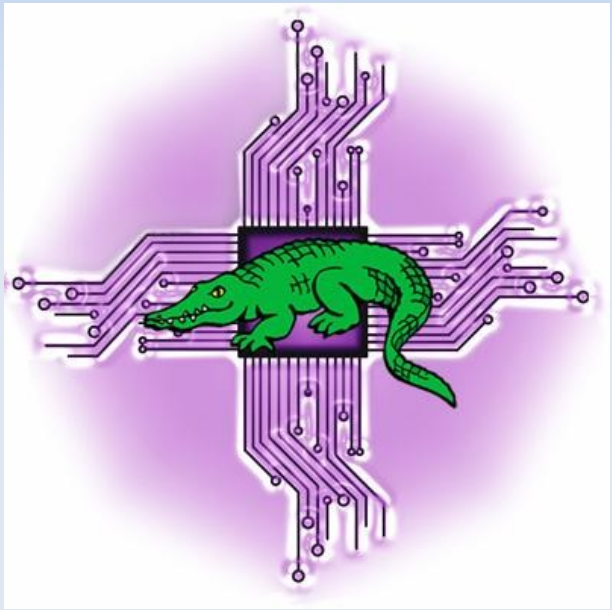
The Linux Command Line

Presentation to Linux Users of Victoria Beginners Workshop



August 18, 2012

http://levlafayette.com

What Is The Command Line?

1.1 A text-based user interface that provides an environment to access the shell, which interfaces with the kernel, which is the lowest abstraction layer to system resources (e.g., processors, i/o). Examples would include CP/M, MS-DOS, various UNIX command line interfaces.

1.2 Linux is the kernel; GNU is a typical suite of commands, utilities, and applications. The Linux kernel may be accessed by many different shells e.g., the original UNIX shell (sh), the TENEX C shell (tcsh), Korn shell (ksh), and explored in this presentation, the Bourne-Again Shell (bash).

1.3 The command line interface can be contrasted with the graphic user interface (GUI). A GUI interface typically consists of window, icon, menu, pointing-device (WIMP) suite, which is popular among casual users. Examples include MS-Windows, or the X-Window system.

1.4 A critical difference worth noting is that in UNIX-derived systems (such as Linux and Mac OS), the GUI interface is an application launched from the command-line interface, whereas with operating systems like contemporary versions of MS-Windows, the GUI is core and the command prompt is a native MS-Windows application.

Why Use The Command Line?

2.1 The command line uses significantly less resources to carry out the same task; it requires less processor power, less memory, less hard-disk etc. Thus, it is preferred on systems where performance is considered critical e.g., supercomputers and embedded systems.

2.2 The command line allows for complex commands with comparative ease and speed, especially with the use of wildcards.

Examples: Resizing multiple image files to 50% of their original (convert -scale 50% *.jpg), Renaming multiple files from one suffix to another (rename .html .php *.html). Exporting multiple files from a ODF format to PDF (soffice --headless --convert-to pdf *.odf)

2.3 The command-line interface allows for the automation of more complex tasks via scripting: (for file in *.mp3; do ffmpeg -i "${file}" "${file/%mp3/ogg}"; done)

2.4 The command line is a natural place for editing of text-based configuration files. Also the command line has a long history of a consistent interface.

Environment Exploration

3.1 Start with opening a terminal. Note the prompt ($) which distinguishes between the normal user and the superuser (#). Find out who you are (whoami) and were you are (pwd), by printing the working directory. Find out what files are in the current directory (ls). Note the current directort (.) and the parent directory (..). Change your context to the root directory (cd /), run a directory listing (ls), run a directory listing of the home directory using tab completion (ls /home/username), and then return to the home directory using the tilde shortcut (cd ~)

3.2 The directory listing in the root directory gives an overview of the Linux filesystem, which will typically consist of: /bin (common programs), /boot (startup files and the kernel), /dev (references to peripherals), /etc (system configuration files), /home (user directories), /lib (library files, /mnt (mount point for external file systems), /opt (extra and third party software), /proc (information about system resources), /root (administrative user's home directory), /sbin (programs available to root), /tmp (temporary system space), /usr (user programs, libraries, documentation etc.), /var (storage for all variable files and temporary files)

Environment Exploration cont.

