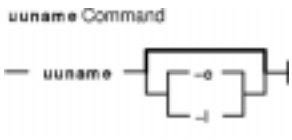
Working with BNU Log Files in *AIX Version 4.3 System Management Guide: Communications and Networks*.

uuname Command

Purpose

Provides information about other systems accessible to the local system.

Syntax



uuname [**−c** | **−l**]

Description

The **uuname** command is a Basic Networking Utilities (BNU) command that displays a list of all the computers networked to the local system. This list of accessible systems is displayed on the screen of the local terminal.

In order for a local system to communicate with a remote system by way of BNU, the remote system must:

- Have a UNIX−based operating system.
- Be connected to the local system. (A telephone line can serve as the connection media.)

BNU can be used to communicate between a workstation and a non−UNIX−based operating system, but such communications may require additional hardware or software. The remote systems accessible with BNU commands are identified when the BNU programs are installed and listed in a BNU **Systems** file (by default, the **/etc/uucp/Systems** file, or one or more files specified in the **/etc/uucp/Sysfiles** file).

Before copying a file to another system with the **uuto** or **uucp** command, issue the **uuname** command to determine the exact name of the remote system.

Flags

−c Displays only the names of systems contained in the **cu Systems** files (configured by the **/etc/uucp/Sysfiles** file). Omission of this flag displays the names of systems contained in the **uucico Systems** files (also configured by the **/etc/uucp/Sysfiles** file). If **/etc/uucp/Sysfiles** is not used to separate **cu** and **uucico** configuration into separate **Systems** files, the names of all systems listed in **/etc/uucp/Systems** are displayed regardless of the **−c** flag.

−l Displays the name of the local system.

Examples

1. To identify the remote systems connected to the local system, enter:

   ```
   uuname
   ```

   The system responds with a list similar to the following:

```
arthur
hera
merlin
zeus
```

2. To identify the name of the local system, enter:

   ```
   uuname -l
   ```

   The system responds with something similar to the following:

   ```
   nostromo
   ```

Files

/usr/bin/uuname	Contains the **uuname** command.
/etc/uucp/Systems	Lists accessible remote systems.
/etc/uucp/Sysfiles	Specifies alternate files to be used as **Systems** files.
/var/spool/uucp	Contains BNU administrative files.
/var/spool/uucppublic	Contains BNU files awaiting transfer (public directory).

Related Information

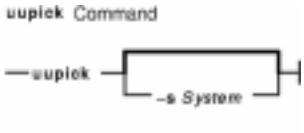
The **ct** command, **cu** command, **uname** command, **uucp** command, **uupick** command, **uustat** command, **uuto** command, **uux** command.

uupick Command

Purpose

Completes the transfer of and handles files sent by the **uuto** command.

Syntax



uupick [−**s***System*]

Description

The **uupick** command is a Basic Networking Utilities (BNU) command that completes the transfer and handles files that the BNU **uuto** command has transmitted to a designated user ID.

Once the copied file is the receive directory, the **rmail** command notifies the recipient that the file has arrived. The recipient then issues the **uupick** command, which searches the public directory on the local system for files sent with some form of the following name:

/var/spool/uucppublic/receive/*User*/*System*/*File*

For each file or directory found, the **uupick** command displays the following message on the screen of the local system:

```
from System: [file File] [dir Directory]
?
```

The question mark prompt (`?`) following the message indicates you can now enter one of the file−handling options.

Flags

−s *System* Searches **/var/spool/uucppublic/receive/***User*/*System* for files sent from the specified system. System names contain only ASCII characters.

File−Handling Options

The question mark prompt (?) following a message indicates that one of the following file−handling options should be entered:

Option	Action
!*Command*	Escapes to a shell to run the specified command. After the command executes, the user is automatically returned to the **uupick** command.
*****	Displays all the file−handling options.
a [*Directory*]	Moves all **uuto** files currently in the **receive** directory into a specified directory on the local

	system. The default is the current working directory. Use a full or relative path name to specify the destination directory.
Ctrl−D	Stops processing and exits from the **uupick** command.
d	Deletes the specified file.
m [*Directory*]	Moves the file to a specified directory. If the *Directory* variable is not specified as a complete path name, a destination relative to the current directory is assumed. If no destination is given, the default is the current working directory on the local system.
new−line	Moves to the next entry in the receive directory when the Enter key is pressed.
p	Displays the contents of the file on the workstation screen.
q	Stops processing and exits from the **uupick** command.

Examples

1. To receive a file sent with the **uuto** command and add it to the current working directory, enter:

   ```
   uupick
   ```

   The system responds with a message similar to:

   ```
   from system anchor: file file1
   ?
   ```

   Enter:

   ```
   a
   ```

   In this example, the `/usr/bin/file1` file sent with the **uuto** command from system `anchor` is added to the current working directory.

2. To receive a file sent with the **uuto** command and add it to a specified directory on your local system, enter:

   ```
   uupick
   ```

   The system responds with a message similar to:

   ```
   from system anchor: file file2
   ?
   ```

   Enter:

   ```
   a /usr/bin1
   ```

   In this example, the `/usr/bin/file2` file sent with the **uuto** command from system `anchor` is added to the `/usr/bin1` directory on the local system.

   > **Note:** The `a /usr/bin1` instruction means move *all* files, not just one. Thus, if any other files are in the `~/anchor/...` directory, they will also be moved.

3. To search for files sent from system `anchor`, enter:

   ```
   uupick −s anchor
   ```

   The system responds with a message similar to:

```
from system anchor: file file1
```

Files

/usr/bin/uupick	Contains the **uupick** command.
/var/spool/uucppublic	Contains the BNU public directory.

Related Information

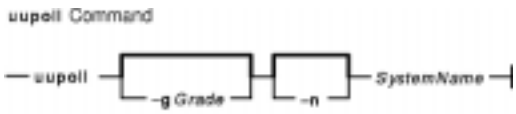
The **ct** command, **cu** command, **uucp** command, **uuname** command, **uustat** command, **uuto** command, **uux** command.

uupoll Command

Purpose

Forces a poll of a remote BNU system.

Syntax



uupoll [**−g***Grade*] [**−n**] *SystemName*

Description

The **uupoll** command forces the Basic Networking Utilities (BNU) to poll the remote system specified by the *SystemName* parameter. The command is usually run by the **cron** daemon or by a user who wants to force a job to be executed immediately. Otherwise, remote systems are polled by the **uudemon.poll** command at times scheduled in the **/etc/uucp/Poll** file and the **/var/spool/cron/crontabs/uucp** file.

Normally, the **uucico** daemon contacts a remote system only at times specified in the **Poll** file or when there is a job queued for that system. The **uupoll** command queues a null job for the remote system and then invokes the **uucico** daemon. This forces the **uucico** daemon to contact the remote system immediately and attempt to send any jobs which are queued for that system. Use the **−g** flag to specify that only high priority jobs be sent.

