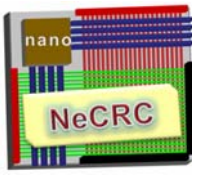
San Francisco State University
School of Engineering
Nano-electronics & Computing Research Center



BASIC UNIX/LINUX COMMANDS

Introduction

A short paper that briefly describes the UNIX shell, environment variables, and most commonly used commands.

UNIX Shells

A shell is a program and is a way to provide the user with an interface to the kernel (operating system). The shell interface could be "text based" or GUI (graphical user interface) based. Some examples of GUI based shells are GNOME and KDE and some examples of "text based" shells are:

- **Bourne shell (sh)**: Bourne Shell is used in the earlier version of UNIX. This is the original UNIX shell written by Steve Bourne of Bell labs. This shell does not have the interactive facilities provided by modern shells such as C shell and Korn Shell.
- **C shell (csh):** This shell was written at the University of California, Berkley. It provides a C like language with which one can write shell scripts- hence its name.
- **BASH shell (bash):** bash is an acronym Bourne Again shell. Bash Shell provides all the interactive features of the C shell (csh) and the Korn shell (ksh). Its programming language is compatible with the Bourne shell (sh).
- **KORN shell (ksh):** This shell was written by David Korn of Bell Labs. It is now provided as the standard shell on UNIX systems. It provides all the features of the C and TC shells with a shell programming language similar to that of the original Bourne shell. It is the most efficient shell.
- **Tenex C shell (tcsh):** is a more advance version of C shell. This shell is available in the public domain. It provides all the features of the C shell together with EMACS style editing of the command line.

The Shell is a command line interpreter and is mainly used for customization of the UNIX sessions and for writing shell scripts. Different shells have different features so in the end, the typical user can change to the shell that he/she is most comfortable with. The shell programs are located in the */bin* directory under the UNIX file structure. For example, the tcsh would be located at */bin/tcsh*. To change from one shell to another, simply type the shell name (ksh, tcsh, etc) into the command line.

Environment Variables

Basically, these are variables that you can edit to change the way your operating system operates. Different shells have different environment variables. These variables have reserved keywords and are case sensitive. Some of the more common variables are listed below:

- PATH is a list of directories, separated by colons (:), that is searched when trying to execute a program(s).
- DISPLAY contains the IP address for where the GUI application should be displayed to.
- MANPATH is the search path that is used when locating a man page, similar to PATH but for man pages.
- HOME is the path name of your home directory.
- TERM specifies the type of computer terminal being used.

So how do I change my environment variable? There are two ways. For temporary changes until you logout then you can edit variables via command line. For permanent changes, environment variables must be stored in the .cshrc file. The .cshrc file is a special file in your home directory (Example: /home/<your login name>) that can be used to customize your shell session (ksh, csh, tsch, sh, etc). This file also contains a list of current aliases.

In short, if you don't have a .cshrc file in your home directory, then you can create one using your favorite text editor (vi or xemacs). In tcsh shell, use the "*setenv*" command to edit an environment variable and "*echo*" command to view/display the value. To get a listing of all the environment variables in your current shell, use the "*env*" command.

Commands

This section contains the most commonly used UNIX commands in alphabetically order. Use the "*man*" command if you want more information on the command or to a detail list of the command options. Even though the commands are listed in capital letters, they are all **lower case** when used in the UNIX environment.

*****

Wildcard character used to match any number of characters in a filename.
Example:

% ls my*.txt

→ This command displays the list of all filenames in the current directory, which starts with "my" and ends with ".txt' extension.

<

The less than symbol is used to redirect the file as an input for a command.

Example:

% *command* < *file*

>

The greater then symbol is used to redirect the command output to a file.

Example:

% *command* > *file*

>>

Output can also be appended to the end of an existing file by using two greater-than symbols.

Example:

% *command* >> *file*

&

Append the ampersand at the end of your command to make it run in background so you can get back your command prompt.

Example:

% command &

|

With the pipe symbol (|) you can have the output of one command fed (piped) to the input of another command.