3.3 Linux commands often come with options expressed as: <command> -<option[s]>. List all files in a directory with long format (ls -lart). The options are file permissions (l), include hidden files ('a', for all), sorted by reverse order ('r'), by modification time ('t'). Options can always be found, in terse form, from man <command> or info <command>.

3.4 The command line allows for tab completion of paths, and a cursor movement through history.

3.5 Some of the hidden files may include .bash_history (recent commands), .bash_profile (startup on login), .bashrc (startup for non-login shells). When Bash is invoked it first reads and executes commands from the file /etc/profile, if that file exists, then ~/.bash_profile, ~/.bash_login, and ~/.profile. Typically an startup file for an interactive shell will include startup commands, paths and environment variables, aliases and terminal colours. When a login shell exits, Bash reads and executes commands from the file ~/.bash_logout, if it exists.

Redirection

4.1 Linux also have very useful 'pipes' and redirect commands. To pipe one command through another use the '|' symbol. To show who who is logged on, how long they've been online and pipe through the less command to show one screen at a time (who -u | less).

4.2 The tee command copies standard input to standard output and also to any files included in the tee (like the letter or pipe). Display a system's IP address and write that to a file, using concatenate (cat) at the same time. (/sbin/ifconfig | tee ipaddress.txt | cat)

4.3 Another environment feature to explore is the ps or process status command. A number of programs can be run by a one or more users simultaneously, including helping programs called daemons (ps afux | less). The 'a' option list the processes of all users, the 'f' shows job hierarchy, the 'u' option provides additional information about each process, and the 'x' option includes non-user programs such as daemons. This is piped through less.

Redirection cont...

4.4 To redirect output use the '>' symbol. To redirect input (for example, to feed data to a command) use the '<'. Concatenation is achieved through the use of '>>' symbol. The command 'w' acts like a combination of who, uptime and ps -a. This is redirected to the file list.txt (w > list.txt). Redirection can be conducted in multiple directions (e.g., grep -i searchstring < searchfile.txt > foundit.txt)

4.5 Redirections can be further modified by placing a number next immediately before the redirector, which affects which stream is being used for redirection. These numbers are 0 for standard input, 1 for standard output and 2 for standard error. Standard error can also to be redirected to the same destination that standard output is directed to using 2>&1; it merges stderr (2) into stdout (1).

4.6 Process substitution can be used with redirection to run a comparison between directories (e.g., sdiff <(ls dir1) <(ls dir2)).

Files and Editing on the Command Line

5.1 Linux expresses its files as words made up of pretty much any characters, excepting the slash (/) which is used for directory navigation. In general however it is a good idea to avoid filenames with punctuation marks, non-printing characters (including spaces) as these can lead to some difficulties and annoyances, especially on the command-line level. It is a convention to use underscores instead of spaces e.g., this_is_a_long_name.txt

5.2 Linux is case-sensitive with its filenames. This means that that list.txt is a different file to LIST.TXT, or even lisT.txT. Files do not usually require a program association suffix, although you may find this handy. The file list can be opened by a text-editor just as easily as list.txt.

5.3 There are three text editors usually available on Linux systems on the command-line. The first is nano, a very easy to use clone from the Pine email client that uses control keys with a the equivalent of a "shortcut bar". The hefty EMacs (Editor Macros) editor and the the "screen orientated" modal text editor vi. Vi or Vim are often installed as the default text editor.

Ownership

6.1 Note the ownership and permissions of a file in user, group, and other (ls -l list.txt). The first character indicates the file type; a "-" for a regular file, a "d" for a directory, and "l" for a symbolic link, "b" for block devices, "c" for serial character devices. After the first character the notation should be understood in terms of three groups of three. The first group refers to what the owners can do with the file, the second group what group members can do and the third group what other users can do. The triplet of characters in each group are usually "r" for "readable", "w" for writable" and "x" for executable.