Use the **−n** flag to queue the null job without starting the **uucico** daemon. Use this option to:

- Queue a null job before invoking the **uucico** daemon for debugging.
- Queue a null job just before the **uucico** daemon is usually invoked, thus forcing the daemon to poll the specified system.

The *SystemName* parameter is required, and specifies the name of the remote system to be polled.

Flags

−g*Grade* Instructs the **uupoll** command to send only jobs of the given grade (specified by the *Grade* parameter) or higher on this call. Jobs of a lower grade will remain in the queue until the next time the remote system is polled.

−n Queues the null job, but does not invoke the **uucico** daemon.

Examples

1. To run the **uupoll** command with the **cron** daemon, place an entry in your **crontabs** file similar to:

   ```
   0  1,7,16  *  *  *  /usr/bin/uupoll hera
   ```

   This polls system `hera` at 0100 hours (1 a.m.), 0700 hours (7 a.m.), and 1600 hours (4 p.m.) daily.

2. If the local system already runs the **uucico** daemon at specific times, you may want to queue a null

job just before the **uucico** daemon normally runs. For example, if your system runs the **uucico** daemon hourly, place an entry similar to the following in your **crontabs** file:

```
0  1,7,16  *  *  *  /usr/bin/uupoll -n zeus
0  5,12,21 *  *  *  /usr/bin/uupoll -n hera
5  *       *  *  *  /usr/sbin/uucp/uucico -r1
```

This queues null jobs for the remote sites on the hour, and they are processed by the **uucico** daemon when it runs at 5 minutes past the hour.

3. To force the **uucico** daemon to transfer all jobs of grade `N` or higher for system `zeus`:

   ```
   uupoll -gN zeus
   ```

Files

/usr/bin/uupoll	Contains the **uupoll** command.
/etc/uucp/Poll	Specifies when the BNU program should poll remote systems to initiate tasks.
/var/spool/cron/crontabs/uucp	Schedules automatic polling of remote systems.
/var/spool/uucp/*SystemName*	Contain files to be transferred to remote systems.

Related Information

The **uucp** command, **uux** command.

The **Uutry** command invokes the **uucico** daemon with debugging turned on.

The **uudemon.poll** and **uudemon.hour** commands perform automatic polling of remote systems as scheduled by the **cron** daemon.

The **uucico** daemon.

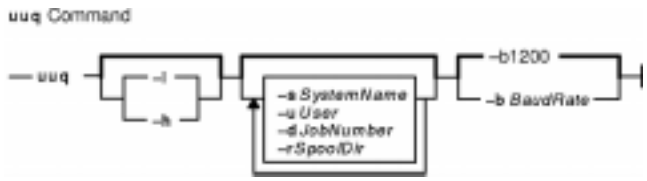
Understanding the BNU Daemons in *AIX Version 4.3 System Management Guide: Communications and Networks*.

uuq Command

Purpose

Displays the BNU job queue and deletes specified jobs from the queue.

Syntax



uuq [**−l** | **−h**] [**−s***SystemName*] [**−u***User*] [**−d***JobNumber*] [**−r***SpoolDir*] [**−b***BaudRate*]

> **Note:** Only a user with root authority can use the **−d** flag.

Description

The **uuq** command is used to list or delete job entries in the Basic Networking Utilities (BNU) job queue.

When listing jobs, the **uuq** command uses a format similar to that used by the **ls** command. In the default format, the **uuq** command lists only the job numbers of the jobs waiting in the queue, followed by a summary line for each system.

In summary format (**uuq −h**) only the summary lines are listed. Summary lines give:

- System name
- Number of jobs for the system
- Total number of bytes to send

In the long format (**uuq −l**), which can be quite slow, the information listed for each job is:

- Job number
- Number of files to transfer
- User who sent the job
- Number of bytes to be sent
- Type of job requested:
  - S Sending a file
  - R Receiving a file
  - X Executing a command on the remote system
- File to be sent or received or the command to be executed

A user with root authority can use the **−d***JobNumber* flag to delete jobs from the queue after running a **uuq** listing to discover the job numbers.

Flags

−b*BaudRate* Uses the baud rate given, instead of the default (1200 baud), to compute the transfer time.

−d *JobNumber* Deletes the job designated by the *JobNumber* variable from the BNU queue. Only someone

	with root authority can delete jobs from the queue.
−h	Shows only the summary lines for each system.
−l	Lists the output in the long format.
−s*SystemName*	Lists only jobs for systems whose system names begin with the string specified in the *SystemName* variable.
−r *SpoolDir*	Searches for files in the spooling directory designated by the *SpoolDir* variable, instead of in the default spooling directory.
−u*User*	Lists only jobs queued by users whose login names begin with the string specified in the *User* variable.

Examples

1. To get a long listing of all jobs spooled for system `hera`, enter:

   ```
   uuq −l −shera
   ```

2. To get a summary listing for all systems, enter:

   ```
   uuq −h
   ```

3. To delete a job for user `nita` from the queue, first use the **uuq** command to find the number of the job you wish to delete, as follows:

   ```
   uuq −l −unita
   ```

   This produces a list of jobs spooled for user `nita`. Find the job you wish to remove. If its job number is `13451`, for example, the following command will delete the job:

   ```
   uuq −d13451
   ```

   Note: You must have root authority or be logged in as **uucp** to delete jobs from the queue.

Files

/usr/bin/uuq	Contains the **uuq** command.
/var/spool/uucp/*SystemName*	Contains spool files for the remote system designated by *SystemName*.
/var/spool/uucp/*SystemName***/C.***	Contain instructions for file transfers.
/var/spool/uucp/*SystemName***/D.***	Contain information about data files to be transferred.
/var/spool/uucp/*SystemName***/X.***	Contain instructions for executing remote commands.

Related Information

The **uucp** command, **uux** command, **uulog** command, **uusnap** command.

Understanding the BNU Daemons, Using BNU Maintenance Commands in *AIX Version 4.3 System Management Guide: Communications and Networks*.

uusched Daemon

Purpose

Schedules work for the Basic Networking Utilities (BNU) file transport program.

Syntax

uusched [**−u***DebugLevel*] [**−x***DebugLevel*]

Description

The **uusched** daemon schedules work for the Basic Networking Utilities (BNU) file transport program. It schedules the transfer of files that are queued in the **/var/spool/uucp/***SystemName* directory. The scheduling daemon first randomizes the work and then starts the **uucico** daemon, which transfers the files.

The **uusched** daemon is usually started by the **uudemon.hour** command, a shell procedure, which is run periodically by the **cron** daemon based on instructions from the **/var/spool/cron/crontabs/uucp** file.

The **uusched** daemon can also be started from the command line for debugging purposes.

> **Note:** Either you must be in the **/usr/sbin/uucp** directory when you start the **uusched** daemon, or you must start the daemon with the full path name, **/usr/sbin/uucp/uusched.**

Flags

−u*DebugLevel* Passes as the **−x***DebugLevel* flag to the **uucico** daemon. The *DebugLevel* variable is a number from 0 to 9, with a default of 5. Higher numbers give more detailed debugging information, which is displayed on the screen of the local system.

