PRACTICAL - 1

1. Execute following Linux commands and describe the output

TOUCH	CAT	LS	MKDIR
RMDIR	CD	CLEAR	CP
CAL	HISTORY	CHMOD	UMASK
HEAD	TAIL	DATE	EXPR
WHO	UNAME	FINGER	CMP
COMM	SORT	SPELL	WC
TYPE	TTY	ECHO	MAN
MORE	PASSWD	PWD	GREP
PS	RM	SET	CUT
READ	JOBS	AWK	LN
ENV	KILL	ALIAS	DIFF
LOCATE	FIND	INFO	

TOUCH: -

Syntax

Touch Option files.

DESCRIPTION

Changes the date/time stamp of the file filename to the current time. Creates an empty file if the file does not exist. You can change the stamp to any date using

touch -t 200201311759.30 (year 2002 January day 31 time 17:59:30).

There are three date/time values associated with every file on an ext2 filesystem:
- the time of last access to the file (atime)
- the time of last modification to the file (mtime)
- the time of last change to the file's inode (ctime).

Touch will change the first two to the value specified, and the last one always to the current system time.

OPTIONS

-a, --time=atime, --time=access, --time use Change the access time only.

-c, --no-create Do not create files that do not exist.

-d, --date time Use time (which can be in various common formats) instead of the current time. It can contain month names, timezones, `am' and `pm', etc.

-m, --time=mtime, --time modify Change the modification time only.

-r, --file reference-file Use the times of reference-file instead of the current time.

-t MMDDhhmm[[CC]YY][.ss] Use the argument (months, days, hours, minutes, optional century and years, optional seconds) instead of the current time.

--help Print a usage message on standard output and exit successfully.

--version Print version information on standard output then exit successfully.

EXAMPLE

% touch analysis_data.xls Changes the timestamp of file analysis_data.xls to the current time.

CAT: -

The cat command can be used to join multiple files together and print the result on screen.

SYNOPSIS

cat filename [-n] [-b] [-u] [-s] [-v]

Filename	The name of the file or files that you wish to look at or perform tasks on.
-n	Precede each line output with its line number.
-b	Number the lines, as -n, but omit the line numbers from blank lines.
-u	The output is not buffered. (The default is buffered output.)
-s	cat is silent about non-existent files.
-v	Non-printing characters (with the exception of tabs, new-lines and form-feeds) are printed visibly. ASCII control characters (octal 000 - 037) are printed as ^n, where n is the corresponding ASCII character in the range octal 100 - 137 (@, A, B, C, . . ., X, Y, Z, [, \,], ^, and _); the DEL character (octal 0177) is printed ^?. Other non-printable characters are printed as M-x, where x is the ASCII character specified by the low order seven bits.
-e	A $ character will be printed at the end of each line (prior to the new-line).
-t	Tabs will be printed as ^I's and formfeeds to be printed as ^L's.

*If the -v is used -e and -t will be ignored.

EXAMPLE

Cat file1.txt file2.txt > file3.txt

- Reads file1.txt and file2.txt and combines those files to make file3.txt.

LS: -

Lists the contents of a directory.

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 you 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 you 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 you 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 -l

In the above example this command would list each of the files in the current directory and the files permissions, the size of the file, date of the last modification, and the file name or directory. Below is additional information about each of the fields this command lists.

Permissions	Directories	Group	Size	Date	Directory or file
drwx------	2	users	4096	Nov 2 19:51	mail/
drwxr-s---	35	www	32768	Jan 20 22:39	public_html/
-rw-------	1	users	3	Nov 25 02:58	test.txt

Below is a brief description of each of the above categories shown when using the ls -l command.
Permissions - The permissions of the directory or file.

Directories - The amount of links or directories within the directory. The default amount of directories is going to always be 2 because of the . and .. directories.

Group - The group assigned to the file or directory
Size - Size of the file or directory.
Date - Date of last modification.
Directory of file - The name of the file or file.
ls ~
List the contents of your home directory by adding a tilde after the ls command.
ls /
List the contents of your root directory.
ls ../
List the contents of the parent directory.
ls */
List the contents of all sub directories.
ls -d */
Only list the directories in the current directory.

MKDIR: -

mkdir [options] directories

Create one or more directories. You must have write permission in the parent directory in order to create a directory. See also rmdir. The default mode of the new directory is 0777, modified by the system or user's umask.

Options

-m mode, --mode mode
Set the access mode for new directories. See chmod for an explanation of acceptable formats for mode.
-p, --parents
Create intervening parent directories if they don't exist.
-v, --verbose
Print a message for each directory created.
--help
Print help message and then exit.
--version
Print version number and then exit.
-Z context, --context=context
Set security context in SELinux.

Examples

Create a read-only directory named personal:

```
mkdir -m 444 personal
```

The following sequence:

```
mkdir work; cd work mkdir junk; cd junk mkdir questions; cd ../..
```

can be accomplished by typing this: mkdir -p work/junk/questions

RMDIR: -

Deletes a directory.

Syntax

```
rmdir [OPTION]... DIRECTORY...
```

--ignore-fail-on-non-empty	ignore each failure that is solely because a directory is non-empty.
-p, --parents	Remove DIRECTORY and its ancestors. E.g., `rmdir -p a/b/c' is similar to `rmdir a/b/c a/b a'.
-v, --verbose	output a diagnostic for every directory processed.
--version	output version information and exit.

Examples

rmdir mydir - removes the directory mydir
rm -r directory - would remove a directory, even if files existed in that directory.

CD: -

Changes the directory.

Syntax

cd [directory]

directory	Name of the directory user wishes to enter.
cd ..	Used to go back one directory on the majority of all Unix shells. It is important that the space be between the cd and the ..
cd -	When in a Korn shell to get back one directory used to go back one directory.

Examples

cd hope
The above example would go into the hope directory if it exists.
cd ../home/users/computerhope
The above example would go back one directory and then go into the home/users/computerhope directory.
cd ../../
Next, the above example would go back two directories.

CLEAR: -

Clears the screen

Syntax

clear

Examples

clear - would clear the screen.

CP: -

Copies files from one location to another.

Syntex

cp [OPTION]... SOURCE DEST
cp [OPTION]... SOURCE... DIRECTORY
cp [OPTION]... --target-directory=DIRECTORY SOURCE...

-a, --archive	same as -dpR
--backup[=CONTROL]	make a backup of each existing destination file
-b	like --backup but does not accept an argument
--copy-contents	copy contents of special files when recursive
-d	same as --no-dereference --preserve=link
--no-dereference	never follow symbolic links
-f, --force	if an existing destination file cannot be opened, remove it and try again
-i, --interactive	prompt before overwrite
-H	follow command-line symbolic links
-l, --link	link files instead of copying
-L, --dereference	always follow symbolic links
-p	same as --preserve=mode,ownership,timestamps
--preserve[=ATTR_LIST]	preserve the specified attributes (default: mode,ownership,timestamps), if possible additional attributes: links, all
--no-preserve=ATTR_LIST	don't preserve the specified attributes
--parents	append source path to DIRECTORY
-P	same as '--no-dereference'
-R, -r, --recursive	copy directories recursively
--remove-destination	remove each existing destination file before attempting to open it (contrast with --force)
--reply={yes,no,query}	specify how to handle the prompt about an existing destination file
--sparse=WHEN	control creation of sparse files
--strip-trailing-slashes	remove any trailing slashes from each SOURCE argument
-s, --symbolic-link	make symbolic links instead of copying
-S, --suffix=SUFFIX	override the usual backup suffix
--target-directory=DIRECTORY	move all SOURCE arguments into DIRECTORY
-u, --update	copy only when the SOURCE file is newer than the destination file or when the destination file is missing
-v, --verbose	explain what is being done
-x, --one-file-system	stay on this file system

Examples

cp file1.txt newdir

Copies the file1.txt in the current directory to the newdir directory.

cp /home/public_html/mylog.txt /home/public_html/backup/mylog.bak

Copies the mylog.txt file in the public_html directory into the public_html/backup directory as mylog.bak. The files are identical however have different names.

cp *.txt newdir

Copy all files ending in .txt into the newdir directory.

cp -r /home/hope/files/* /home/hope/backup

Copies all the files, directories, and subdirectories in the files directory into the backup directory.