6.2 It is also possible to encounter a "s" for setuid. This is only found in the execute field. If the execute bit for the owner is set to "s" the set user ID bit is set causing any user or process that run the executable to have access to system resources as though they are the owner of the file. Finally there is "t", "save text attribute", or more commonly known as "sticky bit". This allows a user to delete or modify only those files in the directory that they own or have write permission for. A typical example is the /tmp directory, which is world-writeable (ls -l /tmp). Everyone can read, write, and access the directory, the "t" indicates that only the user that created a file in this directory can delete that file.

Ownership cont.

6.2 To chmod a file you have to own it. The command is : chmod [option] [symbolic | octal] file. The symbolic or octal option is two ways of setting permissions. For symbolic notation, first establish the user reference, "u" (user), "g" (group), "o" (others), or "a" (all). Determine the operation that is going to be expressed, either "+" (add the mode), "-" remove the mode, or "=" (equals, mode only equals that expression). Finally, specify the mode permissions as described above, "r" (read), "w" (write), "x" (execute), "s" (setuid, setgid), "t" (sticky).

6.3 Another method of using the chmod command is to use octal rather than symbolic notation. In octal notation a three or four digit base-8 value is presented derived from the sum of the component bits, so the equivalent of "r" in symbolic notation adds 4 in octal notation (binary 100), "w" in symbolic notation adds 2 in octal notation (binary 010) and "x" adds 1 in octal notation (binary 001). The sum of the three (or four components) thus makes up an alternative exact notation for the chmod command (e.g., chmod u+rw, g+r, o+r is the same as chmod 0644 filename).

Ownership cont.

6.4 Usually, only those with superuser (root) access make use of the chown (change owner) command. The general syntax for this is chown [option] [user:group] [file | directory]. Usually group is optional on the grounds that users are usually provided ownership. A common use is to provide ownership to web-writeable directories e.g., (chown -R www-data:www-data /var/www/files).

6.5 Remove read permissions to list.txt (chmod -r list.txt, ls -l), edit list.txt (nano list.txt), remove execute rights to the directory (chmod -x listdir) and attempt to change there. Reverse your actions (chmod +x listdir, chmod +r list.txt)

Links, Comparison

7.1 Make a copy of the list.txt file (cp list.txt list2.txt). Create a directory (mkdir listdir). Move the copy into the directory (mv list2.txt listdir). Change your context to that directory (cd listdir). Edit and modify the file (nano list2.txt). Rename the file to the same name as the original (mv list2.txt list.txt). Compare the two files by timestamp (ls -l list.txt, ls -l ../list.txt). Compare via the diff command and the side-by-side diff (diff list.txt ../list.txt, sdiff list.txt .../list.txt).

7.2 Create a symbolic link between the /tmp directory and temp (ln -s /tmp temp). Change to the directory and conduct a listing (cd temp, ls -l). Change out of the directory, and remove the link (cd .., rm temp). A symbolic link associates the file with an abstract location of another file, even across system boundaries and with directories. A hard link associates with an actual file.

Searches, Wildcards, History

8.1 Return to the home directory and search for *.txt files (cd ~, find . -name "*.txt"). Search within a collection of files for a keyphrase (grep -ir bash ~/*). Where there are multiple results, grep will also display the filename. Compressed or gzipped files can be searched with zgrep.

8.2 The wildcard you see most often is * (asterisk), but we'll start with something simpler: ? (question mark). When it appears in a filename, the ? matches any single character. The * wildcard matches any character or group of zero or more characters. For example, *.c matches all files whose names end with .c. Brackets allow for or scopes (e.g., letter[A-Z].txt).