−x*DebugLevel* Outputs debugging messages from the **uusched** daemon. The *DebugLevel* variable is a number from 0 to 9, with a default of 5. Higher numbers give more detailed debugging information, which is displayed on the screen of the local system.

Example

To start the **uusched** daemon from the command line, enter:

```
/usr/sbin/uucp/uusched &
```

This starts the **uusched** daemon as a background process. (Note that the path name is included in the command.)

Files

/etc/locks/*	Contains lock files that prevent multiple uses of devices and multiple calls to systems.
/usr/sbin/uucp/*	Contains the **uusched** daemon and the BNU configuration files.
/etc/uucp/Devices	Contains information about available devices.
/etc/uucp/Maxuuscheds	Limits scheduled jobs.
/etc/uucp/Systems	Describes accessible remote systems.
/var/spool/cron/crontabs/uucp	Schedules BNU jobs for the **cron** daemon, including the **uudemon.hour** shell procedure.
/var/spool/uucp/*SystemName* **/***	Contain files waiting to be transferred.

Related Information

The **uucp** command, **uudemon.hour** command, **uustat** command, **uux** command.

The **cron** daemon, **uucico** daemon.

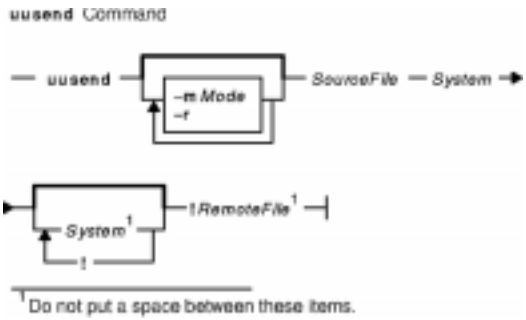
Understanding the BNU Daemons in *AIX Version 4.3 System Management Guide: Communications and Networks*.

uusend Command

Purpose

Sends a file to a remote host.

Syntax



uusend [**−m***Mode*] [**−r**] *SourcefileSystem* [**!***System* ...] **!***RemoteFile*

Description

The **uusend** command sends a file to a given location on a remote system. The remote system need not be directly connected to the local system, but a chain of UUCP links must connect the two systems, and the **uusend** command must be available on each system in the chain.

The chain of systems is given by the *System*[**!***System* ...] parameter, which lists each remote system the file is to be transferred to, separated by ! (exclamation points). The **!***Remotefile* parameter gives the name under which the file is to be stored when it reaches the last system in the chain.

> **Note:** Do not put any spaces between the system names and exclamation points or between the last exclamation point and the remote file name.

The *SourceFile* parameter specifies the name of the file on the local system. If a − (dash) is used, the **uusend** command uses standard input.

Flags

−m *Mode*	Specifies that the mode of the file on the remote system will be taken from the octal number given. If this flag is not specified, the mode of the input file will be used.
−r	Prevents the starting of the **uucico** daemon, which transfers files between systems. The default is to start the **uucico** daemon.

The flags are primarily used internally by the **uusend** command when it is transferring files to the next remote system in the chain.

Example

To send a file across one system to another system, enter:

```
uusend /etc/motd nostromo!gandalf!~nuucp
```

The `/etc/motd` file is sent to system `nostromo` and then to system `gandalf`, and placed in `nuucp`'s home directory, `/var/spool/uucppublic/nuucp`, where `nuucp` is a BNU login ID.

Files

/usr/bin/uusend

Contains the **uusend** command.

Related Information

The **uucp** command, **uux** command.

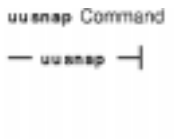
The **uucico** daemon.

uusnap Command

Purpose

Displays the status of BNU contacts with remote systems.

Syntax



uusnap

Description

The **uusnap** command displays a table showing the status of the Basic Networking Utilities (BNU). The table includes the following information for each remote system:

`SystemName`	Specifies the name of the remote system.
`Number Cmds`	Specifies the number of command files (**C.*** files) queued for the remote system.
`Number Data`	Specifies the number of data transfers (**D.*** files) queued for the remote system.
`Number Xqts`	Specifies the number of remote command executions (**X.*** files) queued for the remote system.
`Message`	Specifies the current status message for the site, from the **/var/spool/uucp/.Status/***SystemName* file. The `Message` field may include the time remaining before BNU can retry the remote system, and the count of the number of times (if any) BNU has tried unsuccessfully to reach the system.

Example

To see a snapshot of the status of BNU, enter:

```
uusnap
```

The output from this command is similar to the following:

```
nostromo  4 Cmds  2 Data  2 Xqts  SUCCESSFUL
zeus      2 Cmds  1 Data  2 Xqts  NO DEVICES AVAILABLE
```

These lines indicate that four command files, two data files, and two execute files are currently queued for system `nostromo`. The last connection to `nostromo` was successful. The last attempt to contact system `zeus`, on the other hand, was not successful because no device was available on the local system.

Files

/usr/bin/uusnap	Contains the **uusnap** command.
/var/spool/uucp/.Status/*SystemName*	Records the status of BNU contacts with a remote system.

/var/spool/uucp/*SystemName*	Contains **C.***, **D.***, and **X.*** files to be transferred by the **uucico** daemon.
/var/spool/uucp/*SystemName***/C.***	Instruct BNU about files to be transferred.
/var/spool/uucp/*SystemName***/D.***	Contain files to be transferred by BNU.
/var/spool/uucp/*SystemName***/X.***	Specify commands to be remotely executed by BNU.

Related Information

The **uucp** command, **uux** command, **uuq** command.

The **uucico** daemon.

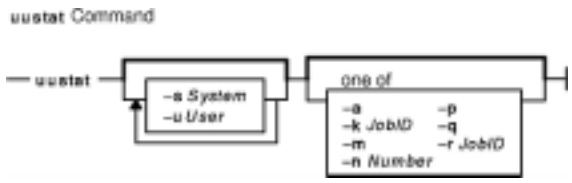
Understanding the BNU File and Directory Structure, Using BNU Maintenance Commands in *AIX Version 4.3 System Management Guide: Communications and Networks*.

uustat Command

Purpose

Reports the status of and provides limited control over BNU operations.

Syntax



uustat [[**−n** *Number*]

[**−a** | **−k** *JobID* | **−m** | **−p** | **−q** | **−r** *JobID*] | [**−s** *System*] [**−u** *User*]]

Description

The **uustat** command is a Basic Networking Utilities (BNU) command that displays status information about several types of BNU operations. It is particularly useful in monitoring the status of BNU requests.

In addition, the **uustat** command also gives a user limited control over BNU jobs queued to run on remote systems. By issuing the command with the appropriate flag, a user can check the general status of BNU connections to other systems and cancel copy requests made with the **uucp** and **uuto** commands.