Example:

% *command1* | *command2*

ALIAS

The '*alias*' command creates a pseudonym or shorthand for a command or series of commands. Whenever the command names are encountered, they are replaced by the string. Alias are stored permanent in the .cshrc or saved temporally until logout via command line. If nothing is specified as the argument, all the current aliases are listed.

Example:

% alias pseudonym command

BG

See FG.

C-z

CTRL-z halts/suspends the current process running the foreground and returns the command prompt. The command does not the process. Use the "fg" command to restart a suspended process.

C-c

CTRL-c kills the current process in the foreground and returns to the command prompt.

C-a

CTRL-a goes to the beginning of a long command that is typed on the shell prompt.

CAT

The '*cat'* command is used to create, view, and concatenate files.
Example:

To creating a file:
% cat >file
Line 1
Line 2
<CTRL-D>
This creates a file with lines 1 and 2.

% cat >>file
Line 3
Line 4
<CTRL-D>
This adds lines 3 and 4 to the existing file.

To view file:
% cat file
Displays the contents of the file

To concatenation a file:
% cat file1 file2 > file3
Concatenates file1 and file2 into a new file; file3.

If no input file is given, cat reads from the standard input.

CD

The '*cd* 'command is used to change the directory. Its usage is as follows:
Example:

% cd <dir name>
This goes down one level into the specified directory.

% cd ..

This moves up one level.

% cd ../user

This command moves up one level and then goes down one level into directory user.

CHMOD

The 'chmod *'utility* changes or assigns the mode of a file or directory. The mode specifies the permissions and other attributes. The syntax is as follows:

Example:

% chmod [user type] [+ / -] [attributes] file name

User type can be u (user/owner), g (group), or o (others)
+ / - is used to add or remove the specified attributes
Attributes may be r (read), w (write), or x (executable)

Attribute specification can also be done in absolute mode using octal numbers of the form 'nnnn'; where n is an integer from 0 to 7.

% chmod 0040 <file>
This command allows the file to be read by the group.
%chmod 4755 <file>

This command sets the user ID, assigns read-write-execute permission by owner, and assigns read-execute permission by group and others

The permissions of files can be viewed using the ls –l command.

CP

The '*cp*' command copies files and directories. If the target is not present, it is created and then the copying is performed.

Example:

% cp source file target file
Copies the contents of source to target file.

% cp <list of source files> target dir
Creates new files with the same names as the source files inside the target directory, and copies the contents of source files into them.

% cp –r <list of source dir> target dir

Copies the contents of all the source directories (and their sub directories) to the target directory.

CMP

The '*cmp* 'command compares file1 with file2. The exit codes are as follows:
0: If files are identical

1: If Files are different
2: If Files are inaccessible
Example:

% cmp [options] file1 file2

DU

The '*du'* command prints disk usage. It will print out the number of 512 byte blocks used by each named directory and its subdirectories.
Example

% du report
55936 report.

DIFF

The '*diff*' command reports the lines that differ between two files (file1 and file2). Output consists of lines of context from each file, with file1 text flagged by a < symbol and file2 text, by a > symbol.
Example:

% diff [options] file1 file2

ECHO

The '*echo'* command simply echoes its argument.
Example:

% echo END
END

There are certain names which correspond to the so-called environment variables, such as DISPLAY, PRINTER, and etcetera. To echo the values of these variables, a dollar sign ($) must be appended in front. The return value is the default value unless it has been set to a new value.

% echo $DISPLAY
192.168.1.214:0.0

EGREP

The' *egrep* 'utility searches a file for a given pattern and prints out all the lines that contain the pattern. If no file name is specified, egrep assumes standard input. '*Egrep*' accepts most of the full regular expressions in its pattern. Its usage is as follows:
Example:

% egrep pattern file name

The output of egrep is usually piped to the less command to enable easy reading of the output.

% egrep pattern filename | less

EXIT

The '*exit'* command is used to exit from a shell.
Example:

% exit
Exits the shell.

FG/BG

The commands '*fg'* and '*bg '* are used to bring jobs into the foreground or push it to the background. They are usually used together with the jobs command. The usage is:
Example:

% bg [%job ID]
This pushes the specified job into the background.

% fg [%job ID]
This brings the named job into the foreground.

In the absence of arguments, bg and fg return the list of jobs running in the background and foreground respectively.