8.3 Search through the history of commands with cursor keys. Search for command phrases with Cntrl-R. Repeat a command with !!. Repeat a command with regular expressions to change the command (e.g., $!!:s/html$/php$)

System Information

9.1 Use the disk usage command for a directory with the -s (summary) and -h (human) format (du -sh). Runs a disk usage in summary, sorts in order of size, cuts to include only the first two fileds, and exports to the file diskuse.txt. The "\n" is to ignore spaces in filenames. (du -sh * | sort -nr | cut -f2 | xargs -d "\n" du -sh > diskuse.txt)

9.2 The commands head and tail print the first and last ten lines of a file by default. The former is often useful to determine what sort of file one is looking at. Tail can be used when compiling programs to see the output in real-time, for example tail -f compile.log .

9.3 A typical command to access system information is uname (unix name), with the simple syntax uname [options]. The most common command is (uname -a). Another useful source for system information is the /proc directory; (cat /proc/cpuinfo, cat /proc/filesystems, cat /proc/meminfo etc).

Regular Expressions, Scripting

10.1 In order to make the best use of these scripting languages knowledge of pattern matching regular expressions is helpful. Whilst simple expressions have a literal character to character match ("a"="a"), regular expressions also have meta-characters, some of which are ^ beginning-of-line anchor (e.g., /^luv/), $ end of line anchor (e.g., /luv$/), . matches one character (e.g., /l.v/), * matches zero or more of the preceding characters (e.g., /*luv/), [ ] matches one in the set (e.g., /[Ll]uv/), \ used to escape a metacharacter (e.g., /luv\./)

10.2 The commands grep, sed and awk are three well known and useful utilities that make significant use of regular expressions. Others include sed, a stream-editor, awk is a field-orientated pattern programming language. Perl is also used a substitute.

10.3 Insert five spaces at the beginning of every line in list.txt (sed -i 's/^/   /' list.txt). Substitute all instances of 'bash' with 'mash' (sed -i '/bash/mash/g' list.txt). Print the second and fourth column from list.txt (awk '{print $1 "," $3}' list.txt). Awk defaults to the space as the internal field separator; the most common others are a comma, a tab and a colon, represented as awk -F",", awk -F"\t" and awk -F":", respectively.

Regular Expressions, Scripting cont.

10.4 The most basic form of scripting simply follows commands in sequence. For example a shell script which adds the date and time to an archive filename would look like:

```
#!/bin/bash
BU=/home/user-$(date +%Y%m%d).tgz
tar cvfz $BU /home/user/
```

10.5 Conditions may be expressed in different structures depending on the test of the conditionals. A single test of conditions and commands can be expressed through an if/then/fi structure. A single test with an alternative set of commands is expressed if/then/else/fi. Finally, a switch-like structure can be constructed through a series of elif statements in a if/then/elif/elif/.../else/fi structure. Conditionals can also be interrupted and resumed using the 'break' and 'continue' statements. The break command terminates the loop (breaks out of it), while continue causes a jump to the next iteration (repetition) of the loop, skipping all the remaining commands in that particular loop cycle.

10.6 Handy extract program at http://levlafayette.com/files/extract.txt

Remote Access and Transfers, Deletion

11.1 The Linux command line is excellent at networking. A usual distribution has the core tools ssh (secure shell), sftp (secure file transfer), and scp (secure copy) built into the shell. These are considered more secure than older UNIX tools (telnet, ftp, rcp) which sent passwords in plain-text.

11.2 The core command for ssh is ssh username@hostname. The core command for secure copy is scp source destination e.g. scp lev@tango.vpac.org:/home/lev/luv-talk-cli.pdf .)

11.3 When deleting files and directories extreme care is emphasised because there is no "trashcan" to easily undelete files. Change to the home directory and delete the list.txt file (cd ~, rm list.txt). Attempt to delete the directory (rmdir listdir). Forcibly and recursively delete (rm -rf listdir). Consider what could happen if a user wanting to delete *.bak files types rm * .bak instead of rm *.bak. Consider what you happen if a root user types rm -rf /.