If the **uustat** command is issued without any flags, the command reports the status of all BNU requests issued by the current user since the last time the holding queue was cleaned up. Such status reports are displayed in the following format:

```
jobid date/time status system_name user_ID size file
```

There are two types of BNU queues:

- The current queue, accessed with the **−q** flag, lists the BNU jobs either queued to run on or currently running on one or more specified computers.
- The holding queue, accessed with the **−a** flag, lists all jobs that have not executed during a set period of time.

  After the time has elapsed, the entries in the holding queue are deleted either manually with the BNU **uucleanup** command or automatically by commands such as **uudemon.cleanu** started by the **cron** daemon.

When sending files to a system that has not been contacted recently, it is a good idea to use the **uustat** command to see when the last access occurred; the remote system may be down or out of service.

Flags

The following flags are mutually exclusive. Use only one at a time with the **uustat** command.

−a Displays information about all the jobs in the holding queue, regardless of the user who issued the original BNU command.

−k*JobID* Cancels the BNU process specified by the *JobID* variable. The person using this flag must either be the one who made the **uucp** request now being canceled or be operating with root authority.

This flag cancels a process only when that job is still on the local computer. Once BNU has moved the job to a remote system for execution, the **−k** *JobID* flag cannot be used to cancel the remote job.

−m Reports the status of the most recent attempt to contact the specified system with a BNU command. If the BNU request was completed, the status report is successful. If the job was not completed, the status report is an error message saying that the login failed.

−n *Number* Allows the user to specify the amount of machines from which to collect BNU status information. The amount specified should be greater than or equal to the amount of machines in the Systems file. The default is 200.

−p Runs a **ps −flp** (process status: full, long list of specified process IDs) for all PID numbers in the lock files.

−q Lists the jobs currently queued to run on each system. These jobs are either waiting to execute or in the process of executing. If a status file exists for the system, its date, time, and status information are reported. Once the job is finished, BNU removes that job listing from the current queue.

In a status report, a number in parentheses next to the number of a **C.*** (command) file or an **X.*** (execute) file represents the age in days of the oldest **C.*** or **X.*** file for that system. The `retry` field represents the number of times BNU tried and failed to execute the command because of, for example, a failed login, locked files, or an unavailable device.

−r*JobID* Marks the files in the holding queue specified by the *JobID* variable with the current date and time. Use this flag to ensure that a cleanup operation does not delete files until the job's modification time reaches the end of the specified period.

You can use either one or both of the following flags with the **uustat** command:

−s *System* Reports the status of BNU requests for the workstation specified by the *System* variable. The *System* name can contain only ASCII characters.

−u*User* Reports the status of BNU requests by the user specified by the *User* variable, for any workstation. The *User* name can contain only ASCII characters.

Examples

1. To display the status of all BNU jobs in the holding queue, enter:

   ```
   uustat −a
   ```

   The system responds with a message similar to the following:

   ```
   heraC3113   11/06−17:47 S  hera    amy  289   D.venus471afd8
   zeusN3130   11/06−09:14 R  zeus    geo  338   D.venus471bc0a
   merlinC3120 11/05−16:02 S  merlin  amy  828   /home/amy/tt
   merlinC3119 11/05−12:32 S  merlin  msg  rmail amy
   ```

Field	Description
1	Job ID of the operation
2	Date and time the BNU command was issued
3	An S or an R, depending on whether the job is to send or receive a file

4	Name of the system on which the command was entered
5	User ID of the person who issued the command
6	Size of the field or the name of the remote command
7	Name of the file.

When the size of the file is given, as in the first three lines of the example output, the file name is also displayed. The file name can be either the name given by the user, as in the `/home/amy/tt` entry, or a name that BNU assigns internally to data files associated with remote executions, such as `D.venus471afd8`.

2. To display the status of all jobs in the current queue, enter:

```
uustat -q
```

The system responds with a message similar to the following:

```
merlin  3C       07/15-11:02   NO DEVICES AVAILABLE
hera    2C       07/15-10:55   SUCCESSFUL
zeus    1C (2)   07/15-10:59   CAN'T ACCESS DEVICE
```

This output tells how many **C.*** (command) files are waiting for each system. The number in parentheses `(2)` in the third line of the example indicates that the **C.*** file has been in the queue for two days.The date and time refer to the current interaction with the system, followed by a report of the status of the interaction.

3. To display all process IDs in the lock file, enter:

```
uustat -p
```

The system responds with a message similar to the following:

```
LCK..tty0:  881
LCK.S.0:  879
LCK..hera:  881
F   S UID  PID  PPID  C  PRI NI ADDR SZ  WCHAN    STIME    TTY
101 S uucp 881  879   26 39  39 370  296 3fffe800 09:57:03  -
TIME  COMD
0:00  UUCICO  -r1  -shera
101 S uuc  879  1     11 33  39 770  156 8d874    09:57:02  -
0:00  /usr/sbin/uucp/uusched
```

4. To cancel a job in the current queue, first determine its job ID and then issue the command to cancel the job. To determine the job ID, enter:

```
uustat -a
```

The system responds with a message similar to the following:

```
heraC3113   11/06-17:47  S  hera    amy  289  D.venus471afd8
merlinC3119 11/06-17:49  S  merlin  geo  338  D.venus471bc0a
```

To cancel the job with the ID of `heraC3113`, enter:

```
uustat -k heraC3113
```

5. To report the status of jobs requested by system `hera`, enter:

   ```
   uustat -s hera
   ```

   The system responds with a message similar to the following:

   ```
   heraN1bd7  07/15-12:09  S  hera  amy  522    /usr/amy/A
   heraC1bd8  07/15-12:10  S  hera  amy   59    D.3b2a12ce4924
   heraC3119  07/15-12:11  S  hera  amy  rmail   msg
   ```

Files

/etc/locks	Contains lock files to prevent multiple uses of devices.
/usr/bin/uustat	Specifies the command pathname.
/var/spool/uucp	Contains BNU status information.

Related Information

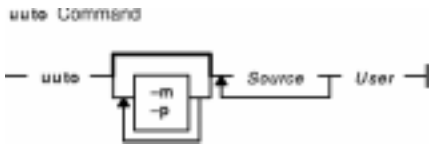
The **cron** daemon, **ct** command, **cu** command, **echo** command, **stty** command, **uucleanup** command, **uucp** command, **uuname** command, **uupick** command, **uuto** command, **uux** command.

uuto Command

Purpose

Copies files from one system to another.

Syntax



uuto [**−m**] [**−p**] *Source ... User*

Description

The **uuto** command is a Basic Networking Utilities (BNU) command that copies one or more *Source* files from one system to a specified *User* on another UNIX based system. This program uses the **uucp** command for the actual file transfer, but the **uuto** command enables the recipient to use the **uupick** command options to handle the transferred file on the local system.

