| **Commands** | **Details** |
| --- | --- |
| **Basic operations** |  |
| ls | directory listing |
| ls -a | show hidden file |
| ls -l | show details for file |
| ln file1 file2 | create file2 (linked with file1). Changes will appear in both file |
| ln -s file1 file2 | create shortcut of file1 as file2 (soft link) |
| mkdir dirname | create directory dirname |
| cd dirname | go to dirname |
| cd | go to home directory |
| pwd | present working directory |
| rm filename | remove (delete) filename |
| rm -r dirname | remove directory dirname |
| rm -f filename | Force remove file filename |
| rm -rf dirname | Force remove directory dirname |
| cp file1 file2 | copy file1 to file2 |
| cp -r dir1 dir2 | copy dir1 to dir2 |
| mv file1 file2 | rename file1 to file2 |
| touch filename | create filename (if not exist) |
| cat > filename | add content to file (ctrl-c to exit) |
| cat >> filename | add content at the end of the filename |
| cat filename | show content of filename (all at once) |
| more filename | show content of filename (fit to screen with more option) |
| head filename | show first 10 lines of filename |
| tail filename | show last 10 lines of filename |
| tail -f filename | last 10 lines of the filename (useful for log files) |
| command > filename | write output of command in the filename (overwrite the fie |
| command >> filname | append the output at the end of file name |
| wc filename (wc -l, -w, -c) | returns number of lines, word, character and filename. |
| chmod 777 filename | read(4), write(2) and execute(1) permission to all [4+2+1=7]  chmod [owner, group, others] filename |

| **pr -x** | **Divides the file into x columns** |
| --- | --- |
| **pr -h** | **Assigns a header to the file** |
| **pr -n** | **Denotes the file with Line Numbers** |
| **lp -nc**  **lpr c** | **Prints “c” copies of the File** |
| **lp -d**  **lpr -P** | **Specifies name of the printer** |
| **apt-get** | **Command used to install and update packages** |
| **mail -s ‘subject’ -c ‘cc-address’ -b ‘bcc-address’ ‘to-address’** | **Command to send email** |
| **mail -s “Subject” to-address < Filename** | **Command to send email with attachment** |

**Shows the location of a file**

whereis filename

**Shows the location of a file if it is in your PATH**

**which filename**

### 

### **Paste**

| **Commands** | **Details** |
| --- | --- |
| **Paste** |  |
| **paste file1 file2 file3** | **paste the lines of file2, then file3 beside the lines of file1** |
| **paste -d’,’ file1 file2** | **put comma at the end of each file content** |
| **paste -s filename** | **merge all lines of filename** |

### 

### **Sort**

| **Commands** | **Details** |
| --- | --- |
| **Sort** |  |
| sort filename | sort the lines of the filename |
| sort -u filename | sort and eliminate duplicates |
| sort -r filename | reverse order sort |
| sort file1 -o file2  sort file1 > file2 | sort data of file1 and save to file2 |
| sort -n filename | sort data based on numeric value i.e. 12 > 4 |
| sort -M fielname | sort data with month name i.e. Jan, Feb, Mar etc. |

### 

### 

### **Cut**

| **Commands** | **Details** |
| --- | --- |
| **Cut** |  |
| cut d”-” 1,4 filename | cut the 1st and 4th column of filename, where - is delimiter |
| cut -c3- filename | remove first 2 character from each line and printed the rest |
| cut -c-3-8 filename | print character 3-8 from each line |
| cut -c1-3,6-8,10- filename | print character 1-3, skip 4, print 6-8, skip 9, print 10 to end from each line |
| cut -d’,’ -f3 filename | extact field-3 from each line with ‘,’ as delimiter |
| cut -d’,’ -f3,6 filename | extact field-3-to-6 from each line with ‘,’ as delimiter |

### **Grep**

| **Commands** | **Details** |
| --- | --- |
| **grep** |  |
| grep pattern filenames | print the lines with pattern in filenames |
| grep ‘[0-9]’ filenames | print the lines from filenames which have numbers 0-9 (regular expression) |
| grep -v pattern filenames | print all lines which do not contain pattern |
| grep -l pattern filenames | print the filenames which have pattern |
| grep -n pattern filenames | print the line numbers as well |

### **Sed**

| **Commands** | **Details** |
| --- | --- |
| **sed** |  |
| sed ‘s/word1/word2/’ filename | replace word1 with word2 in filename |
| sed ‘s/word1/word2/’ file1 > file2  mv file2 > file1 | use two steps to make change permanent |
| sed -n ‘2,4p’ filename | print line 2 to 4 of filename |
| sed -n /word1/p’ filename | print lines which contain word1 in filename |

# Managing FirewallD

Start and Enable Firewalld

# systemctl start firewalld

# systemctl enable firewalld

Stop and Disable Firewalld

# systemctl stop firewalld

# systemctl disable firewalld

Checking the status of Firewalld

# firewall-cmd --state

View the status of the FirewallD daemon

# systemctl status firewalld

To reload a FirewallD configuration:

# firewall-cmd --reload

# Firewall Zones

To view the default zone:

# firewall-cmd --get-default-zone

Changing the default Zone of firewalld

# firewall-cmd --set-default-zone=internal

View the Zones in use

# firewall-cmd --get-active-zones

Get configurations for all zones

# firewall-cmd --list-all-zones

# Using Services

View default available services

# firewall-cmd --get-services

Enable a service

# firewall-cmd --zone=public --add-service=http --permanent

Disable the HTTP service

# firewall-cmd --zone=public --remove-service=http --permanent

Allowing or Denying an Arbitrary Port/Protocol

# firewall-cmd --zone=public --add-port=12345/tcp --permanent

# firewall-cmd --zone=public --remove-port=12345/tcp --permanent

# Port Forwarding

Forward traffic to port on same Server

# firewall-cmd --zone="public" --add-forward-port=port=80:proto=tcp:toport=12345

To remove the rules, replace '--add' with '--remove.'

# firewall-cmd --zone=public --remove-masquerade

# Constructing a Ruleset with FirewallD

Let us assign the DMZ as the default zone to eth0 as it allows only SSH and ICMP.

# firewall-cmd --set-default-zone=dmz

# firewall-cmd --zone=dmz --add-interface=eth0

Permanent rule for HTTP and HTTPS

Add permanent service rules for HTTP and HTTPS to the dmz zone:

# firewall-cmd --zone=dmz --add-service=http --permanent

# firewall-cmd --zone=dmz --add-service=https --permanent

Reload FirewallD so the rules take effect immediately:

# firewall-cmd --reload

If you now run

#firewall-cmd --zone=dmz --list-all

# Permission

**Read** (r) : The read permission allows you to open and read the content of a file. But you can't do any editing or modification in the file.

**Write** (w) : The write permission allows you to edit, remove or rename a file. For instance, if a file is present in a directory, and write permission is set on the file but not on the directory, then you can edit the content of the file but can't remove, or rename it.

**Execute** (x): In Unix type system, you can't run or execute a program unless execute permission is set.But in Windows, there is no such permission available.

| **Permission** | **On a File** | **On a directory** |
| --- | --- | --- |
| r (read) | read file content (cat) | read directory content (ls) |
| w (write) | change file content (vi) | create file in directory (touch) |
| x (execute) | execute the file | enter the directory (cd) |

-rwxr--r--: A regular file whose user class has read/write/execute, group class has only read permissions, other class has only read permissions

drw-rw-r--: A directory whose user class has read/write permissions, group class has read/write permissions, other class has only read permissions

crwxrw-r--: A character special file whose user has read/write/execute permissions, group class has read/write permissions, other class has only read permissions

It is easy to think of the permission settings as a series of bits (which is how the computer thinks about them). Here's how it works:

rwx rwx rwx = 111 111 111

rw- rw- rw- = 110 110 110

rwx --- --- = 111 000 000

Where

rwx = 111 in binary = 7

rw- = 110 in binary = 6

r-x = 101 in binary = 5

r-- = 100 in binary = 4

| **Value** | **Meaning** |
| --- | --- |
| 777 | (rwxrwxrwx) No restrictions on permissions. Anybody may list files, create new files in the directory and delete files in the directory. Generally not a good setting. |
| 755 | (rwxr-xr-x) The directory owner has full access. All others may list the directory, but cannot create files nor delete them. This setting is common for directories that you wish to share with other users. |
| 700 | (rwx------) The directory owner has full access. Nobody else has any rights. This setting is useful for directories that only the owner may use and must be kept private from others. |
| 666 | (rw-rw-rw-) All users may read and write the file. |
| 644 | (rw-r--r--) The owner may read and write a file, while all others may only read the file. A common setting for data files that everybody may read, but only the owner may change. |
| 600 | (rw-------) The owner may read and write a file. All others have no rights. A common setting for data files that the owner wants to keep private. |

## The df Command

The first way to manage your partition space is with the df (disk free) command. The command df -k (disk free) displays the disk space usage in kilobytes

df -k , df -h , df -sh

## The du Command

The du (disk usage) command enables you to specify directories to show disk space usage on a particular directory.

du /etc

The -h option makes the output easier to comprehend −

du -h /etc

## 

## Mounting the File System

mount -t file\_system\_type device\_to\_mount directory\_to\_mount\_to

For example, if you want to mount a CD-ROM to the directory /mnt/cdrom, you can type −

mount -t iso9660 /dev/cdrom /mnt/cdrom

## Unmounting the File System

For example, to unmount cdrom, use the following command −

umount /dev/cdrom

## Managing Users and Groups

There are four main user administration files −

* /etc/passwd − Keeps the user account and password information. This file holds the majority of information about accounts on the Unix system.
* /etc/shadow − Holds the encrypted password of the corresponding account. Not all the systems support this file.
* /etc/group − This file contains the group information for each account.
* /etc/gshadow − This file contains secure group account information.

