**Chapter 3**

**Additional Commands in UNIX**

**1. uname**

By default, the uname utility will write the operating system characteristics to standard output. When options are specified, symbols representing one or more system characteristics shall be written to the standard output. The format and contents of the symbols are implementation-defined.

**Syntax**

**$ uname [-a] [-i] [-n] [-p] [-r] [-v]**

|  |  |
| --- | --- |
| -a | Print basic information currently available from the system. |
| -i | Print the name of the hardware implementation (platform). |
| -n | Print the nodename (the nodename is the name by which the system is known to a communications network). |
| -r | Print the operating system release level. |
| -v | Print the operating system version. |
| -p | Print the current host's ISA or processor type. |

**Example**

**$ uname –n**

**<hostname>**

**2. mv**

mv (short for move) is a Unix command that moves a file from one place to another. The original file is deleted, and the new file may have the same or a different name.

**Syntax**

**$ mv [-f] [-i] oldname newname**

|  |  |
| --- | --- |
| -f | mv will move the file(s) without prompting even if it is writing over an existing target. Note that this is the default if the standard input is not a terminal. |
| -i | Prompts before overwriting another file. |
| oldname | The oldname of the file renaming. |
| newname | The newname of the file renaming. |
| filename | The name of the file we want to move directory - The directory of were we want the file to go. |

**Examples**

**$ mv myfile.txt newdirectory/**

moves the file myfile.txt to the directory newdirectory.

**$ mv myfile.txt ../**

moves the file myfile.txt back one directory (if available).

**3. date**

The date command can be used to display or set the date. If a user has superuser privileges, he or she can set the date by supplying a numeric string with the following command:

Fortunately there are options to manipulate the format. The format option is preceded by a + followed by any number of field descriptors indicated by a % followed by a character to indicate which field is desired. The allowed field descriptors are:

|  |  |
| --- | --- |
| Option | Meaning |
| %n | A newline |
| %t | A tab |
| %m | Month of year (01-12) |
| %d | Day of month (01-31) |
| %y | Last two digits of year (00-99) |
| %D | Date as mm/dd/yy |
| %H | Hour (00-23) |
| %M | Minute (00-59) |
| %S | Second (00-59) |
| %T | Time as HH:MM:SS |
| %j | Day of Year (001-366) |
| %w | Day of week (0-6) Sunday is 0 |
| %a | Abbreviated weekday (Sun-Sat) |
| %h | Abbreviated month (Jan-Dec) |
| %r | 12-hour time w/ AM/PM |

**Examples**

**$ date**

**Mon Jan 6 16:07:23 PST 1997**

**$ date '+%a %h %d %T %y'**

**Mon Jan 06 16:07:23 97**

**$ date '+%a %h %d %n %T %y'**

**Mon Jan 06**

**16:07:23 97**

**3.1. Set Date and Time**

**date [-s datestr]**

|  |  |
| --- | --- |
| **-s datestr** | Sets the time and date to the value specified in the datestr. The datestr may contain the month names, timezones, 'am', 'pm', etc. See examples for an example of how the date and time can be set. |

**Examples**

$ date -s "11/20/2003 12:48:00"

Set the date to the date and time shown.

**4. Removing Files and Directories**

**4.1. rm**

Deletes a file without confirmation (by default).

The options must start with ‘-‘. One or more filenames can be specified, and wildcards are permitted (because the shell, not rm, expands them).

If we are not familiar with wildcards (\* and ? to name the most dangerous), read up on them. Placing a wildcard character in the file name list by accident can make we a very unhappy camper

**Syntax**

**rm [-f] [-i] [-R] [-r] [filenames | directory]**

|  |  |
| --- | --- |
| -f | Remove all files (whether write-protected or not) in a directory without prompting the user. In a write-protected directory, however, files are never removed (whatever their permissions are), but no messages are displayed. If the removal of a write-protected directory is attempted, this option will notsuppress an error message. |
| -i | Interactive. With this option, rm prompts for confirmation before removing any files. It over- rides the –f option and remains in effect even if the standard input is not a terminal. |
| filenames | A path of a filename to be removed. |

**Examples**

To remove the file myfile.txt without prompting the user.

**$ rm myfile.txt**

A number of files can all be removed at the same time, Here we remove all the files with the lis extension.

