*****Strictly confidential only for Accenture use*****

Author: Latesh Sangani Contact: latesh.sangani@accenture.com

Unix Commands For Daily Activities

March 1

2015

This document is prepared for the Accenture Learning where most of the projects are using Unix for the code storage, scheduling jobs and some processing of the files. This document will help users to get understand syntax of the Unix commands and their usages.

Unix
Technical
V1.0

Objective: The purpose of this handout is to explain the various utilities provided in UNIX for programming.

Chapter 1: Introduction to UNIX.

1.1 Introduction

On its own, a computer system is just a collection of metal, silicon, communications equipment and magnetic media. It is the application programs which make a computer system useful to its users. Applications can range from word processing through writing compilers, to generating new operating systems.

To exploit the communications, data storage and information processing capabilities of the computer hardware, the applications software requires some form of overseer, which can handle the details of managing the hardware resources, accessing the files and interacting with the users. These supervisory functions are the job of the operating system, which in this case, is UNIX. It is a relatively small operating system but very useful meeting. It is composed of three major parts:

- **Kernel**: is that part of the system which manages the resources of whatever computer it lives on, to keep track of the disks, tapes, printers, terminals, communication lines and any other devices.

- **File system**: is the organizing structure for data. The file system provides the means of organizing the layout of the data storage in complex ways.

- **Shell**: is the command interpreter. The shell listens to the terminal and translates the requests into actions on the part of the kernel and many other utility programs.

1.2 Following are the features of Unix System:

1. Unix System is an **interactive** operating system. This means that you type commands, the

   System obeys the commands and displays appropriate responses.

2. Unix System is a **multi-tasking** operating system. This means that the system can perform several tasks – called processes – at the same time. More than one task can be executed in the background with some other task running simultaneously in the foreground.

3. Unix is a **multi-user** system. This means that more than one person can use the system at the same time. The system can attend to more than one user at a time. Groups of people can work together, share information and common utilities through the file system. It supports the concept of time-sharing i.e. the server devotes a specific amount of time to each terminal and keeps on switching between terminals after the time slice of the respective terminal is over. The entire process is so fast that each user gets a feeling as if the Server is devoting the entire time to him/herself.

1.3 Differences between DOS and UNIX

1. DOS is a single user operating system. UNIX is a multiuser operating system.

2. In DOS the root directory is **\\,** while in UNIX it is **/.**

3. DOS doesn't allow you to send messages to various terminals. UNIX provides utilities like write, mail to communicate with other users.

4. There is no security level in DOS. Just any user can come and work on a DOS machine. This may lead to tampering other user's data. In UNIX, every user works in his login. He can access the information which belongs to him. He can, however, access other users information depending upon the permission assigned to him.

5. DOS provides programming feature called as Batch Programming, whereas UNIX provides two features, namely: Shell Programming & awk Programming.

6. Manipulating contents of a file is tedious in DOS. UNIX offers various utilities like sort, grep, cut, etc. With the help of which it becomes very easy to manipulate file contents.

7. DOS commands can be classified into internal & external commands. In UNIX, all

   Commands are external.

1.4 User, Group, Other, Super user.

Everyone who uses UNIX is given a special user name to use when logging onto the system. A user name is also called an **'account'**, since in many installations someone is keeping an account of usage by different people and groups.

One person might have several user names. For example, if Sunil Mehta is working on payroll project, he may have an account called **'spay'**. If he is also handling inventory project, he may have an account called '**sacc'**.

Several people working on related things can be grouped together on the system, and the group can be allocated a group name or a group identity. For example: Ashok, Rahul and David belong to the same project. So they will have the same group name, let's say "**payroll**" but individual usernames. All users who do not belong to "payroll" project will be referred to as **others.** Such users will have a group name other than "payroll".

One of the users on any given UNIX system is called the **super-user**. This is the person who has the administrative duties of assigning new user names, and generally looking after the system. The super-user has the name **'root'** as the user name.

Every user has an account consisting of a ***login-id*** and a ***password***. The password is a string of characters which is typed to gain access to the system. The user is given independence to assign any suitable password to his login with/ without letting the super-user know about it.