| **Sr.No.** | **Command & Description** |
| --- | --- |
| 1 | useradd  Adds accounts to the system |
| 2 | usermod  Modifies account attributes |
| 3 | userdel  Deletes accounts from the system |
| 4 | groupadd  Adds groups to the system |
| 5 | groupmod  Modifies group attributes |
| 6 | groupdel  Removes groups from the system |

## 

## Create a Group

groupadd [-g gid [-o]] [-r] [-f] groupname

groupadd developers

The following table lists out the parameters −

| **Sr.No.** | **Option & Description** |
| --- | --- |
| 1 | -g GID  The numerical value of the group's ID |
| 2 | -o  This option permits to add group with non-unique GID |
| 3 | -r  This flag instructs groupadd to add a system account |
| 4 | -f  This option causes to just exit with success status, if the specified group already exists. With -g, if the specified GID already exists, other (unique) GID is chosen |
| 5 | groupname  Actual group name to be created |

## Modify a Group

To modify a group, use the groupmod syntax −

groupmod -n new\_modified\_group\_name old\_group\_name

To change the developers\_2 group name to developer, type −

groupmod -n developer developer\_2

Here is how you will change the financial GID to 545 −

groupmod -g 545 developer

## Delete a Group

groupdel developer

## Create an Account

Let us see how to create a new account on your Unix system. Following is the syntax to create a user's account −

useradd -d homedir -g groupname -m -s shell -u userid accountname

The following table lists out the parameters −

| **Sr.No.** | **Option & Description** |
| --- | --- |
| 1 | -d homedir  Specifies home directory for the account |
| 2 | -g groupname  Specifies a group account for this account |
| 3 | -m  Creates the home directory if it doesn't exist |
| 4 | -s shell  Specifies the default shell for this account |
| 5 | -u userid  You can specify a user id for this account |
| 6 | accountname  Actual account name to be created |

## Performance Tools

Unix provides following important tools to measure and fine tune Unix system performance −

| **Sr.No.** | **Command & Description** |
| --- | --- |
| 1 | nice/renice  Runs a program with modified scheduling priority |
| 2 | netstat  Prints network connections, routing tables, interface statistics, masquerade connections, and multicast memberships |
| 3 | time  Helps time a simple command or give resource usage |
| 4 | uptime  This is System Load Average |
| 5 | ps  Reports a snapshot of the current processes |
| 6 | vmstat  Reports virtual memory statistics |
| 7 | gprof  Displays call graph profile data |
| 8 | prof  Facilitates Process Profiling |
| 9 | top  Displays system tasks |

## Important Log Locations

All the system applications create their log files in */var/log* and its sub-directories. Here are few important applications and their corresponding log directories −

| **Application** | **Directory** |
| --- | --- |
| httpd | /var/log/httpd |
| samba | /var/log/samba |
| cron | /var/log/ |
| mail | /var/log/ |
| mysql | /var/log/ |

## Compressed Files

Files may be compressed to save space. Compressed files can be created and examined.

| **Sr.No.** | **Command & Description** |
| --- | --- |
| 1 | compress  Compresses files |
| 2 | gunzip  Helps uncompress gzipped files |
| 3 | gzip  GNU alternative compression method |
| 4 | uncompress  Helps uncompress files |
| 5 | unzip  List, test and extract compressed files in a ZIP archive |
| 6 | zcat  Cat a compressed file |
| 7 | zcmp  Compares compressed files |
| 8 | zdiff  Compares compressed files |
| 9 | zmore  File perusal filter for crt viewing of compressed text |

## Getting Information

Various Unix manuals and documentation are available on-line. The following Shell commands give information −

| **Sr.No.** | **Command & Description** |
| --- | --- |
| 1 | apropos  Locates commands by keyword lookup |
| 2 | info  Displays command information pages online |
| 2 | man  Displays manual pages online |
| 3 | whatis  Searches the whatis database for complete words |
| 4 | yelp  GNOME help viewer |

## Network Communication

These following commands are used to send and receive files from a local Unix hosts to the remote host around the world.

| **Sr.No.** | **Command & Description** |
| --- | --- |
| 1 | ftp  File transfer program |
| 2 | rcp  Remote file copy |
| 3 | rlogin  Remote login to a Unix host |
| 4 | rsh  Remote shell |
| 5 | tftp  Trivial file transfer program |
| 6 | telnet  Makes terminal connection to another host |
| 7 | ssh  Secures shell terminal or command connection |
| 8 | scp  Secures shell remote file copy |
| 9 | sftp  Secures shell file transfer program |

Some of these commands may be restricted at your computer for security reasons.