**$ rm \*.lis**

**ultimateanswer.lis : ?y**

**lessultimateanswer.lis : ?n**

**moreultimateanswer.lis : …**

Remove can also be used without asking for a confirmation. To do this uses the -force option. Here we remove all the files with the lis extension, without asking for a confirmation.

**$ rm -f \*.lis**

Beware when using the -f option we can easily remove all files from a directory by accident! To remove all files without asking for a confirmation.

**$ rm -f \***

**4.2. rmdir**

Used to remove the directories form the system

**Syntax**

**rmdir [-p] [-s][-r] directoryname**

|  |  |
| --- | --- |
| -p | Allow users to remove the directory dirname and its parent directories which become empty. A message is printed to standard error if all or part of the path could not be removed. |
| -s | Suppress the message printed on the standard error when -p is in effect. |
| -r | Delete nonempty directory |
| directory | The name of the directory that we wish to delete. |

**Examples**

removes the directory mydir

**$ rmdir mydir**

To deletes all directories in the current directory whose directory names begins with the characters "index".

**$ rmdir index\***

To remove a directory, even if files existed in that directory.

**$ rm -r directory**

To delete the directory named "new-novel". This directory, and all of it’s contents, are erased from the disk, including any sub-directories and files.

**$ rm -r new-novel**

**5. ls**

The ls command lists the files in your current working directory. When we log onto your account on UNIX, your current working directory is your home or personal directory. This is the directory in which we have personal disk space to put files on or to create sub-directories under. The ls command also has options available. Options follow the hyphen ( - ) sign. Two of the most useful options are a (return all files, even "hidden") and we (give long or full file information). The ls command also accepts strings with the asterisk \* used as a "wildcard" to tell UNIX to search for all files that contain the specified sub-string.

**Syntax**

**ls [-a] [-A] [-b] [-c] [-C] [-d] [-f] [-F] [-g] [-i] [-l] [-L] [-m] [-o] [-p] [-q] [-r] [-R] [-s] [-t] [-u] [-x] [pathnames]**

|  |  |
| --- | --- |
| -a | Shows us all files, even files that are hidden (these files begin with a dot.) |
| -A | List all files including the hidden files. However, does not display the working directory (.) or the parent  directory (..). |
| -b | Force printing of non-printable characters to be in octal \ ddd notation. |
| -c | Use time of last modification of the i-node (file created, mode changed, and so forth) for sorting (-t) or printing (-l or -n). |
| -C | Multi-column output with entries sorted down the columns. Generally this is the default option. |
| -d | If an argument is a directory it only lists its name not its contents. |
| -f | Force each argument to be interpreted as a directory and list the name found in each slot. This option turns off -l, -t, -s, and -r, and turns on -a; the order is the order in which entries appear in the directory. |
| -F | Mark directories with a trailing slash (/), doors with a trailing greater-than sign (>), executable files with a trailing asterisk (\*), FIFOs with a trailing vertical bar (|), symbolic links with a trailing at-sign (@), and AF\_UNIX address family sockets with a trailing equals sign (=). |
| -g | Same as -l except the owner is not printed. |
| -i | For each file, print the i-node number in the first column of the report. |
| -l | Shows us huge amounts of information (permissions, owners, size, and when last modified.) |
| -L | If an argument is a symbolic link, list the file or directory the link references rather than the link itself. |
| -m | Stream output format; files are listed across the page, separated by commas. |
| -n | The same as -l, except that the owner's UID and group's GID numbers are printed, rather than the associated character strings. |
| -o | The same as -l, except that the group is not printed. |
| -p | Displays a slash ( / ) in front of all directories. |
| -q | Force printing of non-printable characters in file names as the character question mark (?). |
| -r | Reverses the order of how the files are displayed. |
| -R | Includes the contents of subdirectories. |
| -s | Give size in blocks, including indirect blocks, for each entry. |
| -t | Shows us the files in modification time. |
| -u | Use time of last access instead of last modification for sorting (with the -t option) or printing (with the –l option). |
| -x | Displays files in columns. |
| -1 | Print one entry per line of output. |
| pathnames | File or directory to list. |

