Common Linux (Unix) Commands and Concepts

1. **Contrast "stdin," "stdout," and "stderr."**

   All Linux (Unix) programs inherit "genes" from their parents in the form of the controlling terminal, current working directory, and currently opened files, etc. At least three file descriptors are always inherited, standard in (usually the keyboard), standard out (usually the display), and standard error (also the display). This fundamental construct works two ways. First, it allows programs to be easily written without regard to the exact nature of the peripherals with which the program communicates. Second, this construct allows the parent to "switch" different peripherals into standard in and standard out before executing a child program.

   Standard out and standard error generally connect to the same device (the display) but many times the parent will switch the child's normal output to go to one device, while potential error messages are monitored on a separate device.

2. **Explain the significance of the $ *versus* the # prompt symbol and describe how to switch between them.**

   The $ symbol represents non-privileged (positive valued) UIDs whereas # symbol represents superuser status (zero) UID that overrides any file system protection scheme. The superuser is also the only account that may request certain system services, such as changing date/time and increasing a processes priority.

   Administrators can quickly switch from user to superuser status with variants on the `su` command.

   - `su` Change your effective UID to 0 to allow access to files or services.
   - `su - useraccount` Switch real and effective UIDs to the user account (similar to logging in as that user).
   - `su - root` Similar to logging in as root.

   To return to the administrator account, simply logout of the superuser or other user shell.

3. **Explain how to set a password, describe three qualities of a good password, and explain the two ways the superuser can use the password program differently from a regular user.**
   I. Type the `passwd` command and enter the new password. Since the characters are not echoed, the password must be typed twice.
   II.
      a. Use combination of numeric, upper, and lower case characters.
      b. Make the password at least six characters long.
      c. Concatenate abbreviations of familiar concepts, people, and objects to form nonsense

words (e.g., 2Cats1dog, Ponti86, GeksRul2001).

III. The superuser can change the password for any account with the command passwd accountname and the superuser can force the password program to accept any password regardless of weakness.

4. Describe the role of each of these related commands: ls -la, pwd, cd, mkdir, and rmdir.

These commands control access to file and directory names. File names beginning with a "." are not displayed with the usual ls command. But ls -la show the file details plus all the administrative files that begin with "." By definition a user is located in a current working directory and the print working directory (pwd) command reveals which one. Change directory (cd) moves up and down the logical file system hierarchy while make directory (mkdir) and remove directory (rmdir) manipulate nodes in the file system hierarchy.

5. Given the display of the ls -l program output below, describe six functions of the first column, and the role of each of the remaining six columns.

6. drwxr-xr-x 2 ptm html 1024 Aug 26 10:23 images
7. -rw-r--r-- 1 ptm html 11095 Aug 27 08:27 index.0
8. -rw-rw-r-- 1 ptm html 11104 Aug 27 20:46 index.html
9. -rwsrwxr-t 1 ptm html 184 Aug 20 20:46 script

10. Describe at least eight programs that will display the contents of a file in Linux.

text editors: pico, vi, and emacs more, less, cat, od, and dd

11. In terms of inodes, contrast the concepts of directory, file, hard link, and symbolic link.

An inode lists the block numbers that make up a file. Inodes are located in one or more fixed positions within the volume (hard disk partition). A directory contains pairs of file names and inode numbers which "point" to the file data. A hard link is the same inode number associated with two or more file names in two or more directory files. Finally, a symbolic link is a file that contains the path (file name) of another file. Since hard links employ volume relative block numbers, they are restricted to the current volume, but symbolic links may "point" anywhere among the mounted file volumes.

12. Define "shell", describe two ways to differentiate the six Unix shells, and list the most used shells.

A Unix shell is the command line interpreter that provide simple command line editing, executes other programs, and offers flow control in the execution of other programs (batch jobs).

- Interactivity - the newer bash, tcsh, and zsh shells offer command line completion and file name prompting.
- Script execution - sh, bash, ksh, and zsh employ different flow control constructs than the Berkeley derived csh and tcsh.

Thus, most users employ the bash, ksh, or zsh.

13. Contrast the erase, kill, intr(rupt), and susp(end) single character shell commands. Explain the problem with the "backspace" key.

The stty command associates a given key stroke with each type of command.

- erase - Generates backspace, space, & backspace (^H).

- kill - Deletes the line and start over (^U).
- intr - Interrupts or kills the program (^C).
- susp - Suspends the program and begins a new shell (^Z). Also, the internal shell command "fg" (foreground) kills the shell and returns to the program.
- quit - Second type of program kill command (^\).

The problem with backspace is that there are two possible definitions, ^H and DEL, and some programs assume only one definition. Thus, as you run different programs you find yourself switching among the "backspace," "delete," and "^H" keys to do simple command line editing.