The sender issues the **uuto** command to copy one or more files to a specific user ID on another system. The **uucp** command then copies the file to the BNU public directory, **/var/spool/uucppublic**, on the destination system. The **uucp** command also creates an additional subdirectory called **receive** (if it does not already exist) and directories below it in which to hold the files until the recipient retrieves them with the **uupick** command. The full path names to the copied files are some form of the following name:

/var/spool/uucppublic/receive/*UserName*/*System*/*File*

where the *UserName* and *System* directories are created based on the *User* parameter given with the **uuto** command.

Once the copied file is in the **receive** directory, the **rmail** command notifies the recipient that a file has arrived. The recipient then issues the **uupick** command, and this command searches the public directory for files sent to the recipient and notifies the recipient about each file it locates. The recipient then enters one of the **uupick** options to handle the file.

Source and Destination File Names

The sender must give the name of the file to be sent and user and system to which the file is to be transferred. The *Source* parameter is the path name of the source file. This can be the name of the file if the file is in the directory from which the **uuto** command is issued. If the file is in a different directory, the complete or relative path name of the file must be given.

The *User* parameter is the path name to the specific location where the source file is to be copied. This path name must include the user identification of the person the file is being sent to. The *User* parameter has the form:

System!*UserName*

where *System* is the name of the remote system connected to the local system, and *UserName* is the login name of the recipient of the transferred files on the specified system.

When copying a file from one user to another user on the local system, omit the *System* entry; the destination is the ID of the user to whom the file is being sent. System names can contain only ASCII characters.

Flags

−m Notifies the sender by the **bellmail** command when the source file has been successfully copied.

−p Copies the source file to the spool directory on the local system. The source file resides in the spooling directory for a set period of time (defined in the **uusched** program) before the **uucp** command calls the **uucico** daemon, which actually transfers the copy to the public directory on the specified remote system. The default is to transfer a source file directly to the specified user.

Examples

1. To copy a file to a user on a remote system, enter:

   ```
   uuto /home/bin/file1 zeus!karen
   ```

   In this example, the `/home/bin/file1` file is sent to user `karen` on the remote system `zeus.`

2. To copy a file to a user on a remote system and be notified whether the source file was successfully copied, enter:

   ```
   uuto −m /home/bin/file2 zeus!karen
   ```

   In this example, the `/home/bin/file2` file is sent to user `karen` on the remote system `zeus` and a message is returned to the sender verifying that the copy was successful.

3. To copy a file to another user on your local system, enter:

   ```
   uuto /home/bin/file3 ron
   ```

In this example, the `/home/bin/file3` file is sent to user `ron` on the local system. No mail message is sent to the recipient in a local transfer.

Files

/usr/bin/uuto	Contains the **uuto** command.
/var/spool/uucppublic	Is the BNU public directory.

Related Information

The **bellmail** command, **ct** command, **cu** command, **uucp** command, **uuname** command, **uupick** command **uustat** command, **uux** command.

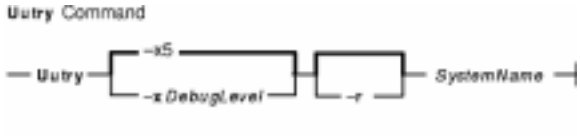
The **uucico** daemon, **uusched** daemon.

Uutry Command

Purpose

Contacts a specified remote system with debugging turned on and saves the debugging output in a temporary file.

Syntax



Uutry [**−x***DebugLevel*] [**−r**] *SystemName*

Description

The **Uutry** command contacts a remote system, specified by the *SystemName* parameter, using debugging mode. Debugging mode provides a means of monitoring Basic Networking Utilities (BNU) connections to remote computers and file transfers.

The **Uutry** command starts the **uucico** daemon, which actually contacts the specified system. The **uucico** daemon produces debugging output that enables you to monitor the daemon's progress as it establishes the connection to the remote system, performs the remote login, and transfers a file.

The debugging output is scrolled on the screen of the local system. Once the system has finished displaying this information, press the Interrupt key to return to the prompt.

In addition to displaying the debugging output on the screen, the **Uutry** command directs this information to a file named **/tmp**/*SystemName*, where the *SystemName* parameter is the name of the remote system you are attempting to contact. Again, when the last of the output has been displayed, press the Interrupt key to return to the prompt.

The *SystemName* parameter, which is required, specifies the name of the remote system you wish to contact.

Notes:

1. Press the Interrupt key while the system is scrolling the output generated by the **Uutry** command to return to the prompt. The **uucico** daemon continues to place the debugging information in the **/tmp**/*SystemName* file.
2. Either you must be in the **/usr/sbin/uucp** directory when you issue the **Uutry** command or you must issue the command with the full path name, **/usr/sbin/uucp/Uutry**.
3. The **Uutry** command is a shell script stored in the **/usr/sbin/uucp** directory.

Flags

−r Overrides the default retry time. If for some reason the **uucico** daemon cannot complete the requested connection, the daemon waits for a set amount of time and tries again. The default retry time is 5 minutes.

Note: The time the remote system was last polled is recorded in the

/var/spool/uucp/.Status/*SystemName* file.

−x*DebugLevel* Overrides the default amount of detail in the debugging information that the command displays on the screen. The valid range for the *DebugLevel* variable is 0 to 9, with a default of 5. Higher numbers cause the final report to be more detailed. If the **−x** flag is not used, the **uucico** daemon is started with the default level, which produces a moderate amount of information.

Example

To change the amount of detail the **Uutry** command provides about the progress of the **uucico** operation, use the **−x** flag to specify a different debugging level. For example, entering:

```
/usr/sbin/uucp/Uutry −x9 venus
```

instructs the **Uutry** command to generate as much information as possible about the way in which the **uucico** daemon is working.

Files

/tmp/*SystemName*	Contains debugging output from the **Uutry** command (temporary file).
/usr/sbin/uucp/Uutry	Contains the **Uutry** command and all the configuration files for BNU.
/etc/uucp/Devices	Contains information about available devices.
/etc/uucp/Dialcodes	Contains dial−code abbreviations.
/etc/uucp/Dialers	Specifies initial handshaking on a connection.
/etc/uucp/Permissions	Describes access permissions for remote systems.
/etc/uucp/Systems	Describes accessible remote systems.
/var/spool/uucp/.Status/*SystemName* file	Lists the last time a remote system was contacted.
/var/spool/uucppublic/*	Contain the BNU public directories.

Related Information

The **uucico** daemon, **uucpd** daemon.

The **tail** command, **uucp** command, **uustat** command, **uutry** command, **uukick** command, **uux** command.

How to Monitor a BNU Remote Connection and How to Monitor a BNU File Transfer in *AIX Version 4.3 System Management Guide: Communications and Networks*.

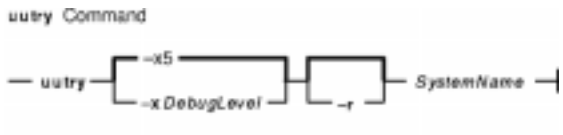
Maintaining BNU, Understanding the BNU Daemons in *AIX Version 4.3 System Management Guide: Communications and Networks*.

uutry Command

Purpose

Contacts a specified remote system with debugging turned on and allows the user to override the default retry time.