1.5 Logging on

To gain access to UNIX, the user has to connect to the system. This process is called as *'logging in'* or *'logging on'*. Any UNIX terminal displays a message as follows:

```
Login: _
```

The exact details of the message will differ from installation to installation. To login to the system the user has to type the login name and press RETURN key. The system responds by asking the user to type the password. After typing the correct password and pressing RETURN key, the system will check if the user is a valid user of the system. If so, then the system executes some initial tasks and the display the **$** prompt to the user. In case the user specifies incorrect login name or invalid password then the system will display suitable error message.

The user can then type any valid UNIX command at the $ prompt. The system interprets the command typed in by the user at the $ prompt, executes it, displays the results and then waits for more commands.

1.6 Logging off

Once the user finishes working on the system, he has to exit from UNIX. This process is called as *''**loggingoff**''*. A user can log off the system by pressing **Ctrl-D(^D)** at the $ prompt after which the system will display the login screen as shown above.

1.7 Unix Utility Software

UNIX comes equipped with a large number of utility software to help users get started with useful applications right away. A large proportion of utility software is devoted to manipulating text files. There are text manipulation tools such as **grep** for selecting lines matching a given criteria, tools like **sed**, **awk** for changing contents of a file, **sort** for rearranging the order of lines in a file.

Chapter 2: UNIX Most Commonly Used Commands.

1) **man**– Online UNIX manual (“man-pages”). It gives detailed instructions for all of the commands accepted by all the UNIX flavors.

2) **catfilename1 > filename2**– Overwrite contents of filename2 with filename1.

3) **catfilename1 >> filename2**– Append contents of filename1 to filename2.

4) **cd/usr**– Change current directory to /usr.

5) **cd** .. – Change current directory to previous higher directory.

6) **cd**– Change to home of current userid.

7) **chgrp group1 filename1**– Change group id to group 1for filename1.

8) **chgrp -R group1 ***– Change group id of all files in current and subdirectories to group 1.

9) **chmod ugo+rwx filename1**– Add read/write/execute permission to filename1 for user/owner, group andothers (world)

10) **chmod o x filename1**– Remove execute from others (world)

11) **chmod 751 filename1**– Set rwx for user/owner, group and others; r=4, w=2, x=1, rwx=7 rx=5, rw=6, wx=3,none=0

12) **chown sam filename1**– Change owner of file filename1 to sam.

13) **chown -R sam ***– Change owner to sam for all files in current and subdirectories.

14) **clear**– Clear the terminal screen.

15) **compress filename1**– Compress file filename1 and replace it with filename1.Z.The opposite action of the compress command is **uncompress.**

16) **cp filename1 filename2**– Copy file filename1 to filename2 destroying the contents of filename2.

17) **cp -i * /usr/local/bin**– Copy all files in current directory to /usr/local/bin directory. –i option will prompt before overwriting files.

18) **cpio** – Copy file archives to disk/tape. See more information on manual pages.

19) **crontab -e**– Edit crontab file for current user.

20) **crontab -e sam**– Edit crontab file for user sam

21) **crontab -l**– List crontab entries for current userid

22) **csh**– Start the c shell process. See manualpages for more information.

23) **cut -d: -f1,5 /etc/passwd**– Extract the 1st and 5th fields (username & real name) from file /etc/passwd where delimiter is colon

24) **who | cut -d" " -f1**– List login names from who command

25) **date**– Display current date string

26) **date +%D**– Display current date, MM/DD/YY format

27) **dd**– Copy file(s) to/from raw devices. See manualpages for more information.

28) **df** – Display free disk blocks and modes on file systems.

29) **df -k**– Display free space in kilobytes for mounted file systems

30) **echo name**– Displays literal "name" on screen

31) **echo ${PATH}**– Displays PATH environment variable

32) **ed/edit/ex**– Alternative line editors, see vi

33) **env**– Displays current environment variables or allows setting

34) **file filename1**– Determines and displays type of file for filename1 (text, data, executable, directory,symbolic link...etc...)

