

---

CSE 374

# Programming Concepts & Tools

Brandon Myers

Winter 2015

Lecture 2a – A Unix Command Sampler

(Courtesy of David Notkin, CSE 303)

---

# 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**
- Parameters can be combined: **ls -l -a -r** can be **ls -lar**
- Many programs accept a **-help** parameter; 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
<code>ls</code>	list files in a directory
<code>pwd</code>	output the current working directory
<code>cd</code>	change the working directory
<code>mkdir</code>	create a new directory
<code>rmdir</code>	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/ <i>username</i> )
~ <i>username</i>	<i>username</i> 's home directory
~/Desktop	your desktop

# Shell/system commands

---

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

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

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

# File commands

---

command	description
<code>cp</code>	copy a file
<code>mv</code>	move or rename a file
<code>rm</code>	delete a file
<code>touch</code>	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? look in the man page for the answer!)

# File examination

---

command	description
<code>cat</code>	output a file's contents on the console
<code>more, less</code>	output a file's contents, one page at a time
<code>head, tail</code>	output the first or last few lines of a file
<code>wc</code>	count words, characters, and lines in a file
<code>du</code>	report disk space used by a file(s)
<code>diff</code>	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
<code>grep</code>	search a file for a given string
<code>sort</code>	convert an input into a sorted output by lines
<code>uniq</code>	strip duplicate lines
<code>find</code>	search for files within a given directory
<code>locate</code>	search for files on the entire system
<code>which</code>	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 ^\ (Note: the original image has a backslash after ^C)	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 <ul style="list-style-type: none"><li>▪ /etc/passwd stores user info</li><li>▪ /etc/shadow stores passwords</li></ul>
/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
<code>ps</code>	list processes being run by a user; each process has a unique integer id (PID)
<code>top</code>	show which processes are using CPU/memory; also shows stats about the computer <i>Keeps executing until killed!</i>
<code>kill</code>	terminate a process by PID
<code>killall</code>	terminate processes by name

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

# Background processes

---

command	description
<code>&amp;</code>	(special character) when placed at the end of a command, runs that command in the background
<code>^Z</code>	(hotkey) suspends the currently running process
<code>fg</code> <code>bg</code>	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`