Syntax



uutry [**−x***DebugLevel*] [**−r**] *SystemName*

Description

The **uutry** command contacts a remote system, specified by the *SystemName* parameter, using debugging mode. Debugging mode provides a means of monitoring Basic Networking Utilities (BNU) connections to remote computers and file transfers. The **uutry** command calls the **uucico** daemon to contact the remote system.

The debugging output is scrolled on the screen of the local system. Once the system has finished displaying this information, press the Interrupt key to return to the prompt.

The **−r** flag overrides the default retry time if the first attempt to contact the remote system is unsuccessful. The default retry time is 5 minutes.

The *SystemName* parameter, which is required, specifies the name of the remote system you wish to contact.

Notes:

1. Either you must be in the **/usr/sbin/uucp** directory when you issue the **uutry** command or you must issue the command with the full path name, **/usr/sbin/uucp/uutry.**
2. The **uutry** command is a shell script stored in the **/usr/lib/uucp** directory.
3. If the debugging output scrolls too quickly to be read, use the **Uutry** command to save the output in a temporary file.

Flags

−r Overrides the default retry time. If for some reason the **uucico** daemon cannot complete the requested connection, the daemon waits for a set amount of time and tries again. The default retry time is 5 minutes.

Note: The time at which the remote system was last polled is recorded in the *SystemName* file in the **/var/spool/uucp/.Status** directory.

−x*DebugLevel* Overrides the default amount of detail in the debugging information that the **uutry** command

displays on the screen. The valid range for the *DebugLevel* variable is 0 to 9, with a default of 5. Higher numbers cause the final report to be more detailed. If the **−x** flag is not used, the **uucico** daemon is started with the default level, which produces a moderate amount of information.

Examples

1. To change the amount of detail the **uutry** command provides about the progress of the **uucico** operation, use the **−x** flag to specify a different debugging level. For example, entering:

   ```
   /usr/sbin/uucp/uutry −x9 venus
   ```

   instructs the **uutry** command to generate as much information as possible about the way in which the **uucico** daemon is working.

2. The default time at which to retry a contact to a remote system when the first contact was unsuccessful is 5 minutes. To shorten the default retry time for contacting the remote system, enter:

   ```
   /usr/sbin/uucp/uutry −r venus
   ```

   Using the **−r** flag instructs the **uucico** daemon to contact remote system `venus`, overriding the default retry time. The daemon attempts to contact system `venus`, retrying periodically until the connection is successful, and then produces debugging output on the display screen of the local system.

Files

/usr/sbin/uucp/uutry	Contains the **uutry** command.
/etc/uucp/Devices	Contains information about available devices.
/etc/uucp/Dialcodes	Contains dial−code abbreviations.
/etc/uucp/Dialers	Specifies initial handshaking on a connection.
/etc/uucp/Permissions	Describes access permissions for remote systems.
/etc/uucp/Systems	Describes accessible remote systems.
/var/spool/uucp/.Status/*SystemName*	Lists the last time the remote system named by the *SystemName* file was contacted.
/var/spool/uucppublic/*	Contain the BNU public directories.

Related Information

The **tail** command, **uucp** command, **Uutry** command, **uukick** command, **uux** command.

The **uucico** daemon, **uucpd** daemon.

How to Monitor a BNU Remote Connection and How to Monitor a BNU File Transfer in *AIX Version 4.3 System Management Guide: Communications and Networks.*

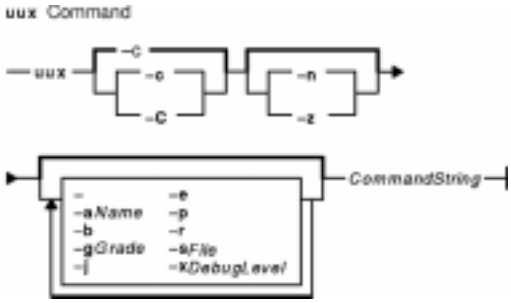
Maintaining BNU, Understanding the BNU Daemons in *AIX Version 4.3 System Management Guide: Communications and Networks.*

uux Command

Purpose

Runs a command on another UNIX−based system.

Syntax



uux [**−c** | **−C**] [**−n** | **−z**] [**−**] [**−a***Name*] [**−b**] [**−g***Grade*] [**−j**] [**−p**] [**−e**] [**−r**] [**−s***File*] [**−x***DebugLevel*] *CommandString*

Description

The **uux** command is a Basic Networking Utility (BNU) that runs a specified command on a specified UNIX−based system while enabling the user to continue working on the local system. Before running the requested command, the **uux** command gathers any necessary files from the designated systems. The user can direct the output from the command to a specific file on a specific system. For security reasons, many installations permit the **uux** command to run only the **rmail** command.

The **uux** commands on other systems create execute (**X.***) files that run commands on the local system. In addition, the **uux** command on the local system creates both command (**C.***) files and data (**D.***) files for transfer to other systems. Execute files contain the command string to be executed on the designated system. Command files contain the same information as those created by the **uucp** command. Data files either contain the data for a remote command execution or else become **X.*** files on remote systems for remote command executions.

The full path name of an execute file is a form of the following:

/var/spool/uucp/*System***/X.***SystemNxxxx*

After creating the files in the spooling directory, the **uux** command calls the **uucico** daemon to transfer the files from the spooling directory on the local system to the designated remote system. Once the files are transferred, the **uuxqt** daemon on the remote system executes the *CommandString* on the specified system, placing any output from the command in the file designated by the original **uux** command request.

The *CommandString* argument is made up of one or more arguments that look like an operating system command line, except that *CommandString* argument may be prefixed by the name of the remote system in the form *System***!**. The default *System* is the local system. Unless the user entering the **uux** command includes the **−n** flag, the command notifies that user if the remote system does not run the command. This response comes by mail from the remote system.

Source and Destination File Names

- When specifying the destination of the output of a command, the **uux** command can be entered in either one of the following formats:
  - **uux** [*Options*] "*CommandString*> *Destination*"
  - **uux** [*Options*] *CommandString*\ *{Destination*\}.
- Destination names can be either of the following:
  - A full path name
  - A full path name preceded by **~***User*, where *User* is a login name on the specified system. The **uux** command replaces this path name with the user's login directory.
- The shell pattern−matching characters ? (question mark), * (asterisk), and [...] (brackets) can be used in the path name of a source file (such as files compared by the **diff** command); the appropriate system expands them. However, using the * character may occasionally produce unpredictable or unanticipated results. Shell pattern−matching characters should not be used in the destination path name.
- Place either two backslashes (\ . . . \) or a pair of quotation marks (" . . . ") around pattern−matching characters in a path name so the local shell cannot interpret them before the **uux** command sends the command to a designated system.
- If you are using the special shell characters > (greater than), < (less than), ; (semicolon), or | (vertical bar) in a path name, place either \ . . . \ or " . . . " around the individual character or around the entire command string.
- Do not use the shell redirection characters << or >> in a path name.
- The **uux** command attempts to move all files specified on the command line to the designated system. Enclose the names of all output files in parentheses so that the **uux** command does not try to transfer them.
- When specifying a *System,* always place it before the *CommandString* argument in the entry. System names can contain only ASCII characters.
- The ! (exclamation point) preceding the name of the local system in a command is optional. If you choose to include the ! to run a command on the local system using files from two different remote systems, use ! instead of *System*! to represent the local system, and add *System*! as the first entry in any path name on the remote systems.
- The exclamation point representing a system in BNU syntax has a different meaning in C shells. When running the **uux** command in a C shell, place a \ (backslash) before the exclamation point in a system name.