## Messages between Users

The Unix systems support on-screen messages to other users and world-wide electronic mail −

| **Sr.No.** | **Command & Description** |
| --- | --- |
| 1 | evolution  GUI mail handling tool on Linux |
| 2 | mail  Simple send or read mail program |
| 3 | mesg  Permits or denies messages |
| 4 | parcel  Sends files to another user |
| 5 | pine  Vdu-based mail utility |
| 6 | talk  Talks to another user |
| 7 | write  Writes message to another user |

# Unix - File Management

In Unix, there are three basic types of files −

* Ordinary Files − An ordinary file is a file on the system that contains data, text, or program instructions. In this tutorial, you look at working with ordinary files.
* Directories − Directories store both special and ordinary files. For users familiar with Windows or Mac OS, Unix directories are equivalent to folders.
* Special Files − Some special files provide access to hardware such as hard drives, CD-ROM drives, modems, and Ethernet adapters. Other special files are similar to aliases or shortcuts and enable you to access a single file using different names.

In the ls -l listing example, every file line begins with a d, -, or l. These characters indicate the type of the file that's listed.

| **Sr.No.** | **Prefix & Description** |
| --- | --- |
| 1 | -  Regular file, such as an ASCII text file, binary executable, or hard link. |
| 2 | b  Block special file. Block input/output device file such as a physical hard drive. |
| 3 | c  Character special file. Raw input/output device file such as a physical hard drive. |
| 4 | d  Directory file that contains a listing of other files and directories. |
| 5 | l  Symbolic link file. Links on any regular file. |
| 6 | p  Named pipe. A mechanism for interprocess communications. |
| 7 | s  Socket used for interprocess communication. |

## Hidden Files

An invisible file is one, the first character of which is the dot or the period character (.). Unix programs (including the shell) use most of these files to store configuration information.

Some common examples of the hidden files include the files −

* .profile − The Bourne shell ( sh) initialization script
* .kshrc − The Korn shell ( ksh) initialization script
* .cshrc − The C shell ( csh) initialization script
* .rhosts − The remote shell configuration file
* Single dot (.) − This represents the current directory.
* Double dot (..) − This represents the parent directory.

## Creating Files

You can use the vi editor to create ordinary files on any Unix system. You simply need to give the following command −

$ vi filename

## Display Content of a File

You can use the cat command to see the content of a file. Following is a simple example to see the content of the above created file −

$ cat filename

You can display the line numbers by using the -b option along with the cat command as follows −

$ cat -b filename

## Counting Words in a File

You can use the wc command to get a count of the total number of lines, words, and characters contained in a file. Following is a simple example to see the information about the file created above −

$ wc filename

## 

## 

## Copying Files

To make a copy of a file use the cp command. The basic syntax of the command is −

$ cp source\_file destination\_file

Following is the example to create a copy of the existing file filename.

$ cp filename copyfile

## Renaming Files

To change the name of a file, use the mv command. Following is the basic syntax −

$ mv old\_file new\_file

The mv (move) command can also be used to rename a directory. The syntax is as follows −

$mv olddir newdir

## Standard Unix Streams

Under normal circumstances, every Unix program has three streams (files) opened for it when it starts up −

* stdin − This is referred to as the *standard input* and the associated file descriptor is 0. This is also represented as STDIN. The Unix program will read the default input from STDIN.
* stdout − This is referred to as the *standard output* and the associated file descriptor is 1. This is also represented as STDOUT. The Unix program will write the default output at STDOUT
* stderr − This is referred to as the *standard error* and the associated file descriptor is 2. This is also represented as STDERR. The Unix program will write all the error messages at STDERR.

## Using chmod with Absolute Permissions

| **Number** | **Octal Permission Representation** | **Ref** |
| --- | --- | --- |
| 0 | No permission | --- |
| 1 | Execute permission | --x |
| 2 | Write permission | -w- |
| 3 | Execute and write permission: 1 (execute) + 2 (write) = 3 | -wx |
| 4 | Read permission | r-- |
| 5 | Read and execute permission: 4 (read) + 1 (execute) = 5 | r-x |
| 6 | Read and write permission: 4 (read) + 2 (write) = 6 | rw- |
| 7 | All permissions: 4 (read) + 2 (write) + 1 (execute) = 7 | rwx |

## PS1 and PS2 Variables

There are quite a few escape sequences that can be used as value arguments for PS1; try to limit yourself to the most critical so that the prompt does not overwhelm you with information.

| **Sr.No.** | **Escape Sequence & Description** |
| --- | --- |
| 1 | \t  Current time, expressed as HH:MM:SS |
| 2 | \d  Current date, expressed as Weekday Month Date |
| 3 | \n  Newline |
| 4 | \s  Current shell environment |
| 5 | \W  Working directory |
| 6 | \w  Full path of the working directory |
| 7 | \u  Current user’s username |
| 8 | \h  Hostname of the current machine |
| 9 | \#  Command number of the current command. Increases when a new command is entered |
| 10 | \$  If the effective UID is 0 (that is, if you are logged in as root), end the prompt with the # character; otherwise, use the $ sign |