HOSTNAME

Hostname prints the name of the current host .This can be useful if the shell prompt is not configured to
display this accordingly.
Example:
% hostname
agape

HEAD

This command prints the first few lines of one or more files. Use –n to print the first n lines (default is 10). Also see *'tail '*.
Example:

% head [-n] [files]

HISTORY

The '*history* 'command displays all the commands previously typed in order, along with the time when they were invoked. The length of the list depends on the history variable.

Example:

% history

To execute commands that are recorded in the history session, we can do! *command number*.

Example

To execute the 55th command in the history session ! 55.

To execute the previous command in the history session do ! $.

IFCONFIG / IPCONFIG

The '*ipconfig'* command is used at the Windows command prompt to display information regarding the IP configuration of the computer. The IP address is shown along with the default gateway. The UNIX equivalent of '*ipconfig* 'is '*ifconfig'*.

Example:

>ipconfig
Windows IP Configuration

Ethernet adapter Local Area Connection

Connection-specific DNS suffix:
IP address : 192.168.1.214
Subnet Mask : 255.255.255.0
Default Gateway : 192.168.1.1

JOBS

The ' *jobs* 'command returns a list of jobs running in the background and foreground from a terminal or x-term. The name of the processes and jobs is returned along with their ID and status.

If the prompt is returned with no information, no jobs are present.

Example:

% jobs
[2] + Running xterm
→ 2 is the process ID.

The [-l] option is used along with the job command to list the process group ID.

% jobs –l
[2] + 12268 Running xterm
→ 12268 is the process group ID.

KILL

The '*kill'* command cancels/kills a job. It is used along with a process ID to indicate which process to cancel.
The '*kil'l* command is used as follows:
Example:

% kill % process ID
(Or)
% kill -9 % process ID

The kill with the -9 option is called hard kill, is faster and is the one that is usually used. Normally, jobs and kill are used together to first list out processes and then cancel particular ones.

LESS

To view/display a text file one page at a time. The less command is opposite to the '*more*' command. It allows both forward and backward movement within a file.
Example:

% less filename

The less command does not require that the entire input file be read. So for large files, it starts up faster than text editors like '*vi*'.

LN

The '*ln*' command creates hard or symbolic links to files. A hard link is pointer to the file and is indistinguishable from the original directory entry. A symbolic link is a new directory entry for the file.
Symbolic links are created using the [-fns] option. Hard links are created using the [-fs] option. Be careful when creating hard links because they can stall your machine when booting if link is not found. Use hard links with caution.
Example:

% ln [–fns] file <target path>
This creates a link for the file at the target path.

% ln [-fns] <list of files> <target dir>
This creates a link for each of the files within the directory specified by target. The specified directory must exist.

LOGOUT

The logout command terminates only a login session. It can be used to terminate a shell script.

MAN

The '*man*' command displays the reference manual pages for its argument. The online reference manual page directories are located conventionally in /usr/share/man. This command is usually used to obtain a quick overview of a command.

Example:

% man more

This displays complete information about the more command from the manual pages.

MKDIR

The '*mkdi*r' command is used to create a new empty directory.

Example:

% mkdir new

This creates a directory called new in the current directory.

MORE

To display, or view the content of a text file on the terminal, one page/screen at a time. It is normally used to read large files one page at a time.

Example:

% more filename

The output pauses after each page and waits for navigation instructions. Navigation in more is accomplished using the following commands:

q <= quits more
SPACE <= reads next page/screen
RETURN <= reads the next line
b <= goes back one page

MV

This is the basic command to move files and directories around the system or to rename them.

Example:

% mv [options] [sources] [target]

[Options]: -f: Force the move, even if target file exists

-i: Inquire; prompts for a (yes) y response before overwriting existing target.

PS

The '*ps*' command displays all active processes under a particular username. By default, it lists the process ID, terminal identifier, cumulative execution time and the name of the command.

Example:

% ps

Will display:

```
  PID   TTY      TIME CMD
13443 pts/76   0:00   tcsh
13811 pts/76   0:00   ps
```

PWD

The' *pwd* ' command is used to find out the current directory you are working on. It displays the entire path to the directory from root.