  Note: The notation ~ (tilde) is the shorthand way of specifying the public spooling directory, **/var/spool/uucppublic**.

Flags

− Makes the standard input to the **uux** command the standard input to the *CommandString* argument.

−a*Name* Replaces the user ID of the person issuing the command with the user ID specified with the *Name* variable.

−b Returns standard input to the command if the exit status is not zero.

−c Transfers the source files to the destination on the specified system. The source files are copied into the spooling directory, and the **uucico** daemon is invoked immediately. This flag is the default.

−C Transfers the source files to the spool directory. After a set period of time (specified in the **uusched** program), the **uucico** daemon attempts to transfer the files to the destination on the specified computer.

Occasionally, there are problems in transferring a source file; for example, the remote computer may not be working or the login attempt may fail. In such cases, the file remains in the spool directory until it is either transferred successfully or removed by the **uucleanup** command.

−e Enables file expansion.

−g*Grade*	Specifies when the files are to be transmitted during a particular connection. The *Grade* variable specifies a single number (0 through 9) or letter (A through Z, a through z); lower ASCII−sequence characters cause the files to be transmitted earlier than do higher sequence characters. The number 0 is the highest (earliest) grade; z is the lowest (latest). The default is **N**.
−j	Displays the job identification number of the process that is running the command on the specified system. Use this job ID with the BNU **uustat** command to check the status of the command or with the **uustat −k** flag to terminate the process.
−n	Prevents user notification by the **mail** command of the success or failure of a command. The default is to notify the user if the command fails.
−p	Uses the standard input to the **uux** command as the standard input to the *CommandString* argument. A − (minus) has the same effect.
−r	Prevents the starting of the spooling program that transfers files between systems. The default is to start the spooling program.
−s*File*	Reports the status of the transfer in a file specified by the *File* variable on the designated system. File names can contain only ASCII characters.
−x*DebugLevel*	Displays debugging information on the screen of the local system. The *DebugLevel* variable must be a number from 0 to 9. A higher number gives a more detailed report.
−z	Notifies the user if the command completes successfully. This flag is the opposite of the system default, which is to notify the user only in the event of a failure.

Examples

1. To run the **qprt** command on a remote system, enter:

   ```
   uux merlin!qprt /reports/memos/lance
   ```

   In this example, the remote file `/reports/memos/lance` is printed on remote system `merlin`. Since neither the **−n** nor **−z** flag is specified, the **uux** command notifies the user only if the remote system fails to run the command. The response comes by the **mail** command from the remote system.

2. To run commands on two remote systems, enter the information on separate command lines:

   ```
   uux merlin!qprt /reports/memos/lance
   uux zeus!qprt /test/examples/examp1
   ```

   In this example, the remote `/reports/memos/lance` file is printed on remote system `merlin`, and the remote `/test/examples/examp1` file is printed on remote system `zeus`. Since neither the **−n** nor **−z** flag is specified, the **uux** command notifies the user only if the remote system fails to run the command. The response comes by the **mail** command from the remote system.

3. To queue a job that compares a file on the local system with a file on a remote system, using the **diff** command on the local system, and get the job ID of the job, enter:

   ```
   uux -j "/usr/bin/diff /usr/amy/f1 hera!/home/amy/f2 > ~/f1.diff"
   ```

   In this example, the `/usr/amy/f1` file on the local system is compared to the `/home/amy/f2` file on the remote system `hera` and the output is placed in the `f1.diff` file in the local public directory (the full path name of this file is `/var/spool/uucppublic/f1.diff`). The destination name must be entered either preceded by a > with the whole command string enclosed in `" "` (quotation marks) or entered enclosed in braces and backslashes, as **\{** *DestinationName* **\}**. The **−j** flag causes the **uux** command to return the BNU job ID of the job.

4. To use the **diff** command on the local system to compare files that are located on two different remote systems, enter:

```
uux "!/usr/bin/diff hera!/usr/amy/f1 venus!/home/amy/f2 > \ !f1.diff"
```

In this example, the `/usr/amy/f1` file from the remote system `hera` is compared to the `/home/amy/f2` file from the remote system `venus` and the output is placed in the file `f1.diff`, located in the current working directory on the local system.

The output file must be write−enabled. If you are uncertain about the permission status of a specific target output file, direct the results to the public directory. The exclamation points representing the local system are optional. The destination name must be entered either preceded by a > with the whole command string enclosed in " " (quotation marks) or entered enclosed in braces and backslashes, as **\{** *DestinationName* **\}**.

5. To execute the **diff** command on two separate files from different systems, enter:

   ```
   uux "hera!/usr/bin/diff /tmp/out1 zeus/tmp/out2 > ~/DF"
   ```

   In this example, the `diff` file is on the remote system `hera`. The first source file is on the remote system `hera`, and the secondfile is on the system `zeus`. (`zeus` may be the local system or another remote system.) The output is directed to the file DF in the public directory on the local system.

6. To specify an output file on a different remote system, enter:

   ```
   uux hera!uucp venus!/home/amy/f1 \{merlin!/home/geo/test\}
   ```

   In this example, the **uucp** command is run on the remote system `hera`, and the `/home/amy/f1` file, stored on system `venus`, is sent to user `geo` on system `merlin` as `test`. The destination name is entered enclosed in braces and backslashes.

7. To get selected fields from a file on a remote system and place them in a file on the local system, enter:

   ```
   uux "cut −f1 −d: hera\!/etc/passwd > ~/passw.cut"
   ```

   In this example, the **cut** command is run on the local system. The first field from each line of the password file on system `hera` is placed in the `passw.cut` file in the public directory on the local system. The **uux** command is running in a C shell, so a \ (backslash) must precede the exclamation point in the name of the remote system.

8. To use the **uux** piping option to specify a remote copy of the `/tmp/example` file to `/tmp/examplecopy` on system `mercury` use the following syntax:

   ```
   uux −p mercury!
   cp /tmp/example /tmp/examplecopy
   ```

The user must enter a Ctrl−D in order to terminate the command input. After Ctrl−D is pressed, the command will be spooled for remote execution on system `mercury`.

