CIS 18A
Introduction to Linux / Unix

Basic Utilities

De Anza College
Instructor: Clare Nguyen

Part 1

Topics:
- General Command Line Format
- Commands to start and end a session
- Commands to look up users
- Commands to look up terminal and system information

Working with Linux

Here are some key points to keep in mind when working with Linux:
- Case sensitive: Linux considers uppercase letters to be different than lowercase letters. This means that the utility named cp is not the same as Cp or CP
- To run a utility, always:
  - make sure you see the shell prompt first. The line you type your input is called the *command line.*
  - start the command line with the command name.
  - hit the enter key when you're done.
- When you work with Linux, you interact with the shell.
- The shell works in a cycle:
  1. prints the shell prompt to indicate that it's ready for your input
  2. interprets the command you type in
  3. runs the corresponding utility that you request
  4. sends the output to where you request
  1. prints the shell prompt again

General Command Format (1 of 4)

- Commands that you type on the command line has the format:

  command_name options arguments
- The *command_name* is one word and is required
- The *options* are optional. There can be 0, 1, or more options.
- The *arguments* maybe required or optional, depending on the utility you select. There can be 0, 1, or more arguments.
- The command_name, options, and arguments are separated by at least one space. Typically one space is used, but more than one space is okay.
- The command_name must be the first word on the command line. The options and arguments can be anywhere else on the command line but after the command_name.

General Command Format (2 of 4)

More on command_name
- The command name tells the shell which utility you want to run.
- Each utility has a unique name. It is one short word or an abbreviation of a word.
- The command name describes what the utility does.
- Examples: find, who, cp (for copy), ln (for link)
- In this class you will learn about different utilities so you can choose the most efficient one to do a task.

General Command Format (3 of 4)

More on options:
- Without an option, the command works in the *default mode.*
  For example, by default a command lists filenames in alphabetical order. With an option, the same command will list filenames in order of creation date.
- Each option start with a + or – symbol, followed by one letter. Most options start with a – symbol.
- When using multiple options, the options are separated by at least one space. Alternatively, you can put together multiple options with no space in between, and with one – symbol in front for options that start with a –. The same for + options.
- With multiple options, the order you list the options is not important
- Examples:
  - command ls with no option: ls
  - command ls with 1 option: ls –a
  - command ls with 2 options: ls –a –f
    or ls –af
    or ls –fa

General Command Format (4 of 4)

More on arguments:

- Arguments are input to the utility. The utility performs its task on the input argument. For example: the copy utility will require 2 arguments: the source file and the destination file.
- Depending on the utility, arguments are either required or optional.
- If using multiple arguments, list them in the order required or the order you want, separated by at least 1 space.
- Example:
  - Command ls with 1 argument: ls fileA
  - Command ls with 2 arguments: ls fileA fileB
  - Command ls with 3 arguments and 2 options:
    It's more common for the options to appear before the arguments: ls -af fileA fileB fileC
    This is less common: ls fileA fileB fileC -af

Basic commands

- The basic commands covered in this section can be grouped into several common categories. They are commands dealing with:
  - The start and end of a session: passwd, exit
  - Information on users on the system: who, whoami, w, finger
  - Information about your system or your terminal: tty, stty, uname, clear
  - Getting labs done: man, lpr, script, cat
  - General use: bc, date, cal, echo
- Disclaimer: The lecture notes on a specific command do not cover *all* arguments and options that a command can have. Rather, they cover the basic and common usage of a particular command.

exit

- The exit command is used to exit out of a current process.
- When you've successfully logged in to the system, the shell *process* starts running to wait for your command. When you type exit, the OS ends the shell process by closing the shell and you are logged out.
- The time from when you log in up to when you log out is one session or one working session.
- During a session you can start a second or third or more processes. Each time you run exit, you will get out of whatever current process you are in. So if you start 2 more processes from when you log in, you will need to run exit 3 times to completely log out.
- The exit command sends an interrupt signal to the OS. Another way to send the interrupt signal is to use the combination of the control and d keys, abbreviated control-d

passwd

- The passwd command allows you to change your password.
- Often in industry you are required to change your password on a regular basis for security reasons.
- System administrators set up the rules that your password needs to follow, such as at least 8 characters, or must be a combination of letters and non-letters, etc.
- If your password doesn't follow the rules, passwd will remind you of the rules.
- To run:
  passwd
  - Type in old password (security measure to prevent someone from maliciously changing your password on you).
  - Type in new password twice, once at each prompt.
  - You should get a confirmation that it has been changed successfully.

who, whoami, w, finger (1 of 2)

Commands to find information on users who are in the system.