<u>CAL: -</u>

Calendar for the month and the year.

<u>Syntax</u>

cal [month] [year]

month	Specifies the month for you 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 you want to be displayed.

<u>Examples</u>

cal -	Would give you the calendar for this month.
cal 12 2000 -	Would give you the calendar for December of 2000.

<u>HISTORY: -</u>

The 'history' utility allows you to use words from previous command lines in the command line you are typing. This simplifies spelling corrections and the repetition of complicated commands or arguments.

<u>Syntax</u>

history [-h] [-r] [n]

!	Start a history substitution, except when followed by a space character, tab, newline, = or (.
!!	Runs the last command that you ran.
!10	Re-run line number 10 in the history.
!-n	Refer to the current command line minus n.

!?str?	Refer to the most recent command containing str (string).
!str	Re runs the last command that you ran that starts with str (string).

Examples

history - Typing history alone would give results similar to the following:

```
2 grep --help
3 bg
4 fg
5 pine
6 cd public_html
7 rm index.html
8 sz index.html
9 ls -laxo
10 chmod 755 index.htm
```

!ls - Would execute the last ls command.

!! - Would execute the last command executed.

CHMOD: -

Changes the permission of a file.

Syntax

```
chmod [OPTION]... MODE[,MODE]... FILE...
chmod [OPTION]... OCTAL-MODE FILE...
chmod [OPTION]... --reference=RFILE FILE...
```

-c, --changes	like verbose but report only when a change is made
--no-preserve-root	do not treat `/' specially (the default)
--preserve-root	fail to operate recursively on `/'
-f, --silent, --quiet	suppress most error messages
-v, verbose	output a diagnostic for every file processed
--reference=RFILE	use RFILE's mode instead of MODE values
-R, --recursive	change files and directories recursively
--help	display this help and exit
--version	output version information and exit

Permissions

u - User who owns the file.
g - Group that owns the file.
o - Other.
a - All.
r - Read the file.
w - Write or edit the file.
x - Execute or run the file as a program.

Numeric Permissions:

CHMOD can also to attributed by using Numeric Permissions:

400 read by owner
040 read by group
004 read by anybody (other)
200 write by owner
020 write by group
002 write by anybody
100 execute by owner
010 execute by group
001 execute by anybody

Examples

The above numeric permissions can be added to set a certain permission, for example, a common HTML file on a Unix server to be only viewed over the Internet would be:

chmod 644 file.htm

This gives the file read/write by the owner and only read by everyone else (-rw-r--r--).

Files such as scripts that need to be executed need more permissions. Below is another example of a common permission given to scripts.

chmod 755 file.cgi

This would be the following 400+040+004+200+020+100+010+001 = 775 where you are giving all the rights but the capability for anyone to edit your file.cgi (-rwxr-xr-x).

Finally, another common CHMOD permission is 666, as shown below, which is read and write by everyone.

chmod 666 file.txt

Additional information

Below is an example of how a file may be listed when typing (ls -l) at the prompt as well as information on how to interpret it.

```
-rw-rw-r-- 1   hope  123   Feb 03 15:36   file.txt
```

-	rw	rw-	r--	1	hope	123	Feb 03 15:36	file.txt
File	owner	group	everyone else	links	owner	size	mod date	file name

UMASK: -

Get or set the file mode creation mask.

Syntax

```
umask [-S] [000] [mask]
```

-S	Produce symbolic output. The default output style is unspecified, but will be recognized on a subsequent invocation of umask on the same system as a mask operand to restore the previous file mode creation mask.
000	Any three octal digits used for permissions.

Examples

```
umask
```

Running umask alone will display the current umask environment settings.

```
umask 000
```

The above example allows any permissions. This does not actually set permissions just allows permissions to be set.

HEAD: -

Displays the first ten lines of a file, unless otherwise stated.

Syntax

```
head [-number | -n number] filename
```

-number	The number of the you want to display.
-n number	The number of the you want to display.
filename	The file that you want to display the x amount of lines of.

Example

head -15 myfile.txt - Would display the first fifteen lines of myfile.txt.

TAIL: -

Delivers the last part of the file.

Syntax

tail [+ number] [-l] [-b] [-c] [-r] [-f] [-c number | -n number] [file]

+number	
-l	Units of lines.
-b	Units of blocks.
-c	Units of bytes.
-r	Reverse. Copies lines from the specified starting point in the file in reverse order. The default for r is to print the entire file in reverse order.
-f	Follow. If the input-file is not a pipe, the program will not terminate after the line of the input-file has been copied, but will enter an endless loop, wherein it sleeps for a second and then attempts to read and copy further records from the input-file. Thus it may be used to monitor the growth of a file that is being written by some other process.
-c number	The number option-argument must be a decimal integer whose sign affects the location in the file, measured in bytes, to begin the copying:

+	Copying starts relative to the beginning of the file.
-	Copying starts relative to the end of the file.
none	Copying starts relative to the end of the file.

The origin for counting is 1; that is, -c+1 represents the first byte of the file, -c-1 the last.

-n number	Equivalent to -c number, except the starting location in the file is measured in lines instead of bytes. The origin for counting is 1; that is, -n+1 represents the first line of the file, -n-1 the last.
file	Name of the file you wish to display

Examples

tail myfile.txt

The above example would list the last 10 (default) lines of the file myfile.txt.

tail myfile.txt -n 100

The above example would list the last 100 lines in the file myfile.txt.

DATE: -

Tells you the date and time in Unix.

Syntax

date [-a] [-u] [-s datestr]

-a	Slowly adjust the time by sss.fff seconds (fff represents fractions of a second). This adjustment can be positive or negative. The system's clock will be sped up or slowed down until it has drifted by the number of seconds specified. Only the super-user may adjust the time.
-u	Display (or set) the date in Greenwich Mean Time (GMT-universal time), bypassing the normal conversion to (or from) local time.
-s datestr	Sets the time and date to the value specfied 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 - Would list the date and time of the server. Below is an example of the output.

Thu Feb 8 16:47:32 MST 2001

date -s "11/20/2003 12:48:00" - Set the date to the date and time shown.

date '+DATE: %m/%d/%y%nTIME:%H:%M:%S' - Would list the time and date in the below format:

```
DATE: 02/08/01
TIME:16:44:55
```

EXPR: -

Evaluate arguments as an expression.

Syntax

expr expression

expr1 | expr2	results in the value expr1 if expr1 is true; otherwise it results in the value of expr2.
expr1 &	results in the value of expr1 if both expressions are true; otherwise it results in 0
expr1 <= expr1 < expr1 = expr1 != expr1 >= expr1 >	If both expr1 and expr2 are numeric, `expr` compares them as numbers; otherwise it compares them as strings. If the comparison is true, the expression results in 1; otherwise it results in 0.
expr1 + expr1 -	performs addition or subtraction on the two expressions. If either expression is not a number, `expr` exits with an error.
expr1 * expr1 / expr1 %	performs multiplication, division, or modulus on the two expressions. If either expression is not a number, `expr` exits with an error. Note that the multiplication symbol (*) is expanded under the KornShell unless you specify it with a leading backslash (`\\*`), or enclosed in single quotes (`'*'`) or double quotes (`"*"`). Under `command.com` you cannot use the backslash to prevent expansion
expr1 : re match expr1 re	matches the regular expression re against expr1 treated as a string. The regular expression is the same as that accepted by **ed**, except that the match is always anchored, that is, there is an implied leading `^`; therefore `expr` does not consider `^` to be a metacharacter. If the regular expression contains `\(...\)` and it matches at least part of expr1, then `expr` results in only that part; if there is no match, `expr` results in 0. If the regular expression doesn't contain this construct, then the result is the number of characters matched. The function `match` performs the same operation as the colon operator.
substr expr1 expr2 expr3	results in the substring of expr1 starting at character position expr2 (origin 1) for the length of expr3 characters.
index expr1 expr2	searches for any of the characters in expr2 in expr1 and returns the first character position (origin 1) at which it finds such a character, or 0 if no such characters are found.
length expr1	returns the length of expr1 in characters.

