CSE 390a
Lecture 3
bash shell continued:
processes; multi-user systems; remote login; editors

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

Tricky Examples

- 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`
- Amongst the top 250 movies in movies.txt, display the third to last movie that contains “The” in the title when movies titles are sorted.
- Find the disk space usage of the man program
  - Hints: use `which` and `du`...
  - Does `which man | du` work?

The back-tick

`command1 ` `command2`
- run command2 and pass its console output to command1 as a parameter;
  - ` ` is a back-tick, on the `~` key; not an apostrophe
- best used when command2’s output is short (one line)

- Finish the example!
  - `du 'which man'`

xargs

<table>
<thead>
<tr>
<th>command</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xargs</td>
<td>run each line of input as an argument to a specified command</td>
</tr>
</tbody>
</table>

- 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`
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

- PID: 1232
  - Name: ls

- PID: 1173
  - Name: gedit

- PID: 1288
  - Name: cp

- PID: 1723
  - Name: Mozilla

- PID: 1343
  - Name: man

Process commands

<table>
<thead>
<tr>
<th>command</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ps</td>
<td>list processes being run by a user; each process has a unique integer id (PID)</td>
</tr>
<tr>
<td>top</td>
<td>show which processes are using CPU/memory; also shows stats about the computer</td>
</tr>
<tr>
<td>kill</td>
<td>terminate a process by PID</td>
</tr>
<tr>
<td>killall</td>
<td>terminate several processes by name</td>
</tr>
</tbody>
</table>

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

Background processes

- If you run a graphical program like gedit 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:
    - `$ gedit resume.txt &`
  - if you forget to use &
  - suspend gedit with `^Z`, then run `bg`
  - let’s play around with an infinite process…

- `&` (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

Connecting with ssh

- 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

The attu server

- **attu**: The UW CSE department’s shared Linux server
  - connect to attu by typing:
    - `ssh attu.cs.washington.edu`
    - (or `ssh username@attu.cs.washington.edu` if your Linux system’s user name is different than your CSE user name)

Multi-user environments

<table>
<thead>
<tr>
<th>command</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>whoami</td>
<td>outputs your username</td>
</tr>
<tr>
<td>passwd</td>
<td>changes your password</td>
</tr>
<tr>
<td>hostname</td>
<td>outputs this computer’s name/address</td>
</tr>
<tr>
<td>w or Finger</td>
<td>see info about people logged in to this server</td>
</tr>
<tr>
<td>write</td>
<td>send a message to another logged in user</td>
</tr>
</tbody>
</table>

- Exercise: Connect to attu, and send somebody else a message.
Network commands

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

Text editors

<table>
<thead>
<tr>
<th>command</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pico or nano</td>
<td>simple but crappy text editors (recommended)</td>
</tr>
<tr>
<td>emacs</td>
<td>complicated text editor</td>
</tr>
<tr>
<td>vi or vim</td>
<td>complicated text editor</td>
</tr>
</tbody>
</table>

- you cannot run graphical programs when connected to attu (yet)
- so if you want to edit documents, you need to use a text-only editor
- most advanced Unix/Linux users learn emacs or vi
- these editors are powerful but complicated and hard to learn
- we recommend the simpler nano (hotkeys are shown on screen)

Mounting remote files

<table>
<thead>
<tr>
<th>command</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sshfs</td>
<td>mount and interact with remote directories and files</td>
</tr>
</tbody>
</table>

- 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
- To mount a remote directory
  - create a local directory to mount to
  - mkdir csehomedir
  - mount your remote files on your local system

sshfs username@attu.cs.washington.edu:/homes/iws/username csehomedir/

Aliases

<table>
<thead>
<tr>
<th>command</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alias</td>
<td>assigns a pseudonym to a command</td>
</tr>
</tbody>
</table>

- alias name=command
  - must wrap the command in quotes if it contains spaces
- 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.