# Lecture 3

bash shell continued:
processes; multi-user systems;
remote login; editors

1

1

## Lecture summary

- A bit more on combining commands
- Processes and basic process management
- Connecting to remote servers (`attu`)
  - multi-user environments
- Text editors

2

2

## Review: Redirection and Pipes

- ***command*** `>` ***filename***
  - Write the output of ***command*** to ***filename*** (>> to append instead)
- ***command*** `<` ***filename***
  - Use ***filename*** as the input stream to ***command***
- ***command1*** `|` ***command2***
  - Use the console output of ***command1*** as the input to ***command2***
- ***command1*** `;` ***command2***
  - Run ***command1*** and then run ***command2***
- ***command1*** `&&` ***command2***
  - Run ***command1***, if completed without errors then run ***command2***

3

3

## Tricky Examples

- Amongst the top 250 movies in movies.txt, display the third to last movie that contains "The" in the title when movie titles are sorted.
- The wc command can take multiple files: wc names.txt student.txt
  - Can we use the following to wc on every txt file in the directory?
    - `ls *.txt | wc`
- Find the disk space usage of the man program
  - Hints: use `which` and du…
  - Does `which man | du` work?

Answers posted in lecture_commands.txt after lecture

4

4

# Command Substitution

***command1*** `$(`***command2***`)`

- run ***command2*** and pass its console output to ***command1*** as a parameter;
- best used when ***command2***'s output is short (one line)

- Finish the example!
  - `du $(which man)`

5

5

# xargs

| command | description |
|---|---|
| `xargs` | run each line of input as an argument to a specified command |

- `xargs` allows you to repeatedly run a command over a set of lines
  - often used in conjunction with `find` to process each of a set of files
- Example: Remove all my .class files.

```
find ~ -name "*.class" | xargs rm
```

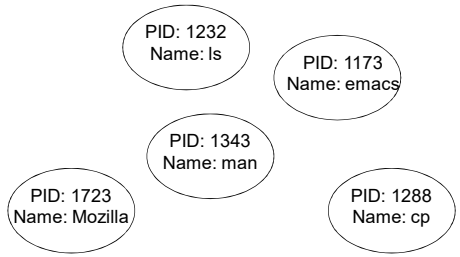
- Find the disk usage of `man` using `xargs`
  - `which man | xargs du`

6

6

# Processes

- **process**: a program that is running (essentially)
  - when you run commands in a shell, it launches a process for each command
  - Process management is one of the major purposes of an OS




7

7

# Process commands

| command | description |
|---|---|
| `ps or jobs` | 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 |
| `kill` | terminate a process by PID (sometimes –KILL is needed) |
| `killall` | terminate several processes by name |

- use `kill` or `killall` to stop a runaway process (infinite loop)
  - similar to ^C hotkey, but doesn't require keyboard intervention

8

8

# 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 |

- On the VM, if you run a graphical program like `gedit` or `emacs` from the shell, the shell will lock up waiting for the graphical program to finish
  - instead, run the program in the background, so the shell won't wait:
    `$ emacs homework1.txt &`
  - if you forget to use & , suspend emacs by hitting ^Z at the TERMINAL (NOT in emacs), then run bg from the terminal
  - lets play around with an infinite process…

9

9

# Connecting with ssh

| command | description |
|---|---|
| `ssh` | open a shell on a remote server |

- Linux/Unix are built to be used in multi-user environments where several users are logged in to the same machine at the same time
  - users can be logged in either locally or via the network
- You can connect to other Linux/Unix servers with `ssh`
  - once connected, you can run commands on the remote server
  - other users might also be connected; you can interact with them
  - can connect even from other operating systems

10

10

# Multi-user environments

| command | description |
|---|---|
| `whoami` | outputs your username |
| `passwd` | changes your password |
| `hostname` | outputs this computer's name/address |
| `w or finger` | see info about people logged in to this server |
| `write` | send a message to another logged in user |

- *Exercise* : Connect to `attu`, and send somebody else a message.

11

11

# Network commands

| command | description |
|---|---|
| `links or lynx` | text-only web browsers (really!) |
| `ssh` | connect to a remote server |
| `sftp or scp` | transfer files to/from a remote server (after starting sftp, use get and put commands) |
| `wget` | download from a URL to a file |
| `curl` | download from a URL and output to console |
| `alpine, mail` | text-only email programs |

12

12

# Text editors

| command | description |
|---|---|
| `pico or nano` | simple editors |
| `emacs` | More advanced text editor |
| `vi or vim` | More advanced text editor |

- **most advanced Unix/Linux users learn `emacs` or `vi`**
  - I would recommend you try to pick up the basics of one of these.
  - Your choice!

13

13

# Aliases

| command | description |
|---|---|
| `alias` | assigns a pseudonym to a command |

`alias` ***name=command***

- must wrap the command in quotes if it contains spaces
- **Do not put spaces on either side of the =**

- Example: When I type `q` , I want it to log me out of my shell.
- Example: When I type `ll` , I want it to list all files in long format.

```
alias q=exit
alias ll="ls -la"
```

- *Exercise* : Make it so that typing `q` quits out of a shell.
- *Exercise* : Make it so that typing `woman` runs `man`.
- *Exercise* : Make it so that typing `attu` connects me to `attu`.

Answers posted in lecture_commands.txt after lecture

14

14

# Mounting remote files

| command | description |
|---|---|
| `sshfs` | mount and interact with remote directories and files |

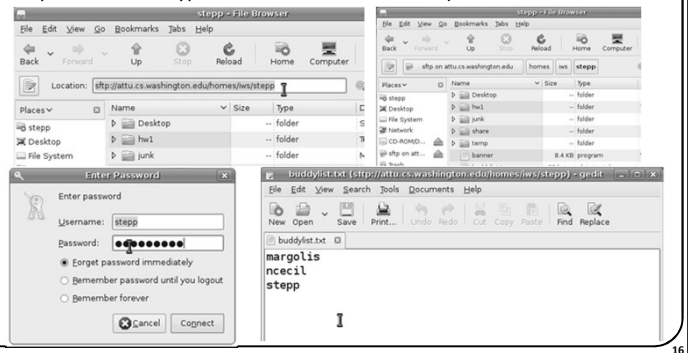
- An alternate usage model to remotely connecting to servers is mounting remote directories and files and work on them locally
  - once mounted, use remote directories and files as if they were local

15

15

# Remote editing

- Gnome's file browser and `gedit` text editor are capable of opening files on a remote server and editing them from your computer
  - press Ctrl-L to type in a network location to open




16

16

# Basic Vim Commands

- :w Write the current file
- :wq Write the current file and exit.
- :q! Quit without writing
- To change into insert mode: i or a
  - Use escape to exit
- search forward /, repeat the search backwards: N
- Basic movement:
  - h l k j character left, right; line up, down (also arrow keys)
  - b w word/token left, right
  - ge e end of word/token left, right
  - 0 $ jump to first/last character on the line
- x delete
- u undo

17

17