(expr)	groups expressions.

Examples

expr "$VAR" : '.*' - return the number of characters in $VAR.

WHO: -

Displays who is on the system.

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 can be 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).

Below is an example of what this may look like

```
chope  pts/0  Apr 23 10:43  (shell.computerhope.com)
```

```
hope   pts/1  May 6 18:19   (shell.computerhope.com)
```

UNAME: -

Print name of current system.

Syntax

```
uname [-a] [-i] [-m] [-n] [-p] [-r] [-s] [-v] [-X] [-S systemname]
```

-a	Print basic information currently available from the system.
-i	Print the name of the hardware implementation (platform).
-m	Print the machine hardware name (class). Use of this option is discouraged; use uname -p instead.
-n	Print the nodename (the nodename is the name by which the system is known to a communications network).
-p	Print the current host's ISA or processor type.
-r	Print the operating system release level.
-s	Print the name of the operating system. This is the default.
-v	Print the operating system version.
-X	Print expanded system information, one information element per line, as expected by SCO Unix. The displayed information includes: • system name, node, release, version, machine, and number of CPUs. • BusType, Serial, and Users (set to "unknown" in Solaris) • OEM# and Origin# (set to 0 and 1, respectively)
-S systemname	The nodename may be changed by specifying a system name argument. The system name argument is restricted to SYS_NMLN characters. SYS_NMLN is an implementation specific value defined in <sys/utsname.h>. Only the super-user is allowed this capability.

Examples

uname -arv - list results similar to the below:

SunOS hope 5.7 Generic_106541-08 sun4m sparc SUNW,SPARCstation-10

FINGER: -

Lists information about the user.

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

finger -b -p ch - Would display the following information about the user ch.

```
Login name: admin In real life: Computer Hope
On since Feb 11 23:37:16 on pts/7 from domain.computerhope.com
28 seconds Idle Time
Unread mail since Mon Feb 12 00:22:52 2001
```

CMP: -

Compares two files and tells you which line numbers are different.

Syntax

cmp [-c] [-i N] [-l] [-s] [-v] firstfile secondfile

-c	Output differing bytes as characters.
-i N	Ignore differences in the first N bytes of input.

-l	Write the byte number (decimal) and the differing bytes (octal) for each difference.
-s	Write nothing for differing files; return exit statuses only.
-v	Output version info.
firstfile	First file that you wish to compare.
secondfile	Second file that you wish to compare to.

Examples

cmp file1.txt file2.txt - Compares file1 to file2 and outputs results. Below is example of how these results may look.

```
file.txt file2.txt differ: char 1011, line 112
```

COMM: -

Select or reject lines common to two files.

Syntax

comm [-1] [-2] [-3] file1 file2

-1	Suppress the output column of lines unique to file1.
-2	Suppress the output column of lines unique to file2.
-3	Suppress the output column of lines duplicated in file1 and file2.
file1	Name of the first file to compare.
file2	Name of the second file to compare.

Examples

comm myfile1.txt myfile2.txt

The above example would compare the two files myfile1.txt and myfile2.txt.

SORT: -

Sorts the lines in a text file.

Syntax

sort [-b] [-d] [-f] [-i] [-m] [-M] [-n] [-r] [-u] [+fields] filename [-o outputfile]

-b	Ignores spaces at beginning of the line.
-d	Uses dictionary sort order and ignores the punctuation.
-f	Ignores caps
-i	Ignores nonprinting control characters.
-m	Merges two or more input files into one sorted output.
-M	Treats the first three letters in the line as a month (such as may.)
-n	Sorts by the beginning of the number at the beginning of the line.
-r	Sorts in reverse order
-u	If line is duplicated only display once
+fields	Sorts by fields , usually by tabs
filename	The name of the file that needs to be sorted.
-o outputfile	Sends the sorted output to a file.

Examples

sort -r file.txt - Would sort the file, file.txt in reverse order.

SPELL: -

Looks through a text file and reports any words that it finds in the text file that are not in the dictionary.

Syntax

spell [-b] [-i] [-l] [-v] [-x] [+wordlist] filenames

-b	Uses British spellings.
-i	Cause deroff to ignore .so and .nx commands. If deroff is not present on the system, then this option is ignored.
-l	Follow the chains of all included files.
-v	Print all words not literally in the spelling list, as well as plausible derivations from the words in the spelling list.
-x	Print every plausible stem, one per line, with = preceding each word.
+wordlist	Adds words to the dictionary so next time they are found it does not think that they are incorrect.
filenames	The name of the file(s) to be spell checked.

Examples

spell myfile.txt - spell checks the file myfile.txt.

WC: -

Short for word count, wc displays a count of lines, words, and characters in a file.

Syntax

wc [-c | -m | -C] [-l] [-w] [file ...]

-c	Count bytes.
-m	Count characters.
-C	Same as -m.
-l	Count lines.
-w	Count words delimited by white space characters or new line characters. Delimiting characters are Extended Unix Code (EUC) characters from any code set defined by iswspace()
file	Name of file to word count.

Examples

wc myfile.txt - Displays information about the file myfile.txt. Below is an example of the output.

```
5    13    57   myfile.txt
```

5 = Lines
13 = Words
57 = Characters

TYPE: -

Allows the user to see the contents of a file.

To edit the files, the user would need to use either edit or copy con.

Syntax

Displays the contents of text files.

TYPE [drive:][path]filename

Examples

type c:\autoexec.bat

This would allow you to look at the autoexec.bat

TTY: -

print the file name of the terminal connected to standard input

SYNOPSIS

tty [OPTION]...

DESCRIPTION

Print the file name of the terminal connected to standard input.

-s, --silent, --quiet	print nothing, only return an exit status
--help	display this help and exit
--version	output version information and exit

ECHO: -

Echo's to the screen what you type after echo. Echo is useful for producing diagnostics in command files, for sending known data into a pipe, and for displaying the contents of environment variables.

Syntax

echo [-n] text

-n	On BSD and some variants derived from BSD does not begin a new line after the echoed text.
text	The text that you want to echo to the screen.

Examples

echo Hello world

The above example would return "Hello world" to the console

echo * | wc

The above example would list a count of all the files and directories in the current directory. Additional examples relating to this example can also be found on document CH000756.

MAN: -

The man command which is short for manual provides in depth information about the requested command or allows users to search for commands related to a particular keyword.

Syntax

Shows you online manuals on Unix commands.

man [-] [-k keywords] topic

-	Displays the manual without stopping.
-k keywords	Searches for keywords in all of the manuals available.
topic	Displays the manual for the topic or command typed in.

Examples

man mkdir - Lists help information on the mkdir command.

man -k irc - Quickly search for manuals containing irc within them. Below is an example of what the results may look like:

MORE: -

Displays text one screen at a time.

Syntax

more [-c] [-d] [-e] [-f] [-i] [-l] [-n number] [-p command] [-r] [-s] [-t tagstring] [-u] [-w] [-lines] [+ linenumber] [+/ pattern] [file ...]

