Some Unix Commands That You Might Find Useful in CPSC 221

Date of Last Update: January 15, 2010

How to display your current directory:

→ pwd = print working directory

How to change from your current directory (folder) to your home (default) directory:

→ cd = change directory

How to change from your current directory to the (existing) sub-directory called lab_01:

→ cd lab_01 = change directory

How to create a sub-directory in Unix:

→ mkdir lab_02 = make directory

How to go to the parent directory:

→ cd .. = change directory (there is a space after "cd")

How to bring up the emacs text editor and at the same time start with an empty file called myProg1.cpp, placing it in the current directory:

→ emacs myProg1.cpp & = launch the emacs text editor, and do so in a brand new window (the & means background process (new window, so that you can continue typing Unix commands in the current window))

Use cntl-x, cntl-c to save and exit emacs.

Use cntl-x, cntl-s to save the current version of the file, but continue working in emacs.

See WebCT: "Extra Tutorial Notes" for summaries of emacs and vi (another Unix text editor)

How to bring up the emacs text editor in the current window (perhaps because it's a short file, and I just want to do a quick update):

→ emacs -nw hours = launch the emacs text editor in the current window, opening the file called "hours" (or an empty file which will be named "hours", if it doesn't currently exist)

How to copy the file called myFile1.cpp to myFile2.cpp, so that I can begin editing a new version of the file, or to make a backup copy (for example):

→ cp myFile1.cpp myFile2.cpp = copy file (it will overwrite the old myFile2.cpp, if it exists)

How to rename the file called myFile1.cpp to factorial.cpp:

→ mv myFile1.cpp factorial.cpp = move file (it will overwrite the old factorial.cpp, if it exists)

How to delete the file called myOldfile.cpp:

→ rm myOldfile.cpp = remove file

How to delete the directory called my_dir2:

→ rmdir my_dir2 = remove directory

How to list the files in my current directory:

→ ls = use "ls –l" if you want a long listing that includes dates and times

How to display a text file, page by page, on the screen, without having it scroll off the screen:

→ more myTextFile.whatever = more means "page at a time" (hit the space bar to advance to the next page)

How to display an entire text file without going page to page. This is good if you only want to see the last page, or if you want to see if the file is non-empty:

→ cat myTextFile.whatever = cat means concatenate (to the input buffer)

How to compile the GNU C++ program called myProg3.cpp, and call the executable file by the default name "a.out":

→ g++ -Wall myProg3.cpp = GNU C++ compiler, turn on all warnings (very useful), compile myProg3.cpp, and name the resulting executable program "a.out" (the default). The executable will be stored in the current directory, and will overwrite the previous a.out file, if it exists.

How to run the "a.out" program, without any parameters on the command line:

→ a.out = name of executable, all by itself (use "a.out 36 myString" if a.out is expecting 2 parameters from the command line, here an integer followed by a string)

How to compile a GNU C++ program called myProg3.cpp, and call the executable file "airport":

→ g++ -Wall –o airport myProg3.cpp = -o flag means "object file" follows … it will overwrite the previous file called airport, if it exists

→ g++ -Wall myProg3.cpp –o airport = same

(Warning! Do not write: "g++ -Wall –o myProg3.cpp airport", as this will destroy your myProg3.cpp source file (unless it's open in a background window, in which case just save it again).

How to run the "airport" program, without any parameters on the command line:

→ airport = name of executable, all by itself (use “airport 36 myString” if airport is expecting 2 parameters from the command line, here an integer followed by a string)

How to display the last n commands that I entered:

→ history

Re-run the last command that I entered that starts with the letters “g+”:

→ !g+ = note: this can save a lot of typing

Display the documentation for a Unix command, like prstat or ls:

→ man prstat = manual pages

→ man ls

How to display all the processes that are running under my userid:

→ps –eaf | grep myuserid = ps means process status, the vertical bar means pipe(line) the output of the ps command into the “grep” (string search) program and output only those lines that contain “myuserid” as a string … otherwise, you may get a very large amount of irrelevant data (about processes that belong to other users)

Once I see my processes (from the above command), here’s how to kill a runaway program, or a window that seems to be frozen:

→ kill 2473 = kill/terminate process ID number 2473 (you get the process ID from the previous ps command; 2473 is just an example)

→ kill -9 2473 = kill force (if the previous kill command doesn’t work)

How do I kill the infinite loop that’s running in the current window:

→ cntl-c = press control + c

How do I see how much CPU time my running program is consuming, as well as the amount of memory that it is using? How do I see who’s hogging the CPU?

→ prstat –n 15 = process statistics (report currently active jobs); “-n 15” means restrict the number of output lines to the top 15 CPU time consumers

How to display the processor (host) name:

→ hostname = host name, e.g., “galiano.ugrad.cs.ubc.ca”

How to display the processor information for the server I’m currently using:

→ psrinfo –v = processor info, e.g., galiano.ugrad has 8 CPUs running at 1050 MHz, SPARC v9

How to display the userids currently logged on:

→ users

How to display the userids that are logged on, and their client machine names:

→ who

How to display the number of lines, words, and characters (bytes) in a file:

→ wc myFile.cpp = word count