35) **find /usr/opt/bin -name "filename1*" -print**– Starts searching in /usr/opt/bin for files starting withfilename1. If found prints the full file names and continues searching subdirectories

36) **find . -type f -print | xargs grep -i [PATTERN]**– Recursive grep for a pattern in a file (searches throughfiles in subdirectories).

37) **find . ! -mtime -<days> | /usr/bin/xargs rm -rf** – Finds and removes files older than <days> specified

38) **finger sam**– Displays data about user session for sam.

39) **ftp solar**– Establishes a File Transfer Protocol sessionover the network between current host and a hostnamed solar. See manualpages for additional ftpcommands

40) **grep jdoe /etc/passwd**– Searches the file /etc/passwdsearching for string "jdoe"; If found, displays on terminal.

41) **grep -i Sam filename1**– Search filename1 for upperor lower case string of Sam and display lines found. See manual for details information and attributes.

42) **groups sam**– List groups that sam is a member of.

43) **head filename1**– By default display first ten lines of filename1. See tail command also.

44) **head -50 filename1**– Display first fifty lines offilename1.

45) **id**– List current user id and any group ids

46) **kill -9 1351**– Terminate process number 1351

47) **ksh**– Start Korn Shell command interpreter; See manualpages for more information. Preferred shell for most of the users.

48) **ln -s filename1 /usr/opt/filename2**– Create asymbolic link named /usr/opt/filename2 that points tofilename1. See manualpages for more information.

49) **lp -d lp1 filename1**– Print filename1 on destinationprinter lp1.

50) **lpstat -d**– Displays name of default printer, if anyone is present.

51) **lpstat -a**– Lists printers accepting print requests.

52) **lpstat -s**– Displays most everything regarding printing.

53) **lpstat -u sam**– Displays status of sam's print jobs.

54) **ls -al**– Displays all files in wide listing.

55) **ls -al *.doc**– Displays files ending with .doc

56) **ls -al /bin/k***– Displays files starting with k in /bin directory.

57) **mailsam**– Starts mail message to sam.

58) **mailx sam**– Nicer looking mail utility with some advanced options.

59) **mkdir -p /usr/opt/dirx**– Creates directory dirx below /usr/opt.

60) **make**– Code compilation utility. It is used when any linux software is installed.

61) **mkdir dirx**– Creates directory dirx

62) **more filename1**– Displays single pages from filename1 pausing after each page. See man pages for many options.

63) **mv filename1 /usr/opt/**– Moves filename1 to directory /usr/opt.mv removes file from origin while cp just copy the contents of the file.

64) **netstat -i**– Show the TCP/IP network interfaces.

65) **netstat -r**– Show network route table.

66) **netstat -rn**– Displays routing information but bypasses hostname lookup.

67) **netstat -a | more**– Show the state of all sockets.

68) **newgrp group1**– Changes current group to group 1.

69) **news**– Displays unread files from /usr/news or /var/news.

70) **nice/renice**– Adjusts process' execution priority.

71) **passwd**– Allows changing your login password.

72) **ps** – Lists all of current user's live processes.

73) **ps -ef**– List all users processes that are executing.

74) **pwd**– Displays current working directory where user is present.

75) **rcp filename1 hostb:/usr/local/bin**– Copies filename1 from current host system to hostb, and places it in the /usr/local/bin directory. The .rhosts or hosts.equiv files must be setup to allow action. See scp commads it is same as rcp.

76) **rksh**– Starts restricted Korn Shell session.

77) **rlogin hostb**– Logs into remote host name hostb.

78) **rm filename1**– Deletes filename1 from the disk without recourse.

79) **rm -i filename1**– Deletes filename1 after promptingfor verification.

80) **rmdir dirx**– Deletes directory dirx.

81) **rmdir -r dirx**– Deletes directory and all contents.

82) **rsh**– Restricted version of Bourne Shell for security.

83) **sar**– System Activity Reporter.

84) **sh**– Bourne Shell command interpreter. Alternative toKorn Shell and C shell but it is the default on mostsystems. It is the older version that is losing popularity.

85) **sleep 3**– Pauses for 3 seconds and continues.

86) **stty sane**– Attempts to restore terminal settings afterthey are hosed; Use CTRL-J with this commandstty erase ^H – Use to reset backspace/delete; ^H isthe key you may want to use for backspace/delete. Typically the backspace key itself.