-c	Clear before displaying. Redraws the screen instead of scrolling for faster displays. This option is ignored if the terminal does not have the ability to clear to the end of a line.
-d	Display error messages rather than ringing the terminal bell if an unrecognized command is used. This is helpful for inexperienced users.
-e	Exit immediately after writing the last line of the last file in the argument list.
-f	Do not fold long lines. This is useful when lines contain nonprinting characters or escape sequences, such as those generated when nroff output is piped through ul.
-i	Perform pattern matching in searches without regard to case.
-l	Ignores form-feed characters (Ctrl + L starts the new page.)
-n number	Specify the number of lines per screenful. The number argument is a positive decimal integer. The -n option overrides any values obtained from the environment.
-p command	For each file examined, initially execute the more command in the command argument. If the command is a positioning command, such as a line number or a regular expression search, set the current position to represent the final results of the command, without writing any intermediate lines of the file.
-r	Displays control keys.
-s	Doesn't display extra blank lines.
-t tagstring	Write the screenful of the file containing the tag named by the tagstring

	argument.
-u	Ignores backspace and underscores.
-w	Normally, more exits when it comes to the end of its input. With -w, however, more prompts and waits for any key to be struck before exiting.
-lines	Display the indicated number of lines in each screenful, rather than the default (the number of lines in the terminal screen less two).
+linenumber	Start up at linenumber
+/pattern	Displays text two lines before the first time text appears.
filename	The name of the file.

Examples

```
more +3 myfile.txt
```

In the above example the command would begin displaying the file myfile.txt at line three.

PASSWD: -

passwd [*options*] [*user*]

Create or change a password associated with a *user* name. Only the owner or a privileged user may change a password. Owners need not specify their *user* name. Users can change their own passwords. For any other operation, you must be root.

Options

-d, --delete Delete the password for the user's account.

-f, --force   Force the operation. Overrides -u.

-?, --help   Display a help message describing the options. See also --usage.

-i *days*, --inactive   *days* Set the number of days after a password has expired before the account

is disabled.

-k, --keep-tokens Keep passwords (authentication tokens) that have not expired.

-l, --lock Lock the user's account.

-n *days*, --minimum =*days* Set the minimum number of days that the password is valid.

-S, --status Print the status of the user's password.

--stdin Read new passwords from standard input.

-u, --unlock Unlock the user's account

--usage Display a brief usage message. See also --help.

-w *days*, --warning=*days* Set the number of days of warning users will get before their password expires.

-x *days*, --maximum=*days* Set the maximum number of days that the password is valid.

<u>PWD: -</u>

Short for print working directory the pwd command displays the name of the current working directory.

<u>Syntax</u>

```
pwd
```

<u>Examples</u>

pwd - Typing pwd at the prompt would give you something similar to:

```
/home/computerhope/public_html
```

Users who are familiar with MS-DOS or the Windows command prompt may type cd alone to print the working directory. However, typing cd alone in Linux / Unix will return you to the home directory.

<u>GREP: -</u>

Finds text within a file.

<u>Syntax</u>

```
grep [options] PATTERN [FILE...]
grep [options] [-e PATTERN | -f FILE] [FILE...]
```

-A NUM, --after-context=NUM	Print NUM lines of trailing context after matching lines. Places a line containing -- between contiguous groups of matches.
-a, --text	Process a binary file as if it were text; this is equivalent to the --binary-files=text option.
-B NUM, --before-context=NUM	Print NUM lines of leading context before matching lines. Places a line containing -- between contiguous groups of matches.
-b, --byte-offset	Print the byte offset within the input file before each line of output.
--binary-	If the first few bytes of a file indicate that the file contains binary data,

files=TYPE	assume that the file is of type TYPE. By default, TYPE is binary, and grep normally outputs either a one-line message saying that a binary file matches, or no message if there is no match. If TYPE is without-match, grep assumes that a binary file does not match; this is equivalent to the -I option. If TYPE is text, grep processes a binary file as if it were text; this is equivalent to the -a option. Warning: grep --binary-files=text might output binary garbage, which can have nasty side effects if the output is a terminal and if the terminal driver interprets some of it as commands.
-C NUM, --context=NUM	Print NUM lines of output context. Places a line containing -- between contiguous groups of matches.
-c, --count	Suppress normal output; instead print a count of matching lines for each input file. With the -v, --invert-match option (see below), count non-matching lines.
--colour[=WHEN], --color[=WHEN]	Surround the matching string with the marker find in GREP_COLOR environment variable. WHEN may be `never', `always', or `auto'
-D ACTION, --devices=ACTION	If an input file is a device, FIFO or socket, use ACTION to process it. By default, ACTION is read, which means that devices are read just as if they were ordinary files. If ACTION is skip, devices are silently skipped.
-d ACTION, --directories=ACTION	If an input file is a directory, use ACTION to process it. By default, ACTION is read, which means that directories are read just as if they were ordinary files. If ACTION is skip, directories are silently skipped. If ACTION is recurse, grep reads all files under each directory, recursively; this is equivalent to the -r option.
-E, --extended-regexp	Interpret PATTERN as an extended regular expression (see below).
-e PATTERN, --regexp=PATTERN	Use PATTERN as the pattern; useful to protect patterns beginning with -.
-F, --fixed-strings	Interpret PATTERN as a list of fixed strings, separated by newlines, any of which is to be matched.
-f FILE, --file=FILE	Obtain patterns from FILE, one per line. The empty file contains zero patterns, and therefore matches nothing.
-G, --basic-regexp	Interpret PATTERN as a basic regular expression (see below). This is the default.
-H, --with-filename	Print the filename for each match.
-h, --no-filename	Suppress the prefixing of filenames on output when multiple files are searched.
--help	Output a brief help message.
-I	Process a binary file as if it did not contain matching data; this is equivalent to the --binary-files=without match option.
-i, --ignore-case	Ignore case distinctions in both the PATTERN and the input files.
-L, --files-without-match	Suppress normal output; instead print the name of each input file from which no output would normally have been printed. The scanning will stop on the first match.
-l, --files-with-matches	Suppress normal output; instead print the name of each input file from which output would normally have been printed. The scanning will stop

	on the first match.
--label=LABEL	Displays input actually coming from standard input as input coming from file LABEL. This is especially useful for tools like zgrep, e.g. gzip -cd foo.gz |grep --label=foo something
--line-buffered	Use line buffering, it can be a performance penalty.
-m NUM, --max-count=NUM	Stop reading a file after NUM matching lines. If the input is standard input from a regular file, and NUM matching lines are output, grep ensures that the standard input is positioned to just after the last matching line before exiting, regardless of the presence of trailing context lines. This enables a calling process to resume a search. When grep stops after NUM matching lines, it outputs any trailing context lines. When the -c or --count option is also used, grep does not output a count greater than NUM. When the -v or --invert-match option is also used, grep stops after outputting NUM non-matching lines.
--mmap	ations, --mmap yields better performance. However, --mmap can cause undefined behavior (including core dumps) if an input file shrinks while grep is operating, or if an I/O error occurs.
-n, --line-number	Prefix each line of output with the line number within its input file.
-o, --only-matching	Show only the part of a matching line that matches PATTERN.
-P, --perl-regexp	Interpret PATTERN as a Perl regular expression.
-q, --quiet, --silent	Quiet; do not write anything to standard output. Exit immediately with zero status if any match is found, even if an error was detected. Also see the -s or --no-messages option.
-R, -r, --recursive	Read all files under each directory, recursively; this is equivalent to the -d recurse option.
--include=PATTERN	Recurse in directories only searching file matching PATTERN.
--exclude=PATTERN	Recurse in directories skip file matching PATTERN.
-s, --no-messages	Suppress error messages about nonexistent or unreadable files. Portability note: unlike GNU grep, traditional grep did not conform to POSIX.2, because traditional grep lacked a -q option and its -s option behaved like GNU grep's -q option. Shell scripts intended to be portable to traditional grep should avoid both -q and -s and should redirect output to /dev/null instead.
-U, --binary	Treat the file(s) as binary. By default, under MS-DOS and MS-Windows, grep guesses the file type by looking at the contents of the first 32KB read from the file. If grep decides the file is a text file, it strips the CR characters from the original file contents (to make regular expressions with ^ and $ work correctly). Specifying -U overrules this guesswork, causing all files to be read and passed to the matching mechanism verbatim; if the file is a text file with CR/LF pairs at the end of each line, this will cause some regular expressions to fail. This option has no effect on platforms other than MS-DOS and MS-Windows.
-u, --unix-	Report Unix-style byte offsets. This switch causes grep to report byte