**Examples**

**$ ls -al \*test\***

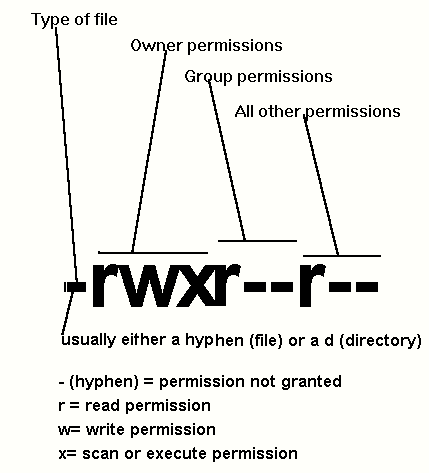
**-rw-r--r-- 1 hcsdar usg 592 Sep 1 1993 .test**

**drwx------ 2 hcsdar usg 512 Nov 11 16:21 dirtest**

**-rw-r--r-- 2 hcsdar usg 1097 Nov 2 1993 test**

**-rw------- 1 hcsdar usg 1097 Oct 19 15:54 test.bin**

**-rw------- 1 hcsdar usg 1216 Jul 15 1993 test.fil**

**What Does Coloumn1 tells us?**

**Column 1-** tells us the type of file, what privileges it has and to whom these privileges are granted. There are three types of privileges. Read and write privileges are easy to understand. The exec privilege is a little more difficult. We can make a file "executable" by giving it exec privileges. This means that commands in the file will be executed when we type the file name in at the UNIX prompt. It also means that when a directory which, to UNIX is a file like any other file, can be "scanned" to see what files and sub-directories are in it. Privileges are granted to three levels of users:

1. The owner of the file. The owner is usually, but not always, the userid that created the file.
2. The group to which the owner belongs. At GSU, the group is usually, but not always designated as the first three letters of the userid of the owner.
3. Everybody else who has an account on the UNIX machine where the file resides.

**Column 2 -**Number of links

**Column 3 -** Owner of the file. Normally the owner of the file is the user account that originally created it.

**Column 4 -** Group under which the file belongs. This is by default the group to which the account belongs or first three letters of the userid. The group can be changed by the chgrp command.

**Column 5 -** Size of file (bytes).

**Column 6 -** Date of last update

**Column 7 -** Name of file

**Examples**

Rather than list the files contained in the /usr directory, this command lists information about the /usr directory itself (without generating a listing of the contents of /usr). This is very useful when we want to check the permissions of the

**$ ls -ld /usr**

List the contents of your home directory by adding a tilde after the ls command.

**$ ls ~**

List the contents of your root directory.

**$ ls /ls ../**

List the contents of the parent directory.

**$ ls \*/**

List the contents of all sub directories.

Only list the directories in the current directory.

**$ ls -d \*/**

**6. finger**

In Unix, finger is a program we can use to find information about computer users. It usually lists the login name, the full name, and possibly other details about the user we are fingering. These details may include the office location and phone number (if known), login time, idle time, time mail was last read, and the user's plan and project files. The information listed varies, and we may not be able to get any information from some sites.

**Syntax**

**finger [-b] [-f] [-h] [-i] [-l] [-m] [-p] [-q] [-s] [-w] [username]**

|  |  |
| --- | --- |
| -b | Suppress printing the user's home directory and shell in a long format printout. |
| -f | Suppress printing the header that is normally printed in a non-long format printout. |
| -h | Suppress printing of the .project file in a long format printout. |
| -i | Force "idle" output format, which is similar to short format except that only the login name, terminal, login time, and idle time are printed. |
| -l | Force long output format. |
| -m | Match arguments only on user name (not first or last name). |
| -p | Suppress printing of the .plan file in a long format printout. |
| -q | Force quick output format, which is similar to short format except that only the login name, terminal, and login time are printed. |
| -s | Force short output format. |
| -w | Suppress printing the full name in a short format printout. |

**Examples**

We can find out someone's username and whether they are logged in with the finger command.

**finger name\_of\_person[@remote\_host]**

"name" can be a first or last name, or a username.

**$ finger wil-1**

**$ Login name: wil-1 In real life: Faculty1**

**how to change Finger**

Most Unix systems have a chfn (change finger) command. It allows us to change the standard information that is displayed when someone fingers your account.