## 

## 

## Environment Variables

Following is the partial list of important environment variables. These variables are set and accessed as mentioned below −

| **Sr.No.** | **Variable & Description** |
| --- | --- |
| 1 | DISPLAY  Contains the identifier for the display that X11 programs should use by default. |
| 2 | HOME  Indicates the home directory of the current user: the default argument for the cd built-in command. |
| 3 | IFS  Indicates the Internal Field Separator that is used by the parser for word splitting after expansion. |
| 4 | LANG  LANG expands to the default system locale; LC\_ALL can be used to override this. For example, if its value is pt\_BR, then the language is set to (Brazilian) Portuguese and the locale to Brazil. |
| 5 | LD\_LIBRARY\_PATH  A Unix system with a dynamic linker, contains a colonseparated list of directories that the dynamic linker should search for shared objects when building a process image after exec, before searching in any other directories. |
| 6 | PATH  Indicates the search path for commands. It is a colon-separated list of directories in which the shell looks for commands. |
| 7 | PWD  Indicates the current working directory as set by the cd command. |
| 8 | RANDOM  Generates a random integer between 0 and 32,767 each time it is referenced. |
| 9 | SHLVL  Increments by one each time an instance of bash is started. This variable is useful for determining whether the built-in exit command ends the current session. |
| 10 | TERM  Refers to the display type. |
| 11 | TZ  Refers to Time zone. It can take values like GMT, AST, etc. |
| 12 | UID  Expands to the numeric user ID of the current user, initialized at the shell startup. |

### The lpstat and lpq Commands

The lpstat command shows what's in the printer queue: request IDs, owners, file sizes, when the jobs were sent for printing, and the status of the requests.

Use lpstat -o if you want to see all output requests other than just your own. Requests are shown in the order they'll be printed −

The lpq gives slightly different information than lpstat -o −

## 

## 

## 

## Sending Email

You use the Unix mail command to send and receive mail. Here is the syntax to send an email −

$mail [-s subject] [-c cc-addr] [-b bcc-addr] to-addr

Here are important options related to mail command −s

| **Sr.No.** | **Option & Description** |
| --- | --- |
| 1 | -s  Specifies subject on the command line. |
| 2 | -c  Sends carbon copies to the list of users. List should be a commaseparated list of names. |
| 3 | -b  Sends blind carbon copies to list. List should be a commaseparated list of names. |

Following is an example to send a test message to admin@yahoo.com.

$mail -s "Test Message" admin@yahoo.com

You are then expected to type in your message, followed by "control-D" at the beginning of a line. To stop, simply type dot (.) as follows −

Hi,

This is a test

.

Cc:

You can send a complete file using a redirect < operator as follows −

$mail -s "Report 05/06/07" admin@yahoo.com < demo.txt

To check incoming email at your Unix system, you simply type email as follows −

$mail

no email

## 

## 

## The ftp Utility

Here, ftp stands for File Transfer Protocol. This utility helps you upload and download your file from one computer to another computer.

The ftp utility has its own set of Unix-like commands. These commands help you perform tasks such as −

* Connect and login to a remote host.
* Navigate directories.
* List directory contents.
* Put and get files.
* Transfer files as ascii, ebcdic or binary.

### Syntax

Following is the simple syntax to use the ftp command −

$ftp hostname or ip-address

The following tables lists out a few important commands −

| **Sr.No.** | **Command & Description** |
| --- | --- |
| 1 | put filename  Uploads filename from the local machine to the remote machine. |
| 2 | get filename  Downloads filename from the remote machine to the local machine. |
| 3 | mput file list  Uploads more than one file from the local machine to the remote machine. |
| 4 | mget file list  Downloads more than one file from the remote machine to the local machine. |
| 5 | prompt off  Turns the prompt off. By default, you will receive a prompt to upload or download files using mput or mget commands. |
| 6 | prompt on  Turns the prompt on. |
| 7 | dir  Lists all the files available in the current directory of the remote machine. |
| 8 | cd dirname  Changes directory to dirname on the remote machine. |
| 9 | lcd dirname  Changes directory to dirname on the local machine. |
| 10 | quit  Helps logout from the current login. |

## Word and Character Searching

These two commands differ only in the direction where the search takes place −

* The / command searches forwards (downwards) in the file.
* The ? command searches backwards (upwards) in the file.