byte-offsets	offsets as if the file were Unix-style text file, i.e. with CR characters stripped off. This will produce results identical to running grep on a Unix machine. This option has no effect unless -b option is also used; it has no effect on platforms other than MS-DOS and MS-Windows.
-V, --version	Print the version number of grep to standard error. This version number should be included in all bug reports (see below).
-v, --invert-match	Invert the sense of matching, to select non-matching lines.
-w, --word-regexp	Select only those lines containing matches that form whole words. The test is that the matching substring must either be at the beginning of the line, or preceded by a non-word constituent character. Similarly, it must be either at the end of the line or followed by a non-word constituent character. Word-constituent characters are letters, digits, and the underscore.
-x, --line-regexp	Select only those matches that exactly match the whole line.
-y	Obsolete synonym for -i.
-Z, --null	Output a zero byte (the ASCII NUL character) instead of the character that normally follows a file name. For example, grep -lZ outputs a zero byte after each file name instead of the usual newline. This option makes the output unambiguous, even in the presence of file names containing unusual characters like newlines. This option can be used with commands like find -print0, perl -0, sort -z, and xargs -0 to process arbitrary file names, even those that contain newline characters.
-z, --null-data	Treat the input as a set of lines, each terminated by a zero byte (the ASCII NUL character) instead of a newline. Like the -Z or --null option, this option can be used with commands like sort -z to process arbitrary file names.

Patterns for searching.

.	Matches single character.
*	Wild character Example C* if found would pull up CC or CAT...
{}	Matches any character contained within the bracket.
^	Represents the beginning of the line, so if you did ^T it would search for any sentence starting with a T.
$	Represents the end of the line, so if you did $. then it would pull up any lines that ended with .
\	Means to take the next character serious so you could search for C\ C.

Note: Be careful using the characters $, *, [, ^, \|, (,), and \ in the pattern list because they are also meaningful to the shell. It is safest to enclose the entire pattern list in single quotes '... '.

<u>Examples</u>

```
grep "unix" *.htm
```

search all .htm files in the current directory for any reference of unix and give results similar to the below example text.

```
asoftwar.htm: href="win95.htm">Windows 95</a>, <a href="unix.htm">Unix</a>, <a
href="msdos.htm">MS-DOS</a>,
asoftwar.htm: <td><font face="Times New Roman"><a name="U"></a><a
href="unix.htm"><strong>Unix</strong></a></font></td>
learnhtm.htm: <a href="unix.htm">Unix help</a><br>
os.htm: <a href="unix.htm">Unix</a><br>
```

As seen above the grep command has found references of unix in some of the HTML files in our home directory. The file name that contains unix is listed at the beginning of the line followed by a colon and the text continuing unix.

Argument list too long

When using grep or other commands that requires a listing or search through several thousand files you may get the "Argument list too long" or "/bin/grep: Argument list too long." error. When this occurs you may want to use a command similar to the below, using the find command and xargs command in conjunction with the grep.

```
find Members/ -type f -print0 | xargs -0 grep "examplestring"
```

In the above example the find command finds all files in the Members directory each file that is found is then searched using grep for the text "examplestring". This above example had no problems searching over 100 thousand files.

PS: -

Reports the process status.

Syntax

```
ps [-a] [-A] [-c] [-d] [-e] [-f] [-j] [-l] [-L] [-P] [-y] [ -g grplist ] [ -n namelist ] [-o
format ] [ -p proclist ] [ -s sidlist ] [ -t term] [ -u uidlist ] [ -U uidlist ] [ -G gidlist ]
```

-a	List information about all processes most frequently requested: all those except process group leaders and processes not associated with a terminal.
-A	List information for all processes. Identical to -e, below.
-c	Print information in a format that reflects scheduler properties as described in priocntl. The -c option affects the output of the -f and -l options, as described below.
-d	List information about all processes except session leaders.

-e	List information about every process now running.
-f	Generate a full listing.
-j	Print session ID and process group ID.
-l	Generate a long listing.
-L	Print information about each light weight process (lwp) in each selected process.
-P	Print the number of the processor to which the process or lwp is bound, if any, under an additional column header, PSR.
-y	Under a long listing (-l), omit the obsolete F and ADDR columns and include an RSS column to report the resident set size of the process. Under the -y option, both RSS and SZ will be reported in units of kilobytes instead of pages.
-g grplist	List only process data whose group leader's ID number(s) appears in grplist. (A group leader is a process whose process ID number is identical to its process group ID number.)
-n namelist	Specify the name of an alternative system namelist file in place of the default. This option is accepted for compatibility, but is ignored.
-o format	Print information according to the format specification given in format. This is fully described in DISPLAY FORMATS. Multiple -o options can be specified; the format specification will be interpreted as the space-character-separated concatenation of all the format option-arguments.
-p proclist	List only process data whose process ID numbers are given in proclist.
-s sidlist	List information on all session leaders whose IDs appear in sidlist.
-t term	List only process data associated with term. Terminal identifiers are specified as a device file name, and an identifier. For example, term/a, or pts/0.
-u uidlist	List only process data whose effective user ID number or login name is given in uidlist. In the listing, the numerical user ID will be printed unless you give the -f option, which prints the login name.
-U uidlist	List information for processes whose real user ID numbers or login names are given in uidlist. The uidlist must be a single argument in the form of a blank- or comma-separated list.
-G gidlist	List information for processes whose real group ID numbers are given in gidlist. The gidlist must be a single argument in the form of a blank- or

comma-separated list.

Examples

ps

Typing ps alone would list the current running processes. Below is an example of the output that would be generated by the ps command.

```
PID   TTY    TIME   CMD
6874  pts/9  0:00   ksh
6877  pts/9  0:01   csh
418   pts/9  0:00   csh
```

ps -ef

Display full information about each of the processes currently running.

```
UID PID PPID C STIME TTY TIME CMD
hope 29197 18961 0 Sep27 ? 00:00:06 sshd: hope@pts/87
hope 32097 29197 0 Sep27 pts/87 00:00:00 -csh
hope 7209 32097 0 12:17 pts/87 00:00:00 ps -ef
```

RM:-

Deletes a file without confirmation (by default).

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 not suppress 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.
-R	Same as -r option.
-r	Recursively remove directories and subdirectories in the argument list. The directory will be emptied of files and removed. The user is normally

prompted for removal of any write-protected files which the directory contains. The write-protected files are removed without prompting, however, if the -f option is used, or if the standard input is not a terminal and the -i option is not used. Symbolic links that are encountered with this option will not be traversed. If the removal of a non-empty, write-protected directory is attempted, the utility will always fail (even if the -f option is used), resulting in an error message.

filenames	A path of a filename to be removed.

Examples

rm myfile.txt

Remove the file myfile.txt without prompting the user.

rm -r directory

Remove a directory, even if files existed in that directory.

Note that if you use rm to remove a file, it is usually possible to recover the contents of that file. If you want more assurance that the contents are truly unrecoverable, consider using shred.

SET:-

In C shell sets the value of an environment variable.