87) **su sam**– Log on as user sam.

88) **su - sam**– Log on as sam, and execute his profile too.

89) **tail filename1**– By default display last ten lines of filename1.

90) **tail -50 filename1**– Display last fifty lines of filename1.

91) **tail -f**– Continually reads updating file. Great for monitoring a log file while being written by the jobs.

92) **tar -cvf /dev/rmt/0c /usr/bin/***– Tape archive utility. Copies all files in /usr/bin directory to tape device/dev/rmt/

93) **tar -xvf /dev/rmt/0 /usr/bin/**– Extracts all files from tape device /dev/rmt/0 and writes them to /usr/bin/

94) **tar -tvf /dev/rmt/0c**– Read tape on device /dev/rmt/0c and lists contents in verbose mode.

95) **tee**– Used in scripts to split output to two outputs. Usually used with a pipe command (|)

96) **tee -a /tmp/filename1**– Append the output to filename1 without overwriting its original contents.

97) **telnet hostb**– Create a remote terminal on hostb.

98) **touch filename1**– Creates an empty file named filename1. Changes modification time to current timeif the file already exists.

99) **uname -a**– Lists 0/S revision, host name, hardware

100) **uncompress filename1**– Uncompresses file with .Z suffix, created by compress command previously.

101) **uptime**– Displays current time, time logged-in, number of users etc.

102) **users**– Displays current logged-in users in a listing

103) **uucp**– UNIX-to-UNIX copy utility. See man pages for more information.

104) **view filename1**– A read only version of vi editor.

105) **w**– Combination of uptime, who and ps -a commands

106) **wall – Write to all** Allows entry of message to be sent to line 25 of all terminals; End message with a CTRLD command.

107) **wc -l filename1**– Utility that counts the number of lines in filename1

108) **wc -c filename1**– Utility that counts the number of bytes in filename1.-m provides number of characters.

109) **wc -w filename1**– Utility that counts the number of the words in filename1

110) **whence filename2**– Prints path name location of executable filename2

111) **which filename2**– It is similar to whence command.

112) **who**– Displays login name, terminal name, date, and time of login, of users currently logged in.

113) **who am i**– Displays effective user id of user

114) **who -b**– Displays date/time of last reboot

115) **who -r**– Displays current system run level

Chapter 3: Unix Visual Editor Commands and Navigation.

3.1 Cursor Movement

1) **h,j,k,l** Move left, down, up and right

2) **O,$** Jump to front, end of line

3) **w** Skip to next word

4) **CTRL-D** Down one page

5) **CTRL-U** Up one page

6) **G** Skip to end of file

7) **1G** Skip to top of file

8) **CTRL-R or CTRL-L** Refresh screen.

3.2 Searching

1) **/** text Search forward for text.

2) **?** text Search backward for text.

3) **n** Repeat search after find.

3.3 Line Numbering

1) **CTRL-G** Display current line number.

2) **:99** Move to line number 99.

3.4 Inserting Text

1) **a** Append after cursor

2) **A** Append to end of line

3) **i** Insert before cursor

4) **I** Insert at beginning of line

5) **o**Open new line below cursor

6) **ESC** Terminate edit mode return tocommand mode

3.5 Changing Text

1) **cw** Change word

2) **cc** Change whole line

3) **C** Change text to end of line

4) **dd** Delete line

5) **5dd** Delete current line and next 4

6) **D** Delete to end of line

7) **u** Undo last change

8) **U** Restore current line

3.6 Moving Text

1) **yy** Yank a copy of current line

2) **p** Insert the previously yanked line

3.7 Saving and Exiting

1) **ZZ** Save and exit file

2) **:wq** Same as ZZ

3) **:q!** Exit without saving

4) **:n** Go to next file to be edited

5) **:w** filename1 Save edited file as filename1

3.8 Symbols

1) >Redirect output

2) < Redirect input

3) >> Append to file

4) | Pipe Output

5) **&** Run process in background

6) **;** Separate commands

7) *****Match any character(s)

8) **?** Match a character