CSE 303
Lecture 2

Introduction to bash shell

read *Linux Pocket Guide* pp. 37-46, 58-59, 60, 65-70, 71-72, 77-80

slides created by Marty Stepp
http://www.cs.washington.edu/303/

Lecture summary

- Unix file system structure
- basic shell commands
- commands for file manipulation, examination, searching
- keyboard shortcuts and special characters

Unix file system

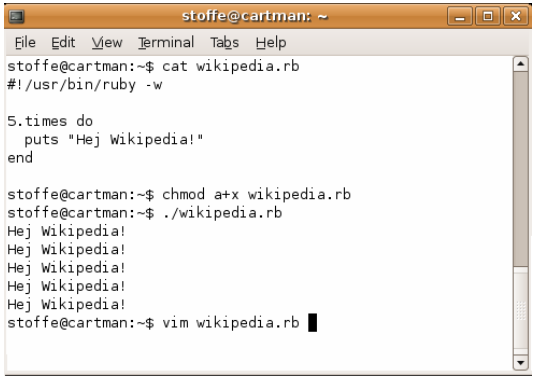
directory	description
`/`	root directory that contains all others (drives do not have letters in Unix)
`/bin`	programs
`/dev`	hardware devices
`/etc`	system configuration files ▪ `/etc/passwd` stores user info ▪ `/etc/shadow` stores passwords
`/home`	users' home directories
`/media`, `/mnt`, …	drives 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

Relative directories

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 commands

- many accept **arguments** or **parameters**
  - example: cp (copy) accepts a source and destination file path



- a program uses 3 streams of information:
  - stdin, stdio, stdout (standard in, out, error)

- **input**: comes from user's keyboard
- **output**: goes to console
- **errors** can also be printed (by default, sent to console like output)

- parameters vs. input
  - *parameters*: before Enter is pressed; sent in by shell
  - *input*: after Enter is pressed; sent in by user

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)

- some commands (`cd`, `exit`) are part of the shell ("builtins")
- others (`ls`, `mkdir`) are separate programs the shell runs

Command-line arguments

- most options are a `-` followed by a letter such as `-c`
  - some are longer words preceded by two `-` signs, such as `--count`
- parameters can be combined: `ls -l -a -r` can be `ls -lar`
- many programs accept a `--help` or `-help` parameter to give more information about that command (in addition to `man` pages)
  - or if you run the program with no arguments, it may print help info
- for many commands that accept a file name parameter, if you omit the parameter, it will read from standard input (your keyboard)
  - note that this can conflict with the previous tip

Shell/system commands

command	description
`man or info`	get help on a command
`clear`	clears out the output from the console
`exit`	exits and logs out of the shell

command	description
`date, time`	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

```
man ls
man man
```

File commands

command	description
`cp`	copy a file
`mv`	move or rename a file
`rm`	delete a file
`touch`	create a new empty file, or update its last-modified time stamp

- caution: the above commands do not prompt for confirmation
  - easy to overwrite/delete a file; this setting can be overridden (how?)

- *Exercise* : Given several albums of `.mp3` files all in one folder, move them into separate folders by artist.
- *Exercise* : Modify `HW4.java` to make it seem as though you finished writing it on March 15 at 4:56am.

File examination

command	description
`cat`	output a file's contents on the console
`more or 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 303...

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 actually 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?

Programming

command	description
`javac` ***`ClassName`***`.java`	compile a Java program
`java` ***`ClassName`***	run a Java program
`python, perl, ruby, gcc, sml, ...`	compile or run programs in various other languages

- *Exercise* : Write/compile/run a program that prints "Hello, world!"

```
$ javac Hello.java
$ java Hello
Hello, world!
$
```

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

Links

command	description
`ln`	create a link to a file
`unlink`	remove a link to a file

- **hard link**: Two names for the same file.

```
$ ln foo bar
```

  - the above command links `bar` as a duplicate name for `foo`
    - if one is modified, the other is too; if one is deleted, both will go away

- **soft (symbolic) link**: A reference to another existing file.

```
$ ln -s foo bar
```

  - the above command creates a reference bar to the file foo
    - `bar` can be used as though it were foo
    - but if `bar` is deleted, `foo` will be unaffected