Example:

% pwd
/export/home/user

RLOGIN

The rlogin command establishes a remote login session from your terminal to the remote machine named hostname. The remote terminal type and the terminal or window size is the same as that of the local terminal.

Example:

% rlogin [-l username] hostname

By default, the local and rlogin usernames are the same.

The [-l username] can be used to specify a different username for the rlogin session than the local username.

RM

The' *rm* 'command is used to remove any files or directories permanently.

Example:

To delete files:

% rm -f <file names>

The -f is optional.

To delete a directory:

% rm –rf dir name

This removes the specified directory immediately including all files.

% rm –r dir name

If the directory is non-empty it asks to confirm removal of each file.

SET AUTOC

Command that will turn on the auto correct option where it will automatically correct misspell words.

Example:

%set autoc

SET AUTOLIST

Command used to automatically complete (or list) the rest of the file name or command by press the <TAB> key.

Example:

% set autolist

```
% unse <TAB>
unset    unsetenv
```

SORT

The '*sort*' command is used to sort the lines of a file. More than one file can be specified, in which case all the lines together are sorted. The sorted lines are then printed out. The default sort order depends on the value of LC_COLLATE. If this is set to C, sorting is in ASCII order.

Example:

```
% sort text.tcl
```

This sorts the file text.tcl in the default sort order.

The sort command can be used with the +n option , where we can skip n fields before sorting. This is applicable to files which are based on fields.

SSH

The ' *ssh* 'command, meaning Secure Shell, is a program for logging into a remote machine, and provides secure encrypted communication between two un-trusted hosts over an insecure network. There are different protocols for authentication.

Example:

```
% ssh [-l login name] [hostname | user @ hostname] [command]
```

Where the [-l login name] option is used to specify the login username.

SU

This command allows you to login as somebody else. You also need to provide the password.

Example:

```
% pwd
/home/user/hima
% su tony
password:
% pwd
/home/user/tony
```

TAR

Command used to compress or decompress files/directories and their sub directories.

Options:

```
-f = file
-v = verbose
-x = decompress/extract
-c = compress
```

Example:

To create/compress files/directories
% tar –cvf <destination.tar> <source.tar>

To extract a tar file to the current directory
% tar –xvf <file.tar>

TAIL

This command prints the last ten lines of the file given.
Example:
%tail [options] [file]

TOP

The' *top'* command displays all the processes running on that machine.
Example:
% top
The [-u] option can be used to display processes according to user name.
% top –u name.

TOUCH

The '*touch*' command updates the access time and the modification time to the current time and date. Touch is useful in forcing other commands to handle files a certain way. E.g. the operation of make relies on the files access and modification time.

VI

A text editor used to create, edit, and view any ASCII files.
Example:
% vi my_file.txt
Will create a file called my_file.txt or if they file already exists, then it will open that file.

WHICH

The '*which* 'command is used to locate a command and displays its full pathname. It takes a list of names and looks for files which would have been executed had these names been given as commands. The aliases and paths are taken from the user's .cshrc file.
Example:
% which ps
/bin/ps
This indicates the location of the file that is actually being executed upon typing ps.

WC

This command prints a character, word and line count for files. If no files are given, its reads standard input. The options are:
-c: Prints character count only

-l: Prints line count only
-w: Prints word count only

Example:

% wc [options] [files]

XEMACS

A text used to create, edit, and view any ASCII files.

Example:

% xemacs my_file.txt &

Will open up an xemacs editor in the background. The following are some of the basic Xemacs editor commands.

Ctrl-h: Online Help
Ctrl-x Ctrl-s: Save file
Ctrl-x Ctrl-c: Exit Xemacs
Ctrl-x u: Undo last edit
Ctrl-w: Delete selected part
Ctrl-v: Move forward by one screen
Ctrl-p: Go up by one line or character

XTERM

The xterm program is a terminal emulator for the X Window System. It provides DEC VT102 and Tektronix 4014 compatible terminals for programs that can't use the window system directly. The VT102 and Tektronix 4014 terminals each have their own window so that you can edit text in one and look at graphics in the other at the same time. The usage is as follows:

Example:

% xterm

This opens a new terminal.

xterm can also be used along with & to open a terminal in the background.

% xterm &