- who: gives you a snapshot of users who are currently logged in: user ID, terminal ID, date and location logged in.

```
[cnguyen@voyager ~]$ who
cnguyen  pts/1        2016-09-03 12:37 (c-67-180-237)
cnguyen  pts/2        2016-09-03 13:07 (c-67-180-237)
```

- whoami: shows your user id only. Useful when you log in to multiple sessions with different user IDs.
- w: similar to who, but with more information such as the task the user is running.

```
[cnguyen@voyager ~]$ w
13:17:06 up 7 days, 21:15,  2 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
cnguyen  pts/1    c-67-180-237-243 12:37    38:24  0.00s  0.00s man ls
cnguyen  pts/2    c-67-180-237-243 13:07     0.00s  0.02s  0.00s w
```

- finger: similar to who, but with more information such as the user's name.

```
[cnguyen@voyager ~]$ finger
Login    Name           Tty      Idle  Login Time   Office     Office Phone
cnguyen  Clare Nguyen   pts/142        Sep 3 12:37 (c-67-180-237)
cnguyen  Clare Nguyen   pts/2          Sep 3 13:07 (c-67-180-237)
```

who, whoami, w, finger (2 of 2)

- finger can also accept an argument of a user ID, or a last name, or first name. When given an argument, finger will find one or more users that match the argument and print more detailed information on each match.
- Example: find all users with the name 'nguyen'

```
[cnguyen@voyager ~]$ finger nguyen
Login: bachlan                          Name: Bachlan Nguyen
Directory: /home/staff/bachlan          Shell: /bin/bash
Never logged in.
Mail last read Wed Aug 27 16:13 2016 (PDT)
No Plan.

Login: cnguyen                          Name: Uyen Clare Nguyen
Directory: /home/staff/cnguyen          Shell: /bin/bash
On since Wed Sep  3 13:07 (PDT) on pts/2 from c-67-180-237 (messages off)
No mail.
No Plan.
```

tty, stty, uname, clear

Commands related to your terminal or system

- tty: (terminal type) shows the ID of the terminal that you are logged in at.

```
[cnguyen@voyager ~]$ tty
/dev/pts/1
```

- stty: shows basic settings of the terminal you are logged in at
  - -a option: shows all settings.

```
[cnguyen@voyager ~]$ stty
speed 38400 baud; line = 0;
-brkint -imaxbel
```

- uname: shows basic system information.
  - -a option: shows all system information

```
[cnguyen@voyager ~]$ uname -a
Linux voyager.deanza.edu 2.6.18-92.1.10.el5 #1 SMP Wed Jul 23
03:56:11 EDT 2008 x86_64 x86_64 x86_64 GNU/Linux
```

- clear: clears the screen.

Part 2

Topics:

- Commands to get information on a utility
- Commands to work with text files
- Command to capture screen output
- Calculator
- Commands to look up time and day
- Command to print text to screen

man

- Each utility has a help page that explains how the utility works and lists all options for the utility.
- The help page is part of the online manual that is typically packaged with Linux, and the help page is commonly called the man page (short for manual page).
- To see the man page for a utility: man utility_name
  - The required argument is the utility name.
  - All information about the utility is shown one page at a time.
  - To go to the next page: space bar
  - To go to the previous page: b (for back)
  - To get back to the shell prompt: q

lpr, cat

Commands to work with text files

- lpr: (line printer) prints a file.
  - Filename is the required argument.
- cat: (concatenate) shows the content of a file.
  - Filename is required if you want to see the content of a specific file.

cat will be covered in more detail in the next module.

script (1 of 3)

- script: captures output on screen into a file
  - Useful when you want to show proof of work, or show program output, or do error reporting by capturing the error on screen.
  - When script runs, every character that appears on the screen also appears in an output file that you choose. When you exit out of script, the output file is saved and closed.
  - A filename is a recommended argument. If you don't give a filename, a default filename (which is *typescript*) is used.
  - If you give the same filename as an existing file, the existing file will be overwritten.
  - The –a option will cause script to append to an existing file, rather than overwrite it.
  - Don't run script when another script session is already running. This causes the output file to be very big and unreadable.
  - Don't make too many typing mistakes when script is running. The mistaken characters and all the backspacing characters to fix the typing mistakes will all be recorded and appear as one long, ugly string in your output file.