14. **Describe bash shell commands for command line review, editing, ^T editing, and file name completion.**
    - Review - previous commands can be viewed with the up and down arrow keys. In this way, a series of commands can be repeated by pressing the up arrow until the command is found and then pressing the Enter key.
    - Editing - the display cursor may be moved to the left or right with the same arrow keys thereby positioning the cursor to the right side of where you wish to delete or add characters.
    - ^T - the ^T command saves three keystrokes by exchanging the character to the left of the cursor with the character under the cursor. For example, if the cursor were on top of the "o" in mroe, then ^T would create "more" with the cursor moving over to the "e." Typing ^T more than once will "drag" the character (left of the cursor) to the right side of the command line.
    - File name completion - the bash shell will attempt to finish a partially completed command name or file name when the Tab key is pushed. If progress cannot be made and the tab key is tapped twice, then file names from the specified path directory are displayed.
15. **Describe the After Step desktop equivalents to MS-Windows "cool key" and cut/paste shortcuts.**
16. **Describe the following wild card conventions: "?," "[ ]," "[-]," and "*." Explain how they relate to the "noclobber" shell variable.**
17. **Explain the difference among the following shell redirect operators "<," ">," ">>," "&>," "2>file1 >/dev/null."**
18. **Describe the following shell construction: "zcat archive.tar.gz | tar -tvf - | more."**

    The pipe operator (|) is used to concatenate three independent operations into one action. The `zcat` command will read a GNU compressed file and un-compress it to standard out. The output is a tape archive (tar) formatted file that is fed into the `tar` program with the first pipe. The `tar` program views the archive table of contents (`-tv`) and reads the archive from standard in (`f -`). The output is then fed into the `more` program with the second pipe so that the user may view the archived files one screen full at a time.

19. **Contrast the roles of "du" *versus* "df."**

    Disk usage, `du`, lists all files and sub-directories from the specified or implied (current working directory) and their size in **logical OS blocks**. Disk free, `df`, shows where the various volumes are mounted in the directory structure and the amount of free space in **bytes**. `du` is used to see how much space a given user or application occupies, where `df` is used to see how disk space in general is being allocated.

20. **What is a Linux command and how is it found?**

    There are two types of Linux (Unix) commands, internal to the shell and external to the shell. Internal commands are specific to the type of shell (bash and ksh *versus* tcsh and csh) and only affect the present shell. For example, `cd` is an internal command that leads to problems when a shell script

terminates. The user does not usually realize that upon script completion and return to the parent shell, the current directory returns to the parent's value since the parent never executed the internal command.

External Linux commands consist of files that are located, loaded, and executed by the shell. The shell searches common directories that are setup by the system administrator in the /etc/profile, /etc/csh.cshrc, .profile, or .cshrc files. Common directories include: /usr/local/bin, /usr/bin, /bin, /usr/bin/X11, and /usr/games but not "." or the current directory. Thus, to execute a command or a program in the current directory, one must type "./command."

To provide for communication among successive generations of shells, Linux (Unix) pass "environmental variables" from parent to child. These variables are ASCII strings that may be seen with the set or echo $NAME internal commands. For example, the command echo $PATH will display the shell's search path for external commands.

21. Explain how to use the copy/paste operations to quickly add the current directory "." to your shell search path. Explain how to automate the operation.

For the bash or ksh:

0. Display the search path by typing the internal shell command "set."
1. Using the mouse cursor, highlight the path list.
2. Type the first part of the internal command "export PATH="
3. Click the middle mouse button (or simultaneously hold and release both buttons of a "two button" mouse) and the previously highlighted path will be added to the command line.
4. Use the arrow keys to position the cursor anywhere in the path and enter the additional directory to search plus the ":" delimiter.

To add the new directory to the end of the path, use the command export PATH=$PATH:.

For the csh or tcsh, the internal command is set path = ($PATH .)

To automate the process, edit the .bashrc or .cshrc files with one of the above commands.

22. Describe eight ways to discover more information concerning an external Linux command.

0. Ask the command itself with the arguments "--help," "-h," or "-?."
1. Ask the GNU "info" program for more details on the command with "info command."
2. Check the old manual pages with "man command."
3. Check for cross references with "whatis command," "man -k command," or "apropos command."
4. Check the directories "/usr/doc/command" and "/usr/lib/command"
5. Use the locate command "locate '*command*'"
6. Use the find command "find / -name '*command*'"
7. Use the Debian package manager "dpkg --listfiles command," or "dpkg --search command."