These characters must be preceded by a backslash (\) to be included as part of the search expression.

| **Sr.No.** | **Character &Description** |
| --- | --- |
| 1 | ^  Searches at the beginning of the line (Use at the beginning of a search expression). |
| 2 | .  Matches a single character. |
| 3 | \*  Matches zero or more of the previous character. |
| 4 | $  End of the line (Use at the end of the search expression). |
| 5 | [  Starts a set of matching or non-matching expressions. |
| 6 | <  This is put in an expression escaped with the backslash to find the ending or the beginning of a word. |
| 7 | >  This helps see the '<' character description above. |

## 

## 

## Set Commands

You can change the look and feel of your vi screen using the following :set commands. Once you are in the command mode, type :set followed by any of the following commands.

| **Sr.No.** | **Command & Description** |
| --- | --- |
| 1 | :set ic  Ignores the case when searching |
| 2 | :set ai  Sets autoindent |
| 3 | :set noai  Unsets autoindent |
| 4 | :set nu  Displays lines with line numbers on the left side |
| 5 | :set sw  Sets the width of a software tabstop. For example, you would set a shift width of 4 with this command — :set sw = 4 |
| 6 | :set ws  If *wrapscan* is set, and the word is not found at the bottom of the file, it will try searching for it at the beginning |
| 7 | :set wm  If this option has a value greater than zero, the editor will automatically "word wrap". For example, to set the wrap margin to two characters, you would type this: :set wm = 2 |
| 8 | :set ro  Changes file type to "read only" |
| 9 | :set term  Prints terminal type |
| 10 | :set bf  Discards control characters from input |

## 

## 

## Shell Types

In Unix, there are two major types of shells −

* Bourne shell − If you are using a Bourne-type shell, the $ character is the default prompt.
* C shell − If you are using a C-type shell, the % character is the default prompt.

The Bourne Shell has the following subcategories −

* Bourne shell (sh)
* Korn shell (ksh)
* Bourne Again shell (bash)
* POSIX shell (sh)

The different C-type shells follow −

* C shell (csh)
* TENEX/TOPS C shell (tcsh)

# Unix - Special Variables

The following table shows a number of special variables that you can use in your shell scripts −

| **Sr.No.** | **Variable & Description** |
| --- | --- |
| 1 | $0  The filename of the current script. |
| 2 | $n  These variables correspond to the arguments with which a script was invoked. Here n is a positive decimal number corresponding to the position of an argument (the first argument is $1, the second argument is $2, and so on). |
| 3 | $#  The number of arguments supplied to a script. |
| 4 | $\*  All the arguments are double quoted. If a script receives two arguments, $\* is equivalent to $1 $2. |
| 5 | $@  All the arguments are individually double quoted. If a script receives two arguments, $@ is equivalent to $1 $2. |
| 6 | $?  The exit status of the last command executed. |
| 7 | $$  The process number of the current shell. For shell scripts, this is the process ID under which they are executing. |
| 8 | $!  The process number of the last background command. |

# Unix - Regular Expressions with SED

| **Sr.No.** | **Range & Description** |
| --- | --- |
| 1 | p  Prints the line |
| 2 | d  Deletes the line |
| 3 | s/pattern1/pattern2/  Substitutes the first occurrence of pattern1 with pattern2 |

## Deleting All Lines with sed

cat /etc/passwd | sed 'd'

The following command does exactly the same as in the previous example, without the cat command −

$ sed -e 'd' /etc/passwd

$sed 's/Country/India/g' sample.txt

## **Introduction to Crontab in Unix**

The syntax of Crontab is as follows:

minute(s) hour(s) day(s) month(s) weekday(s) command(s)

* Crontab provides six fields. The schedule for the execution of the statement is contained in the first five fields and hence they are integer fields.
* As the name describes, the minute field holds the value in the range 0-59 showing the amount of time in minutes that the command is executed.
* As the name describes, the hour field holds the value in the range 0-23 showing the amount of time in hours that the command is executed.
* As the name describes, the day field holds the value in the range 1-31 showing the amount of time in days that the command is executed.
* As the name describes, the month field holds the value in the range 1-12 showing the amount of time in months that the command is executed.
* As the name describes, the weekday field holds the value in the range 0-6 showing the amount of time in days that the command is executed.
* The Bourne shell command is the sixth field that is to be executed.
* Using an asterisk (\*) in the first five fields denotes that all possible values are allowed.
* Using a comma (,) denotes a list of values.
* Using a hyphen (-) denotes a range of values.
* Using a separator (/) denotes a step value.

The following options can be used with the crontab command:

* Crontab -a filename: The crontab file is the filename to be installed.
* Crontab -e: This option is used in the crontab file is to be edited. If a crontab file doesn’t already exist, a new file is created.
* Crontab -l: This option displays the crontab file.
* Crontab -r: This option removes the crontab file.
* Crontab -v: The time at which the crontab file was edited the last time is displayed.