script (2 of 3)

Example of using script to capture screen output into a file called *sample*:

```
[cnguyen@voyager ~]$ script sample          <- start screen capture into file called sample
Script started, file is sample
[cnguyen@voyager ~]$ who
tran     pts/1        2016-09-03 14:05 (153.18.21.226)
cnguyen  pts/2        2016-09-03 14:21 (c-67-180-237)
[cnguyen@voyager ~]$ finger clare
Login: cnguyen                          Name: Clare Nguyen
Directory: /home/staff/cnguyen          Shell: /bin/bash
On since Wed Sep 3 14:21 (PDT) on pts/2 from c-67-180-237 (messages off)
No mail.
No Plan.
[cnguyen@voyager ~]$ exit                   <- end screen capture
exit
Script done, file is sample
[cnguyen@voyager ~]$ script -a sample       <- start screen capture again
Script started, file is sample                 and append into file called sample
[cnguyen@voyager ~]$ finger
Login    Name           Tty   Idle Login Time   Office     Office Phone
cnguyen  Clare Nguyen   pts/2       Sep  3 14:21 (c-67-180-237)
tran                    pts/1 15    Sep  3 14:05 (153.18.21.226)
[cnguyen@voyager ~]$ exit                   <- end screen capture
exit
Script done, file is sample
```

script (3 of 3)

Resulting sample file from the previous script sessions:

```
Script started on Wed 03 Sep 2016 02:21:32 PM PDT      <- first script session
[cnguyen@voyager ~]$ who
tran     pts/1        2016-09-03 14:05 (153.18.21.226)
cnguyen  pts/2        2016-09-03 14:21 (c-67-180-237)
[cnguyen@voyager ~]$ finger clare
Login: cnguyen                          Name: Clare Nguyen
Directory: /home/staff/cnguyen          Shell: /bin/bash
On since Wed Sep  3 14:21 (PDT) on pts/2 from c-67-180-237 (messages off)
No mail.
No Plan.
[cnguyen@voyager ~]$ exit
exit
Script done on Wed 03 Sep 2016 02:22:00 PM PDT
Script started on Wed 03 Sep 2016 02:22:22 PM PDT      <- second script session
[cnguyen@voyager ~]$ finger
Login    Name           Tty      Idle Login Time   Office     Office Phone
cnguyen  Clare Nguyen   pts/2        Sep  3 14:21 (c-67-180-237)
tran                    pts/1    15  Sep  3 14:05 (153.18.21.226)
[cnguyen@voyager ~]$ exit
exit
Script done on Wed 03 Sep 2016 02:22:46 PM PDT
```

bc

- bc: (**b**asic **c**alculator)
  - This calculator is actually not so basic, since it can support some programming (see the man page for bc). However in this class we will concentrate on basic arithmetic.
  - To run: bc
  - To stop: quit
  - To do basic arithmetic, enter the arithmetic expression with or without space: 2 + 3 or 6/3
    After the enter key, the result will appear on the next line.
  - The basic operators: + (add), - (subtract), * (multiply), / (divide)
  - To change the number of digits after the decimal point:
    scale=*n* where n is the number of digits
    Example: scale=2 for 2 digits after the decimal point
    scale=4 for 4 digits after the decimal point
    scale=0 for integer only (no digit after decimal point)

date, cal

- date: shows the current time and date.

```
[cnguyen@voyager ~]$ date
Wed Sep  3 20:18:13 PDT 2016
```

- cal: shows the calendar.
  - With no argument: shows the current month.
  - With 1 argument: shows the year given by the argument, note that 2009 is not the same year as 09.
  - With 2 arguments: the first argument is for the month (only values 1-12 are valid), the second argument is for the year.

```
[cnguyen@voyager ~]$ cal 2 2009
    February 2009
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
```

echo

- echo: prints to screen (echoes) the text that is given as argument(s)
  - Can accept one or many arguments.
  - Arguments are text words, for now.
  - Useful to display a text string to screen, and in shell scripting, to send output of the script to screen.
  - To run: echo any text string

Next stop: Regular Files