Files

/usr/bin/uux	Contains the **uux** command.
/var/spool/uucp	Is the spooling directory.
/var/spool/uucppublic	Is the public directory.

Related Information

The **ct** command, **cu** command, **mail** command, **rmail** command, **uucleanup** command, **uucp** command, **uuname** command, **uupick** command, **uustat** command, **uuto** command, **qprt** command.

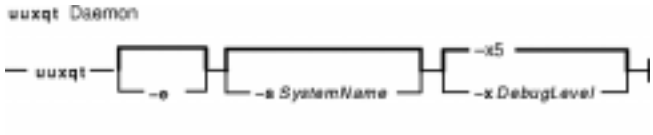
The **sendmail** daemon, **uucico** daemon, **uuxqt** daemon.

uuxqt Daemon

Purpose

Executes Basic Networking Utilities (BNU) remote command requests.

Syntax



uuxqt [**−e**] [**−s***SystemName*] [**−x***DebugLevel*]

Description

The Basic Networking Utilities (BNU) **uuxqt** daemon executes commands on designated remote systems.

The **uuxqt** daemon on each networked system periodically searches the spool directory for remote execute (**X.***) files. These files are sent to the directory by the **uucico** daemon in response to a **uux** command.

When it finds **X.*** files, the **uuxqt** daemon checks each file to make sure that:

- All the required data (**D.***) files are available.
- The requesting system has the necessary permissions to access the data files and run the requested commands.

  Note: The **uuxqt** daemon uses the **/etc/uucp/Permissions** file to validate file accessibility and command execution permission.

If the data files are present and the requesting system has the appropriate permissions, the **uuxqt** daemon executes the commands.

> **Note:** The **uuxqt** command is usually executed from the **uudemon.hour** command, a shell procedure, and not entered from the command line. You must have root user privileges to issue the **uuxqt** command from the command line.

Flags

−e	Enables file expansion.
−s*SystemName*	Designates the remote system to be contacted. Use only when starting the **uuxqt** command manually. The system name is supplied internally when the **uuxqt** command is started automatically. **Note:** System names can contain only ASCII characters.
−x*DebugLevel*	Displays debugging information on the screen of the local system. The *DebugLevel* variable is a single digit between 0 and 9, with a default of 5. The higher the *DebugLevel* variable, the more detailed the debugging information.

Security

Access Control: You must have root authority to start the **uuxqt** daemon from the command line.

Example

To start the **uuxqt** daemon for debugging, enter:

```
/usr/sbin/uucp/uuxqt -svenus -x7
```

This instructs the command to contact remote system `venus` and provide fairly detailed information about the contact.

Files

/usr/sbin/uucp/uuxqt	Contains the **uuxqt** daemon.
/etc/locks	Contains lock files that prevent multiple uses of devices and multiple calls to systems.
/etc/uucp/Maxuuxqts	Limits remote command executions.
/etc/uucp/Permissions	Describes access permissions for remote systems.
/var/spool/uucp/*	Contain the execute and data files.

Related Information

The **uucp** command, **uudemon.hour** command, **uustat** command, **uux** command.

The **cron** daemon, **uucico** daemon.

Understanding the BNU File and Directory Structure, Understanding the BNU Daemons, Understanding BNU Security in *AIX Version 4.3 System Management Guide: Communications and Networks*.

Vos remarques sur ce document / Technical publication remark form

Titre / Title : Bull AIX Commands Reference Vol.5 ruutry to uux

Nº Reférence / Reference Nº : 86 A2 42JX 02

Daté / Dated : April 2000

ERREURS DETECTEES / ERRORS IN PUBLICATION

AMELIORATIONS SUGGEREES / SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Vos remarques et suggestions seront examinées attentivement.
Si vous désirez une réponse écrite, veuillez indiquer ci-après votre adresse postale complète.

Your comments will be promptly investigated by qualified technical personnel and action will be taken as required.
If you require a written reply, please furnish your complete mailing address below.

NOM / NAME : _____ Date : _____

SOCIETE / COMPANY : _____

ADRESSE / ADDRESS : _____

_____

Remettez cet imprimé à un responsable BULL ou envoyez-le directement à :

Please give this technical publication remark form to your BULL representative or mail to:

BULL ELECTRONICS ANGERS
CEDOC
34 Rue du Nid de Pie – BP 428
49004 ANGERS CEDEX 01
FRANCE

Technical Publications Ordering Form

Bon de Commande de Documents Techniques

To order additional publications, please fill up a copy of this form and send it via mail to:
Pour commander des documents techniques, remplissez une copie de ce formulaire et envoyez-la à :

BULL ELECTRONICS ANGERS
CEDOC
ATTN / MME DUMOULIN
34 Rue du Nid de Pie – BP 428
49004 ANGERS CEDEX 01
FRANCE

Managers / Gestionnaires :
Mrs. / Mme : **C. DUMOULIN** +33 (0) 2 41 73 76 65
Mr. / M : **L. CHERUBIN** +33 (0) 2 41 73 63 96

FAX : +33 (0) 2 41 73 60 19
E–Mail / Courrier Electronique : srv.Cedoc@franp.bull.fr

Or visit our web site at: / Ou visitez notre site web à:
`http://www–frec.bull.com` (PUBLICATIONS, Technical Literature, Ordering Form)

CEDOC Reference # Nº Référence CEDOC	**Qty** Qté	**CEDOC Reference #** Nº Référence CEDOC	**Qty** Qté	**CEDOC Reference #** Nº Référence CEDOC	**Qty** Qté
_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]	
_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]	
_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]	
_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]	
_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]	
_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]	
_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]	
[_ _] : **no revision number means latest revision** / pas de numéro de révision signifie révision la plus récente					

NOM / NAME : ____________________ Date : ________

SOCIETE / COMPANY : ____________________

ADRESSE / ADDRESS : ____________________

____

PHONE / TELEPHONE : ____________________ FAX : ____________________

E–MAIL : ____________________

For Bull Subsidiaries / Pour les Filiales Bull :
Identification: ____________________

For Bull Affiliated Customers / Pour les Clients Affiliés Bull :
Customer Code / Code Client : ____________________

For Bull Internal Customers / Pour les Clients Internes Bull :
Budgetary Section / Section Budgétaire : ____________________

For Others / Pour les Autres :
Please ask your Bull representative. / Merci de demander à votre contact Bull.

PLACE BAR CODE IN LOWER LEFT CORNER

BULL ELECTRONICS ANGERS
CEDOC
34 Rue du Nid de Pie – BP 428
49004 ANGERS CEDEX 01
FRANCE

ORDER REFERENCE
86 A2 42JX 02

Bull

Utiliser les marques de découpe pour obtenir les étiquettes.
Use the cut marks to get the labels.

AIX
AIX Commands
Reference Vol.5
ruutry to uux

86 A2 42JX 02

AIX
AIX Commands
Reference Vol.5
ruutry to uux

86 A2 42JX 02

AIX
AIX Commands
Reference Vol.5
ruutry to uux

86 A2 42JX 02