### **Ten VALUABLE UNIX Commands**

Once you have mastered the basic UNIX commands, these will be quite valuable in managing your own account.

| **Command** | **Example** | **Description** |
| --- | --- | --- |
| 1.  **grep <str><files>** | grep "bad word" \* | Find which files contain a certain word |
| 2.  **chmod <opt> <file>** | chmod 644 \*.html  chmod 755 file.exe | Change file permissions read only  Change file permissions to executable |
| 3.  **passwd** | passwd | Change passwd |
| 4.  **ps <opt>** | ps aux  ps aux  **|** grep dhyatt | List all running processes by #ID  List process #ID's running by dhyatt |
| 5.  **kill <opt> <ID>** | kill -9 8453 | Kill process with ID #8453 |
| 6.  **gcc (g++) <source>** | gcc file.c -o file  g++ fil2.cpp -o fil2 | Compile a program written in C  Compile a program written in C++ |
| 7.  **gzip <file>** | gzip bigfile  gunzip bigfile.gz | Compress file  Uncompress file |
| 8.  **mail**  **(pine)** | mail me@tjhsst.edu **<** file1  pine | Send file1 by email to someone  Read mail using pine |
| 9.  **telnet <host>**  **ssh <host>** | telnet vortex.tjhsst.edu  ssh -l dhyatt jazz.tjhsst.edu | Open a connection to vortex  Open a secure connection to jazz as user dhyatt |
| 10.  **ftp <host>**  **ncftp <host/directory>** | ftp station1.tjhsst.edu  ncftp metalab.unc.edu | Upload or Download files to station1  Connect to archives at UNC |

### **Ten FUN UNIX Commands**

These are ten commands that you might find interesting or amusing. They are actually quite helpful at times, and should not be considered idle entertainment.

| **Command** | **Example** | **Description** |
| --- | --- | --- |
| 1.  **who** | who | Lists who is logged on your machine |
| 2.  **finger** | finger | Lists who is on computers in the lab |
| 3.  **ytalk <user@place>** | ytalk dhyatt@threat | Talk online with dhyatt who is on threat |
| 4.  **history** | history | Lists commands you've done recently |
| 5.  **fortune** | fortune | Print random humerous message |
| 6.  **date** | date | Print out current date |
| 7.  **cal <mo> <yr>** | cal 9 2000 | Print calendar for September 2000 |
| 8.  **xeyes** | xeyes & | Keep track of cursor (in "background") |
| 9.  **xcalc** | xcalc & | Calculator ("background" process) |
| 10.  **mpage <opt> <file>** | mpage -8 file1 **|** lpr | Print 8 pages on a single sheet and send to printer (the font will be small!) |

### **Ten HELPFUL UNIX Commands**

These ten commands are very helpful, especially with graphics and word processing type applications.

| **Command** | **Example** | **Description** |
| --- | --- | --- |
| 1.  **netscape** | netscape & | Run Netscape browser |
| 2.  **xv** | xv & | Run graphics file converter |
| 3.  **xfig / xpaint** | xfig & (xpaint &) | Run drawing program |
| 4.  **gimp** | gimp & | Run photoshop type program |
| 5.  **ispell <fname>** | ispell file1 | Spell check file1 |
| 6.  **latex <fname>** | latex file.tex | Run LaTeX, a scientific document tool |
| 7.  **xemacs / pico** | xemacs (or pico) | Different editors |
| 8.  **soffice** | soffice & | Run StarOffice, a full word processor |
| 9.  **m-tools (mdir, mcopy,**  **mdel, mformat, etc. )** | mdir a:  mcopy file1 a: | DOS commands from UNIX (dir A:)  Copy file1 to A: |
| 10.  **gnuplot** | gnuplot | Plot data graphically |

### **Ten USEFUL UNIX Commands:**

These ten commands are useful for monitoring system access, or simplifying your own environment.

| **Command** | **Example** | **Description** |
| --- | --- | --- |
| 1.  **df** | df | See how much free disk space |
| 2.  **du** | du -b subdir | Estimate disk usage of directory in Bytes |
| 3.  **alias** | alias lls="ls -alF" | Create new command "lls" for long format of ls |
| 4.  **xhost** | xhost + threat.tjhsst.edu  xhost - | Permit window to display from x-window program from threat  Allow no x-window access from other systems |
| 5.  **fold** | fold -s file1  **|** lpr | Fold or break long lines at 60 characters and send to printer |
| 6.  **tar** | tar -cf subdir.tar subdir  tar -xvf subdir.tar | Create an archive called subdir.tar of a directory  Extract files from an archive file |
| 7.  **ghostview (gv)** | gv filename.ps | View a Postscript file |
| 8.  **ping**  **(traceroute)** | ping threat.tjhsst.edu  traceroute www.yahoo.com | See if machine is alive  Print data path to a machine |
| 9.  **top** | top | Print system usage and top resource hogs |
| 10.  **logout (exit)** | logout or exit | How to quit a UNIX shell. |

### **Find Command in Unix**

**Syntax:**

find [options] [paths] [expression]