Syntax

set [--] [-a] [-e] [-f] [-h] [-k] [-m] [-n] [-p] [-s] [-t] [-u] [-v] [-x] [-o option] [-A name] [arg]

-	- Do not change any of the flags; useful in setting $1 to -.
a	-Mark variables which are modified or created for export.
e	-Exit immediately if a command exits with a nonzero exit status.
f	-Disable file name generation.
h	-Locate and remember function commands as functions are defined (function commands are normally located when the function is executed).
k	-All keyword arguments are placed in the environment for a command, not just those that precede the command name.
m	-Background jobs will run in a separate process group and a line will print upon completion. The exit status of background jobs is reported in a completion message. On systems with job control, this flag is turned on automatically for interactive shells.
	-Read commands but do not execute them.

n	-Disables processing of the $HOME/.profile file and uses the file /etc/suid_profile instead of the ENV file. This mode is on whenever the effective uid is not equal to the real uid, or when the effective gid is not equal to the real gid. Turning this off causes the effective uid and gid to be set to the real uid and gid.
p	-Sort the positional parameters lexicographically.
s	-Exit after reading and executing one command.
t	-Treat unset variables as an error when substituting.
u	-Print shell input lines as they are read.
v	-Print commands and their arguments as they are executed.
x	-The following argument can be one of the following option names:
o option	
- A name	Array assignment. Unset the variable name and assign values sequentially from the list arg. If +A is used, the variable name is not unset first.

* Using + rather than - causes these flags to be turned off.

Examples **setenv PATH "/bin:/usr/bin:/usr/sbin:ucb/bin" - Sets the environment path to search for files in the /bin, /usr/bin, /usr/sbin and usb/bin directory.**

CUT:-

Cut out selected fields of each line of a file.

Syntax

cut [-b] [-c] [-f] list [-n] [-d delim] [-s] [file]

-b list	The list following -b specifies byte positions (for instance, -b1-72 would pass the first 72 bytes of each line). When -b and -n are used together, list is adjusted so that no multi-byte character is split. If -b is used, the input line should contain 1023 bytes or less.
-c list	The list following -c specifies character positions (for instance, -c1-72 would pass the first 72 characters of each line).
-f list	The list following -f is a list of fields assumed to be separated in the file by a delimiter character (see -d); for instance, -f1,7 copies the first and seventh field only. Lines with no field delimiters will be passed through intact (useful for table subheadings), unless -s is specified. If -f is used, the input line should contain 1023 characters or less.
list	A comma-separated or blank-character-separated list of integer field numbers (in increasing order), with optional - to indicate ranges (for

	instance, 1,4,7; 1-3,8; -5,10 (short for 1-5,10); or 3- (short for third through last field)).
-n	Do not split characters. When -b list and -n are used together, list is adjusted so that no multi-byte character is split.
-d delim	The character following -d is the field delimiter (-f option only). Default is tab. Space or other characters with special meaning to the shell must be quoted. delim can be a multi-byte character.
-s	Suppresses lines with no delimiter characters in case of -f option. Unless specified, lines with no delimiters will be passed through untouched.
file	A path name of an input file. If no file operands are specified, or if a file operand is -, the standard input will be used.

Examples

name=`who am i | cut -f1 -d' '` - set name to current login name.

READ:-

read - Read from a channel

SYNOPSIS

read ?-nonewline? channelId

read channelId numChars

DESCRIPTION

In the first form, the read command reads all of the data from channelId up to the end of the file. If the -nonewline switch is specified then the last character of the file is discarded if it is a newline. In the second form, the extra argument specifies how many characters to read. Exactly that many characters will be read and returned, unless there are fewer than numChars left in the file; in this case all the remaining characters are returned. If the channel is configured to use a multi-byte encoding, then the number of characters read may not be the same as the number of bytes read.
If channelId is in nonblocking mode, the command may not read as many characters as requested: once all available input has been read, the command will return the data that is available rather than blocking for more input. If the channel is configured to use a multi-byte encoding, then there may actually be some bytes remaining in the internal buffers that do not form a complete character. These bytes will not be returned until a complete character is available or end-of-file is reached. The -nonewline switch is ignored if the command returns before reaching the end of the file.

JOBS:-

Lists the jobs that you are running in the background and in the foreground. If the prompt is returned with no information no jobs are present. Note: not all shells are capable of running this command.

Syntax

```
jobs [-p | -l] [-n] [-p] [-x] [job id]
```

-p | -l	Report the process group ID and working directory of the jobs.
-n	Display only jobs that have stopped or exited since last notified.
-p	Displays only the process IDs for the process group leaders of the selected jobs.
-x	Replace any job_id found in command or arguments with the corresponding process group ID, and then execute command passing it arguments.
job id	The job id.

Examples

```
jobs
```

Would display results similar to the below if jobs were running in the background.

```
[1] + Stopped (user)      man jobs
```

As you can see in the above job example the id is 1 it has been stopped by the user and the process in this case is man jobs (looking at the manual for jobs).

```
jobs -l
```

The above command would not just list the jobs running but also this group ID and the working directory of the jobs. Below is an example of what this would display.

```
[3] 16882 Running ./chsearchproc (wd: ~/public_html/cgi-bin/chsearch)
```

AWK:-

Originally based of the awk script processing language awk also known as oawk, gawk, mawk and nawk allows

Syntax

```
awk [program | -f program file] [flags / variables] [files]
```

-f program file	Run an awk script from the specified file instead of from the command line.

variable	Initializes the awk variable with the specified. Syntax is variable=value

The awk program can consist of one or more awk commands separated by a \n or semicolons.

Examples

awk myscript.sh

In the above example this command would execute the awk script "myscript.sh".

LN:-

Creates a link to a file.

Syntax

ln [-f] [-n] [-s] existingfile newname

-f	Link files without questioning the user, even if the mode of target forbids writing. This is the default if the standard input is not a terminal.
-n	Does not overwrite existing files.
-s	Makes it so that it does not create a symbolic link (not on System V.) existingfile - The file that you want to create a new link.
existingfile	Specifies file(s) that you want to create a link to.
newname	The new name of the file.
directory	The directory were you want the new link.

Examples

ln public_html/myfile.txt

Would create a hard link to myfile.txt in the current directory assuming that it is not public_html. Note: A link cannot be created to a file if currently in the directory where the file exists.

ln -s file1 file2

Creates a symbolic link to file1 with the name of the file2.

ENV:-

Displays your UNIX environment variables.

Syntax

```
env
```

Examples

env - Typing env at the prompt would give something similar to.

```
HOME=/computerhope/public_html
PATH=/usr/local/bin:
LOGNAME=admin
HZ=100
TERM=vt100
TZ=MST7MDT
SHELL=/bin/csh
MAIL=/var/mail/computerhope
_INIT_UTS_PLATFORM=SUNW,SPARCstation-10
_INIT_UTS_RELEASE=5.7
_INIT_UTS_SYSNAME=SunOS
_INIT_UTS_VERSION=Generic_106541-08
EDITOR=pico -t
OPENWINHOME=/usr/openwin
MANPATH=/usr/man:/usr/local/man:/usr/openwin/man
LD_LIBRARY_PATH=/usr/local/lib:/usr/openwin/lib
PAGER=more
```

Above is a wide variety of information. Below is brief description of each of the more commonly listed environment variables.

EDITOR	Name of editor used.
HOME	The directory that you are first logged into.
SHELL	The program you run as your command-line interpreter.
TERM	The type of terminal emulation used.
PATH	Listing of directories searched when logging on.
MAIL	Location of where the mail is stored
MANPATH	Location of your Manuals. See man command.
LOGNAME	The login name.
TZ	Time zone of computer

KILL:-

Cancels a job.