To change your finger information, on most systems, at the Unix shell prompt, enter chfn. We will be prompted to enter values for each of the following fields:

Changing finger information for username

**$ chfin**

**Name [your name]:**

**Location [XY 0436]:**

**Office Phone [555-1212]:**

**Home Phone [555-7352]:**

**7. who**

The who command displays a list of users currently logged in to the local system. It displays each users login name, the login device (TTY port), the login date and time. The command reads the binary file /var/admn/utmpx to obtain this information and information about where the users logged in from

If a user logged in remotely the who command displays the remote host name or internet Protocol (IP) address in the last column of the output.

**Syntax**

$ who [-a] [-b] [-d] [-H] [-l] [-m] [-nx] [-p] [-q] [-r] [-s] [-t] [-T] [-u] [am i] [ file ]

|  |  |
| --- | --- |
| -a | Process /var/adm/utmp or the named file with -b, -d, -l, -p, -r, -t, -T, and -u options turned on. |
| -b | Indicate the time and date of the last reboot. |
| -d | Display all processes that have expired and not been respawned by init . The exit field appears for dead processes and contains the termination and exit values (as returned by wait), of the dead process. This canbe useful in determining why a process terminated. |
| -H | Output column headings above the regular output. |
| -l | List only those lines on which the system is waiting for someone to login. The name field is LOGIN in such cases. Other fields are the same as for user entries except that the state field does not exist. |
| -m | Output only information about the current terminal. |
| -n x | Take a numeric argument, x, which specifies the number of users to display per line. x must be at least 1. The -n option may only be used with -q. |
| -p | List any other process which is currently active and has been previously spawned by init . The name field is the name of the program executed by init as found in /sbin/inittab. The state, line , and idle fields have no meaning. The comment field shows the id field of the line from /sbin/inittab that spawned this process. |
| -q | (quick who ) display only the names and the number of users currently logged on. When this option is used, all other options are ignored. |
| -r | Indicate the current run-level of the init process. |
| -s | (default) List only the name, line, and time fields. |
| -t | Indicate the last change to the system clock (using the date utility) by root. See su and date. |
| -T | Same as the -s option, except that the state field is also written. state is one of the characters listed under the /usr/bin/who version of this option. If the -u option is used with -T, the idle time is added to the end of the previous format. |
| -u | List only those users who are currently logged in. The name is the user's login name. The line is the name of the line as found in the directory /dev. The time is the time that the user logged in. The idle column contains the number of hours and minutes since activity last occurred on that particular line. A dot (.) indicates that the terminal has seen activity in the last minute and is therefore ``current''. If more than twenty-four hours have elapsed or the line has not been used since boot time, the entry is marked old. This field is useful when trying to determine whether a person is working at the terminal or not. The pid is the process-ID of the user's shell. The comment is the comment field associated with this line as found in /sbin/inittab. This can contain information about where the terminal is located, the telephone number of the dataset, type of terminal if hard- wired, and so forth. |
| am i | In the "C" locale, limit the output to describing the invoking user, equivalent to the -m option. The am and i or I must be separate arguments. |
| file | Specify a path name of a file to substitute for the database of logged-on users that who uses by default. |

**Examples**

**$ who**

The general format for output is:

**name [state] line time [idle] [pid] [comment] [exit]**

**where:**

name user's login name.

state capability of writing to the terminal.

line name of the line found in /dev.

time time since user's login.

idle time elapsed since the user's last activity.

pid user's process id.

comment comment line in inittab(4).

**8. cal**

Print a 12-month calendar (beginning with January) for the given year, or a one-month calendar of the given month and year. month ranges from 1 to 12. year ranges from 1 to 9999. With no arguments, print a calendar for the current month.

Before we can do the calendar program we must have a file named calendar at the root of your profile. Within that file we may have something similar to:

**Syntax**

**$ cal [options] [[month] year]**

|  |  |
| --- | --- |
| -j | Display Julian dates (days numbered 1 to 365, starting from January 1). |
| -m | Display Monday as the first day of the week. |
| -y | Display entire year. |
| -V | Display the source of the calendar file. |
| month | Specifies the month for us want the calendar to be displayed. Must be the numeric representation of the month. For example: January is 1 and December is 12. |
| year | Specifies the year that we want to be displayed. |

**EXAMPLES**

$ cal

$ cal -j

$ cal –m

$ cal –y

$ cal –y 1980

$ cal 12 2006

$ cal 2006 > year\_file