* Option expressions are used to constrain the find operation, and always return true.
  + - -depth: process the directory contents before processing the directory itself.
    - -maxdepth: the max levels below the provided paths to descend for a match.
    - -mindepth: the min levels beyond the provided paths to descend before matching.
* Test expressions are used to evaluate specific properties of the files and return true or false accordingly. (Wherever a count ‘n’ is used: without any prefix the match is for the exact value of n; with a ‘+’ prefix, the match is for values greater than n; and with a ‘-‘ prefix, the match is for values lesser than n.)
  + - -atime n: Returns true if the file was accessed n days ago.
    - -ctime n: Returns true if the file’s status was changed n days ago.
    - -mtime n: Returns true if the file’s contents were modified n days ago.
    - -name pattern: Returns true if the file’s name matches the provided shell pattern.
    - -iname pattern: Returns true if the file’s name matches the provided shell pattern. The matching here is case insensitive.
    - -path pattern: Returns true if the file’s name with the path matches the shell pattern.
    - -regex pattern: Returns true if the file’s name with the path matches the regular expression.
    - -size n: Returns true if the file size is n blocks.
    - -perm – mode: Returns true if all the permission bits for mode are set for the file.
    - -type c: Returns true if the file is of type c (e.g. ‘b’ for block device file, ‘d’ for directory etc.).
    - -username: Returns true if the file is owned by username ‘name’.
* The action expressions are used to define actions that have side effects and may return true or false. If not actions are specified, the ‘-print’ action is performed for all matching files.
  + - -delete: Delete the matched file, and return true if successful.
    - -exec command: Execute the given command for each matching file, and return true if the return value is 0.
    - -ok command: Like the ‘exec’ expression, but confirms with the user first.
    - -ls: List the matching file as the per ‘ls -dils’ format.
    - -print: Print the name of the matching file.
    - -prune: If the file is a directory, do not descend into it, and return true.
* The expression is evaluated from left to right and is put together using the following operators.
  + - **\(** expr **\)**: Used to force precedence.
    - **!** expr: Used to negate an expression.
    - expr1 **-a** expr2: The result is an ‘and’ of the two expressions. The expr2 is only evaluated of expr1 is true.
    - expr1 expr2: The ‘and’ operator is implicit in this case.
    - expr1 **-o** expr2: The result is an ‘or’ of the two expressions. The expr2 is only evaluated of expr1 is false.

**List all files found in the current directory and its hierarchy**

$ find.

**List all files found in the current hierarchy, and all the hierarchy below /home/xyz**

$ find. /home/XYZ

**Search for a file by the name abc in the current directory and its hierarchy**

$ find ./ -name abc

**Search for a directory by the name xyz in the current directory and its hierarchy**

$ find ./ -type d -name xyz

**Search for a file by the name abc.txt below the current directory, and prompt the user to delete each match.**

Note that the “{}” string is substituted by the actual file name while running and that the “\;” string is used to terminate the command to be executed.

$ find ./ -name abc.txt -exec rm -i {} \;

**Search for files that were modified in the last 7 days below the current directory**

$ find ./ -mtime -7

**Search for files that have all permissions set in the current hierarchy**

$ find ./ -perm 777

### **Tar Command in Unix with Examples**

The archive format preserves the directory structure, and the file system attributes such as permissions and dates.

**Tar Syntax:**

tar [function] [options] [paths]

**Tar options:**

**The tar command supports the following functions:**

* tar -c: Create a new archive.
* tar -A: Append a tar file to another archive.
* tar -r: Append a file to an archive.
* tar -u: Update files in an archive if the one in the filesystem is newer.
* tar -d: Find the diff between an archive and the filesystem.
* tar -t: List the contents of an archive.
* tar -x: Extract the contents of an archive.

While specifying the function, the ‘-‘ prefix is not required, and the function can be followed by other single letter options.

**Some of the supported options include:**

* -j: Read or write archives using the bzip2 compression algorithm.
* -J: Read or write archives using the xz compression algorithm.
* -z: Read or write archives using the gzip compression algorithm.
* -a: Read or write archives using the compression algorithm determined by the archive file name.
* -v: Perform the operations verbosely.
* -f: Specify the file name for the archive.

**Examples:**

**Create an archive file containing file1 and file2**

$ tar cvf archive.tar file1 file2

**Create an archive file containing the directory tree below dir**

$ tar cvf archive.tar dir

**List the contents of archive.tar**

$ tar tvf archive.tar

**Extract the contents of archive.tar to the current directory**

$ tar xvf archive.tar

**Create an archive file containing the directory tree below dir and compress it using gzip**

$ tar czvf archive.tar.gz dir

**Extract the contents of the gzipped archive file**

$ tar xzvf archive.tar.gz

**Extract only the given folder from the archive file**

$ tar xvf archive.tar docs/work

**Extract all “.doc” files from the archive**

$ tar xvf archive.tar –-wildcards ‘\*.doc’