23. Contrast the roles of "locate log" *versus* "find / -name '*log*' -print."

The locate program is a fast and easy to use GNU utility (*i.e.*, locate log). It is fast because it searches a database of file names for the specified file instead of having to ask the file manager to sequentially step through all the file system inodes looking for the file name. Locate also helps with Network File System (NFS) volumes since it avoids the network as well as the remote file manager. However, the

results of locate may be incomplete since one usually wants to locate odd files that have been added recently and are probably not in the database.

The find program is a slow but complete utility in that it searches out every branch of the directory hierarchy. The find utility was designed to be flexible and, as a result, requires arcane arguments. For example, find / -name "*log*" -print tells the utility to begin searching at the root directory for any file or directory name consisting of "log" or having the sub-string "log" embedded within the name. The quotation marks are required to pass the "*" characters onto the find program without the shell interpreting them as wild card operators.

24. Describe three levels-of-multi-tasking that one user may control from the console terminal. Explain the difference between "fg %1" and "fg 1234."

. Use the Fx keys to switch among virtual terminals.
I. Use the "&" internal command to tell the parent shell to return to the console for more commands.
II. Use the X-window desktop to launch a new "X-term" window.

The foreground, fg, command brings terminal control back to a background process. The %1 matches the job number assigned to the process when it was placed in the background with the "&" command. The 1234 argument specifies the process identifier (PID) returned by the process status, ps, command.

25. Contrast the Linux (Unix) strongly typed files.

0. Regular - A file containing a stream of ASCII bytes.
1. Directory - A file containing pairs of inode numbers and file names.
2. Symbolic - A file containing the path to another file.
3. Device - A file containing Logical Unit Table (LUT) offsets that point to a device driver or an offset to a system abstraction such as "/dev/null," or "/proc."
4. Socket - A file that holds the state of a local or remote protocol-based conversation.

26. Describe the role of "permission" and contrast file versus directory permissions.

Permission is the idea that a file's contents can be read or written, and if the file is executable, it's the idea that the file can be run. The same permission bits shift in meaning for directory files. The "executable" permission determines if one can enter or change their working directory to the directory or allow access to a file in the directory. Directory read permission determines if the file name may be viewed. Write permission indicates that a file maybe created or deleted. Thus, it is possible to have writable access to a file but not be able to see its name.

27. Contrast the SUID and SGID permission bits for regular files versus directory files.

Set User Identifier (SUID) and Set Group Identifier (SGID) apply to executable files. They cause the user's effective ID to switch to the file's ID upon execution. In this way, system programs may temporally promote a user to superuser so that it may access system resources. Generally, the SGID is not used by system programs.

Again, the meaning shifts for directories. The SUID bit is unused; while the SGID bit will allow another user to create a file in the directory, the GID will have to be the GID of the directory, not the GID of the user. For example, if several users belonged to the group "games" and the directory was "SGID games" then, regardless of the users' current group IDs, the new files will belong to the group "games." This allows several users to create separate files and also share read/write access with other

group members.

28. **Describe the interaction among permission and user, group, and other.**

    Permissions apply at three levels, the file owner (the user), the file group (the group), and anyone else (other). A user must first have permission to "enter" the directory either as the owner, a group member, or as anyone. Once in the directory, the user must have permission to access an individual file as the owner, group member, or anybody.

29. **Describe an ACL and its design tradeoff.**

    An Access Control List (ACL) is a sparse matrix of files (as columns) and users (as rows). If a user has access to a file then the intersection is "checked." There is a matrix for each file attribute such as read, write, execute. ACLs provide find-grained control over file system access determining exactly which accounts may access which files. But, the sparse matrix(es) take up space and require a lot of update time.

30. **Describe how to approximate an ACL with Linux groups and the SGID permission bit.**

    Any number of groups may be created in Linux. Files with restricted access are setup with a common group and user accounts are assigned to the same group (as well as other groups). To access a given restricted file, the user issues a newgrp command to switch to the desired group. A problem arises when the user wants to move files between two restricted file groups. In this case, files would have to be copied from one restricted area to another area accessible by the user. Then the user would newgrp othergroup to copy the files into the other restricted directory that has its SGID bit set.

31. **Contrast "chown," "chgrp," and "chmod."**
32. **Give the tradeoff of using symbolic versus numeric arguments for the chmod command.**
33. **Explain the role of umask and its insane arguments.**
34. **Explain the role "rc" files, their naming conventions, and give six types of applications that use them.**