Syntax

```
kill [-s] [-l] %pid
```

-s	Specify the signal to send, using one of the symbolic names defined in the <signal.h> description. Values of signal will be recognized in a case independent fashion, without the SIG prefix. In addition, the symbolic name 0 will be recognized, representing the signal value zero. The corresponding signal will be sent instead of SIGTERM.
-l	Write all values of signal sup ported by the implementation, if no operand is given. If an exit_status operand is given and it is a value of the ? shell special parameter and wait corresponding to a process that was ter minated by a signal, the signal corresponding to the signal that terminated the process will be written. If an exit_status operand is given and it is the unsigned decimal integer value of a signal number, the signal corresponding to that signal will be written. Otherwise, the results are unspecified.
pid	One of the following: 1. A decimal integer specifying a process or process group to be signaled. The process or processes selected by positive, negative and zero values of the pid operand will be as described for the kill function. If process number 0 is specified, all processes in the process group are signaled. If the first pid operand is negative, it should be preceded by -- to keep it from being interpreted as an option. 2. A job control job ID that identifies a background process group to be signaled. The job control job ID notation is applicable only for invocations of kill in the current shell execution environment. Note the job control job ID type of pid is available only on systems supporting the job control option.

Examples

```
kill -s kill 100 -165
```

Kills job 1 of uid 165

When running the kill command you may receive the error "Operation not permitted", this is often encountered when you're killing the wrong group id (often 1,2,3 or low number jobs) that you don't have permission to kill. If you wish to see the group id of the background task run jobs -l

ALIAS:-

Commonly used for a long strings that are frequently used. Alias allows you to have a small more familiar command or name to execute a long string.

Syntax

alias [name=['command']]

name	Specifies the alias name.
command	Specifies the command the name should be an alias for.
-a	Removes all alias definitions from the current shell execution environment.
-t	Sets and lists tracked aliases.
-x	Sets or prints exported aliases. An exported alias is defined for scripts invoked by name.

Examples

alias home 'cd public_html' - Sets home to type cd public_html

to remove this alias use the unalias command

We recommend that ISP's or Unix systems with users who may be unfamiliar with Unix setup the following aliases:

clr clear
cls clear
copy cp -i
del rm -i
delete rm -i
dir ls -alg
home cd ~
ls ls -F
md mkdir
move mv -i
pwd echo $cwd
type more

DIFF:-

Displays two files and prints the lines that are different.

Syntax

diff [-b] [-i] [-t] [-w] [-c] [-C] [-e] [-f] [-h] [-n] [-D string] [-l] [-r] [-s] [-S name] [fileone filetwo] [directoryone directorytwo]

-b	Ignores spacing differences.
-i	Ignores case.
-t	Expands TAB characters in output lines. Normal or -c output adds

	character(s) to the front of each line that may adversely affect the indentation of the original source lines and make the output lines difficult to interpret. This option will preserve the original source's indentation.
-w	Ignores spaces and tabs.
-c	Produces a listing of differences with three lines of context. With this option output format is modified slightly: output begins with identification of the files involved and their creation dates, then each change is separated by a line with a dozen *'s. The lines removed from file1 are marked with '-'; those added to file2 are marked '+'. Lines that are changed from one file to the other are marked in both files with '!'.
-C	Produces a listing of differences identical to that produced by -c with number lines of context.
-e	Produces a script of only a, c, and d commands for the editor ed , which will recreate file2 from file1. In connection with -e, the following shell program may help maintain multiple versions of a file. Only an ancestral file ($1) and a chain of version-to-version ed scripts ($2,$3,...) made by diff need be on hand. A ``latest version'' appears on the standard output.

```
(shift; cat $*; echo '1,$p') | ed - $1
```

-f	Produces a similar script, not useful with ed , in the opposite order.
-h	Does a fast, half-hearted job. It works only when changed stretches are short and well separated, but does work on files of unlimited length. Options -c, -e, -f, and -n are unavailable with -h. diff does not descend into directories with this option.
-n	Produces a script similar to -e, but in the opposite order and with a count of changed lines on each insert or delete command.
-D string	Creates a merged version of file1 and file2 with C preprocessor controls included so that a compilation of the result without defining string is equivalent to compiling file1, while defining string will yield file2.
-l	Produce output in long format. Before the diff, each text file is piped through pr(1) to paginate it. Other differences are remembered and summarized after all text file differences are reported.
-r	Applies diff recursively to common subdirectories encountered.
-s	Reports files that are the identical; these would not otherwise be mentioned.
-S name	Starts a directory diff in the middle, beginning with the file name.
filenameone	File one for comparing.
filenametwo	File two for comparing.
directoryone	Directory one for comparing.
directorytwo	Directory two for comparing.

Examples

diff help dir2 - Compares the directory named help with the directory named dir2. Below is an example of the output when typing this command.

```
Only in help: tab2.gif
Only in help: tab3.gif
Only in help: tab4.gif
```

Only in help: tape.htm
Only in help: tbernoul.htm
Only in help: tconner.htm
Only in help: tempbus.psd

LOCATE:-

List files in databases that match a pattern.

Syntax

locate [-d path | --database=path] [-e | --existing] [-i | --ignore-case] [--version] [--help] pattern...

-d path --database=path	Instead of searching the default file name database, search the file name databases in path, which is a colon-separated list of database file names. You can also use the environment variable LOCATE_PATH to set the list of database files to search. The option overrides the environment variable if both are used. The file name database format changed starting with GNU find and locate version 4.0 to allow machines with different byte orderings to share the databases. This version of locate can automatically recognize and read databases produced for older versions of GNU locate or Unix versions of locate or find.
-e --existing	Only print out such names that currently exist (instead of such names that existed when the database was created). Note that this may slow down the program a lot, if there are many matches in the database.
-i --ignore-case	Ignore case distinctions in both the pattern and the file names.
--help	Print a summary of the options to locate and exit.
--version	Print the version number of locate and exit.

Examples

```
locate perl
```

In the above example the system would locate perl on the local machine.

Note: You may need to run the "updatedb" command to update the database in order to find the file you are searching for. This command should be ran anytime *nix is first installed or a major update occurs.

FIND:-

Finds one or more files assuming that you know their approximate filenames.

Syntax

find path expressions

path	A path name of a starting point in the directory hierarchy.
-atime n	True if the file was accessed n days ago. The access time of directories in path is changed by find itself.
-cpio device	Always true; write the current file on device in cpio format (5120-byte records).
-ctime n	True if the file's status was changed n days ago.
-depth	Always true; causes descent of the directory hierarchy to be done so that all entries in a directory are acted on before the directory itself. This can be useful when find is used with cpio to transfer files that are contain edin directories without write permission.
-exec command	True if the executed command returns a zero value as exit status. The end of command must be punctuated by an escaped semicolon. A command argument {} is replaced by the current path name.
-follow	Always true; causes symbolic links to be followed. When following symbolic links, find keeps track of the directories visited so that it can detect infinite loops; for example, such a loop would occur if a symbolic link pointed to an ancestor. This expression should not be used with the -type l expression.
-fstype type	True if the filesystem to which the file belongs is of type type .
-group gname	True if the file belongs to the group gname. If gname is numeric and does not appear in the /etc/group file, it is taken as a group ID.
-inum n	True if the file has inode number n.
-links	True if the file has n links.
-local	True if the file system type is not a remote file system type as defined in the /etc/dfs/fstypes file. nfsis used as the default remote filesystem type if the/etc/dfs/fstypes file is not present.
-ls	Always true; prints current path name together with its associated statistics. These include (respectively): inode number size in kilobytes (1024 bytes) protection mode number of hard links user

group

size in bytes

modification time.

If the file is a special file the size field will instead contain the major and minor device numbers.

If the file is a symbolic link the pathname of the linked-to file is printed preceded by `->'. The format is identical to that of ls -gilds ls Note: Formatting is done internally, without executing the ls program.

