CSE 374
Programming Concepts & Tools

Hal Perkins

Spring 2022

Lecture 2a – A Unix Command Sampler

Command line arguments

- Most options are given after the command name using a dash followed by a letter: **`-c`**, **`-h`**, **`-S`**, …
- Some options are longer words preceded by two dashes:
  `--count`, **`--help`**
- Options often can be combined: **`ls -l -a -r`** can be **`ls -lar`**
- Many programs accept **`--help`**; others provide help if run with no arguments
- Many commands accept a file name parameter; if it is omitted, the program will read from standard input

Directory commands

command	description
`ls`	list files in a directory
`pwd`	output the current working directory
`cd`	change the working directory
`mkdir`	create a new directory
`rmdir`	delete a directory (must be empty)

Relative naming

directory	description
`.`	the directory you are in ("working directory")
`..`	the parent of the working directory (`../..` is grandparent, etc.)
`~`	your home directory (on many systems, this is /home/***username***)
~*username*	***username***'s home directory
`~/Desktop`	your desktop

Shell/system commands

command	description
`man or info`	get help on a command
`apropos (man -k)`	search for commands by keyword
`clear`	clears out the output from the console
`exit`	exits and logs out of the shell

command	description
`date`	output the system date/time
`cal`	output a text calendar
`uname`	print information about the current system

- "man pages" are a very important way to learn new commands

File commands

command	description
`cp`	copy a file
`mv`	move or rename a file
`rm`	delete a file
`touch`	update a file's last-modified time stamp (or create a new empty file)

- CAUTION: the above commands do not prompt for confirmation, so it's easy to overwrite/delete a file
- This setting can be overridden (how?)

File examination

command	description
`cat`	output a file's contents on the console
`more, less`	output a file's contents, one page at a time
`head, tail`	output the first or last few lines of a file
`wc`	count words, characters, and lines in a file
`du`	report disk space used by a file(s)
`diff`	compare two files and report differences

- Suppose you are writing a paper, and the teacher says it can be anything as long as it is at least 200 words long and mentions chocolate...

Searching and sorting

command	description
`grep`	search a file for a given string
`sort`	convert an input into a sorted output by lines
`uniq`	strip duplicate lines
`find`	search for files within a given directory
`locate`	search for files on the entire system
`which`	shows the complete path of a command

- `grep` is a very powerful search tool; more later...
- *Exercise* : Given a text file `students.txt`, display the students arranged by the reverse alphabetical order of their last names.
  - Can we display them sorted by first name?

Keyboard shortcuts

^***KEY*** means hold `Ctrl` and press ***KEY***

key	description
`Up arrow`	repeat previous commands
`Home/End or ^A/^E`	move to start/end of current line
`"`	quotes surround multi-word arguments and arguments containing special characters
`*`	"wildcard" , matches any files; can be used as a prefix, suffix, or partial name
`Tab`	auto-completes a partially typed file/command name
`^C or ^\`	terminates the currently running process
`^D`	end of input; used when a program is reading input from your keyboard and you are finished typing
`^Z`	suspends (pauses) the currently running process
`^S`	don't use this; hides all output until ^G is pressed

File system

directory	description
`/`	root directory that contains all others (drives do not have letters in Unix)
`/bin`	programs
`/dev`	hardware devices
`/etc`	system configuration files - `/etc/passwd` stores user info - `/etc/shadow` stores passwords
`/home`	users' home directories
`/media,/mnt,...`	drives and removable disks that have been "mounted" for use on this computer
`/proc`	currently running processes (programs)
`/tmp, /var`	temporary files
`/usr`	user-installed programs

Process commands

command	description
`ps`	list processes being run by a user; each process has a unique integer id (PID)
`top`	show which processes are using CPU/memory; also shows stats about the computer *Keeps executing until killed!*
`kill`	terminate a process by PID
`killall`	terminate processes by name

- use **`kill`** or **`killall`** to stop a runaway process (infinite loop)
- similar to **`^C`** hotkey

Background processes

command	description
`&`	(special character) when placed at the end of a command, runs that command in the background
`^Z`	(hotkey) suspends the currently running process
`fg` `bg`	resumes the currently suspended process in either the foreground or background

- You would like some processes to continue while you are doing other things – maybe your editor, maybe a browser, etc.
- You can do this by running some processes “in the background”, so the shell doesn’t have to wait until those processes finish; ex:

  ```
  $ emacs &
  ```
- If you forget to use **`&`**, suspend your process with **`^Z`**, then run **`bg`**