    Shells, editors, mail, news, X-Window, and desktop GUI

35. **Give the two dictionary definitions of "daemon" and explain the role of /etc/inittab and /etc/inetd.conf.**
    - inittab - init, kflush, kswap, syslogd, crond, named, sendmail, httpd, and inetd
    - inetd - in.telnetd, in.fingerd, in.timed, in.ftpd, in.pop3d, in.lpd, in.rshd, in.rwhod, etc.
36. **Define a Linux (Unix) "process." Contrast fork and execute, and describe the series of events that lead from system boot up to your login prompt.**
    - ASCII tty - init -> read inittab -> fork getty per terminal -> exec login -> exec shell
    - GUI - startx -> xinit -> Xwrapper -> afterstep ->
        - Desktop - Winlist, Animate, Wharf, Pager, asclock, asfsm, asload, asmail
        - Applications - xman, nexterm, netscape, bash
37. **Describe the role of "ps eax" and "kill -9" utilities. Explain the reason for the "eax" and "9" arguments.**
38. **Describe the command "rdev /boot/vmlinz" versus "rdev /boot/vmlinz /dev/hda1."**
39. **Describe the command "dd if=/boot/vmlinz of=/dev/fd0 bs=8192" and explain how the "dd" utility differs from the "cp" utility.**

40. Explain the concepts behind "LILO."

Setup special kernel arguments such as `append=128M` or `append="ether=11,0x280,eth0 ether=5,0x300,eth1"` MBR, description of boot disk, an OS stanza, kernel arguments, extra kernel stanza, the MBR write operation, and the Tab key.

41. Explain why the Linux boot partition may or may not be set "active."

This depends on which boot program is loading Linux. If LILO is booting, then it ignores the active partition. If DOS or OS/2 is booting, then they need to see an active partition.

42. Describe the command "dd if=/boot/boot.300 of=/dev/hda bs=446 count=1"

43. If given a list of kernel boot up messages, be able to comment on each line.

44. Define runlevels 0,1,3,5, & 6. Explain how init switches among runlevels. And explain the difference between the init actions "wait" and "respawn."

Runlevels used by the Red Hat distribution are:

- 0 - halt (Do NOT set initdefault to this)
- 1 - Single user mode
- 2 - Multiuser, without NFS (The same as 3, if you do not have networking)
- 3 - Full multiuser mode
- 4 - unused
- 5 - X11
- 6 - reboot (Do NOT set initdefault to this)

The init process is always waiting for events. The Ctrl-Alt-Del key sequence sends a signal via the keyboard driver. Special devices such as the UPS can switch runlevels sending the power fail signal. Or, running the init program with a runlevel argument (init 5) will also change runlevels. Wait means wait for another event before re-executing a runlevel. Respawn means wait for the process to terminate and then restart it.

45. Describe the role, contents, and interaction of passwd and group files.

- Role of password and group files - Associate an account name with file and system service access permissions. Allow one account to selectively share access through group membership.
- Password - AccountName:EncodedPassword:UID:GID:Comment:HomeDir:Shell
- Interaction - Account names and passwords greater than eight characters have unpredictable results across system programs and types of systems. No password entry means no password required when asked for a password. The comment field may have up to four subfields: full-name, office, office-phone, and home-phone that are maintained with the "change finger," `chfn`, command. An account may belong to any number of groups and switch among groups with the `newgrp groupname` command. Past and present group membership plus all allowable groups can be seen with the `id` command.

46. Describe the six aspects of adding a new user account to a Linux system.

Use "adduser" or "vipw" (must be superuser) to:

- Add the account name, UID, GID, user name (address), home directory, and login shell to the `/etc/passwd` file.

- Optionally edit the /etc/shadow password file.
- Create the home directory for the user making the account name the owner of the directory.
- Use the "passwd" program to create a password for the new account.
- Use the template file "/etc/skel" to copy initialization files for common applications to the new user's home directory.
- Create (touch) the incoming mail file "/var/spool/mail/newacountname" and make the account name the owner. Change group owner to "mail." And set "rw" for owner and group, and "---" for everyone else.

The adduser program can do one or more of the first five steps automatically.

47. Comment the following shell (sh) script that simplifies adduser by only requiring the user account name.

```
48. echo -n "Enter Account name: "
49. read name
50. while [ $name != "done" ]
51. do
52.       adduser -g users -s /bin/bash -d /home/$name -p $name $name
53.       echo -n "Enter Account name: "
54.       read name
55. done
```

- The echo command displays a prompt with new line suppressed.
- The read accepts an alpha-numeric string into the internal shell variable "name."
- The while command evaluates the string compare expression and continues the commands in the "do-to-done" block until the contents of the internal variable "name" equal "done."
- The adduser command installs a user account always putting the user in the group "users," giving the user a bash shell, creating a user account with the user's account name, and creating an initial password with the user's account name.
- The echo and the read commands are used again, but now they are processing a list of names

If the name of this script was "newaccount" then a batch of new accounts could be processed with the command ./newaccount < users.

56. Explain the mechanism of how a new user could create files but the files are owned by a present, old, or nonexistent user.

57. Explain how to disable or modify a user's account. What must the administrator do if the user's UID is modified?