-mount	Always true; restricts the search to the file system containing the directory specified. Does not list mount points to other file systems.
-mtime n	True if the file's data was modified n days ago.
-name pattern	True if pattern matches the current file name. Normal shell file name generation characters (see sh) may be used. A backslash (\) is used as an escape character within the pattern. The pattern should be escaped or quoted when find is invoked from the shell.
-ncpio device	Always true; write the current file on device in cpio -c format (5120 byte records).
-newer file	True if the current file has been modified more recently than the argument file.
-nogroup	True if the file belongs to a group not in the /etc/group file.
-nouser	True if the file belongs to a user not in the /etc/passwd file.
-ok command	Like -exec except that the generated command line is printed with a question mark first, and is executed only if the user responds by typing y.
-perm [-]mode	The mode argument is used to represent file mode bits. It will be identical in format to the <symbolicmode> operand described in chmod, and will be interpreted as follows. To start, a template will be assumed with all file mode bits cleared. An op symbol of:

+	will set the appropriate mode bits in the template;
-	will clear the appropriate bits;
=	will set the appropriate mode bits, without regard to the contents of process' file mode creation mask.

The op symbol of - cannot be the first character of mode; this avoids ambiguity with the optional leading hyphen. Since the initial mode is all bits off, there are not any symbolic modes that need to use - as the first character.

If the hyphen is omitted, the primary will evaluate as true when the file permission bits exactly match the value of the resulting template.

Otherwise, if mode is prefixed by a hyphen, the primary will evaluate as true if at least all the bits in the resulting template are set in the file permission

	bits.
-perm [-]onum	True if the file permission flags exactly match the octal number onum see chmod). If onum is prefixed by a minus sign (-), only the bits that are set in onum are compared with the file permission flags, and the expression evaluates true if they match.
-print	Always true; causes the current path name to be printed.
-prune	Always yields true. Do not examine any directories or files in the directory structure below the pattern just matched. If -depth is specified, -prune will have no effect.
-size n[c]	True if the file is n blocks long (512 bytes per block). If n is followed by a c, the size is in bytes.
-type c	True if the type of the file is c, where c is b, c, d, D, f, l, p, or s for block special file, character special file, directory, door, plain file, symbolic link, fifo (named pipe), or socket, respectively.
-user uname	True if the file belongs to the user uname . If uname is numeric and does not appear as a login name in the /etc/passwd file, it is taken as a user ID.
-xdev	Same as the -mount primary.

When using find to determine files modified within a range of time, one must use the ?time argument before the -print argument; otherwise, find will give all files.

Examples

find -name 'mypage.htm'

In the above command the system would search for any file named mypage.htm in the current directory and any subdirectory.

find / -name 'mypage.htm'

In the above example the system would search for any file named mypage.htm on the root and all subdirectories from the root.

find -name 'file*'

In the above example the system would search for any file beginning with file in the current directory and any subdirectory.

find -name '*' -size +1000k

In the above example the system would search for any file that is larger then 1000k.

INFO:-

info - readable online documentation

DESCRIPTION

The Info file format is an easily-parsable representation for online documents. It can be read by emacs(1) and info(1) among other programs.

Info files are usually created from texinfo(5) sources by makeinfo(1), but can be created from scratch if so desired.

For a full description of the Texinfo language and associated tools, please see the Texinfo manual (written in Texinfo itself). Most likely, running this command from your shell:

```
info texinfo
```

or this key sequence from inside Emacs:

```
M-x info RET m texinfo RET
```

will get you there.

2. Explain basic command of Vi Editor

Despite its very limited ergonomics, Vi i is one of the most popular text editors texte under Unix type systems (with Emacs and pico). Under Linux, there is a free version of Vi called Vim (Vi Improved). Vi (pronounced vee-eye) is an editor that is fully in text mode, which means that all actions are carried out with the help of text commands. This editor, although it may appear of little practical use at first, is very powerful and can be very helpful in case the graphical interface malfunctions.

The syntax to launch Vi is as follows:

```
vi name_of_the_file
```

Once the file is open, you can move around by using cursors or the keys h, j, k and l (in case the keyboard does not have any arrow cursors).

Vi modes

Vi has three operating modes:

Regular mode: This is the mode you enter whenever you open a file. This mode allows typing commands

Insertion mode: This mode makes it possible to insert characters you capture inside of the document. To switch to insertion mode, just press the key Insert on your keyboard or, by default, the key i

Replacement mode: This mode allows you to replace existing text by the text you capture. Just hit r again to go to replacement mode and hit the key Esc to return to regular mode

Basic commands

Command	Description
:q	Quit the editor (without saving)
:q!	Forces the editor to quit without saving (even if changes were made to the document)
:wq	Saves the document and quits the editor
:file*name*	Saves the document under the specified name

Editing commands

Command	Description
x	Deletes the character that is currently under cursor
dd	Deletes the line that is currently under cursor
d*x*d	Deletes x lines starting with the one currently under the cursor
*n*x	Deletes n characters starting with the one currently under the cursor
x>>	Indents x lines to the right starting with the one currently under the cursor
x<<	Indents x lines to the left starting with the one currently under the cursor

Searching and replacing

To search for a word in a document, in regular mode, just type / followed by the chain of characters to be searched for and confirm by hitting the Enter key. Use the n key to go from occurrence to occurrence.
To replace a chain of characters by another on a line, you will find a very powerful command in Vi by using the regular expressions. Its syntax is as follows:

```
:s/chain_to_be_replaced/replacement_chain/
```

IThe replacement can be made throughout the entire document with the following syntax:

```
:%s/chain_to_be_replaced/replacement_chain/
```

Copy-paste and cut-paste

In Vi, it is possible to copy-paste a selection of lines. To do so, just type in the following command to copy n lines:

nyy

For example, the following command will copy 16 lines onto the clipboard:

16yy

To past the selection, just type the letter p. Cutting-pasting n lines is similar by using the command:

ndd

Then p to paste!

Cursor Movement (command mode):

		Scroll Backward 1 screen	**<ctrl>b**		
		Scroll Up 1/2 screen	**<ctrl>u**		
Go to beginning of line	**0**	Go to line n	**nG**	Go to end of line	**$**
		Scroll Down 1/2 screen	**<ctrl>d**	Go to line number ##	**:##**
		Scroll Forward 1 screen	**<ctrl>f**		
		Go to last line	**G**		
Scroll by sentence f/b	**( )**				
Scroll by word f/b	**w b**	Move left, down, up, right	**h j k l**	Left 6 chars	**6h**
		Directional Movement	**Arrow Keys**	Go to line #6	**6G**

Deleting text (command mode):

Change word	**cw**	Replace one character	**r**		
Delete word	**dw**	Delete text at cursor	**x**	Delete entire line (to buffer)	**dd**
		Delete current to end of line	**D**	Delete 5 lines (to buffer)	**5dd**
				Delete lines 5-10	**:5,10d**

Editing (command mode):

Copy line	**yy**	Copy n lines	**nyy**	Copy lines 1-2/paste after 3	**:1,2t 3**
Paste above current line	**P**				

Paste below current line	**P**			Move lines 4-5/paste after 6	**:4,5m 6**
				Join previous line	J
Search backward for string	?string	Search forward for string	/string	Find next string occurrence	n
% (entire file) s (search and replace) /old text with new/ c (confirm) g (global - all)	:%s/oldstring/newstring/cg			Ignore case during search	:set ic
Repeat last command	.	Undo previous command	u	Undo all changes to line	U

Save and Quit (command mode):

Save changes to buffer	**:w**	Save changes and quit vi	**:wq**	Save file to new file	**:w file**
		Quit without saving	**:q!**	Save lines to new file	**:10,15w file**

3. Explain I/O redirection, Piping and Command Substitution

   Grep “uvpce”< text.sh

   Cat f1 >> f2

   Cat > f1

   Ls –l | more

   A=`expr $a + $b`

4. What is Shell Script? Explain with Example.

shell script is a text file that contains a sequence of commands for a Unix-based operating system. It's called a shell script because it combines into a "script" in a single file a sequence of commands that would otherwise have to be presented to the system from a keyboard one at a time. The shell is the operating system's command interpreter and the set of commands you use to communicate with the system. A shell script is usually created for command sequences for which a user has a repeated need. You initiate the sequence of commands in the shell script by simply entering the name of the shell script on a command line.