
CSE 391

Lecture 3

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

slides created by Marty Stepp, modified by Jessica Miller and Ruth Anderson

<http://www.cs.washington.edu/391/>

Lecture summary

- A bit more on combining commands
- Processes and basic process management
- Connecting to remote servers (attu)
 - multi-user environments
- Text 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 movie titles are sorted.
- 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

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``

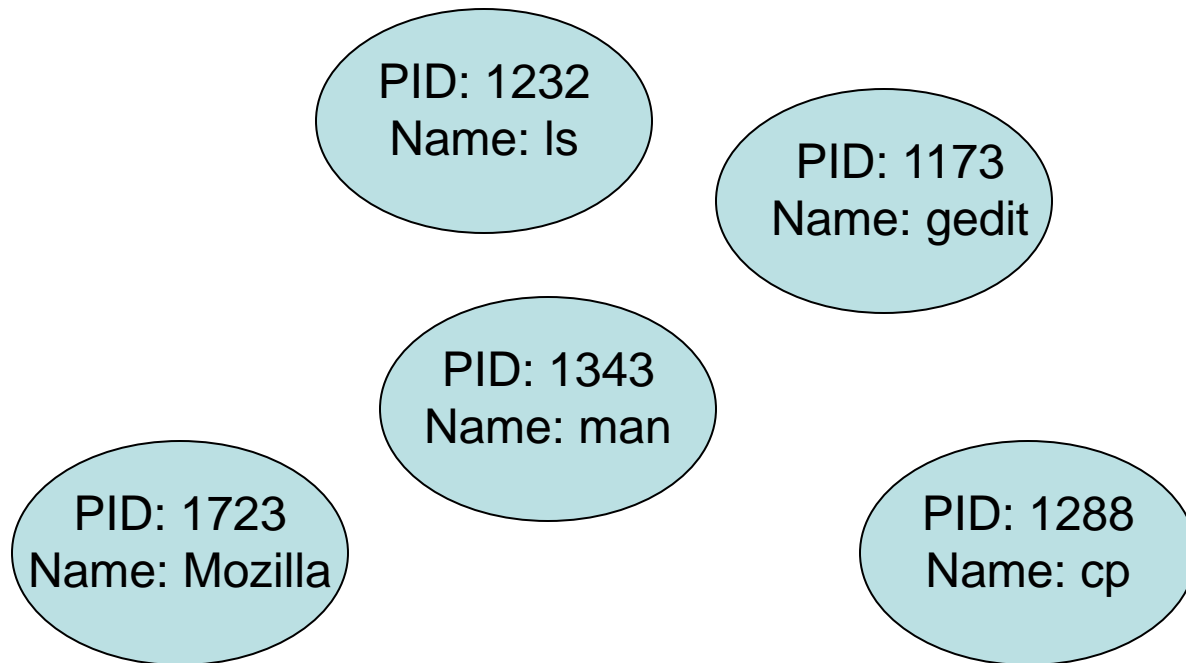
xargs

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`

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



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

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

- 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...

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

The CSE attu server

- `attu` : The UW CSE department's shared Linux server (only available to CSE majors)
- 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)

Students with EE accounts may [use their departmental servers](#)

All UW students may use [UW Linux servers](#). ([how to activate your account](#))

- Note: There are several computers that respond as `attu` (to spread load), so if you want to be on the same machine as your friend, you may need to connect to `attu2`, `attu3`, etc.

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.

Network commands

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

Text editors

command	description
<code>pico</code> or <code>nano</code>	simple editors
<code>emacs</code>	More advanced text editor
<code>vi</code> or <code>vim</code>	More advanced text editor

- 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`**
 - I would recommend you try to pick up the basics of one of these.
 - Your choice!

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`.

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

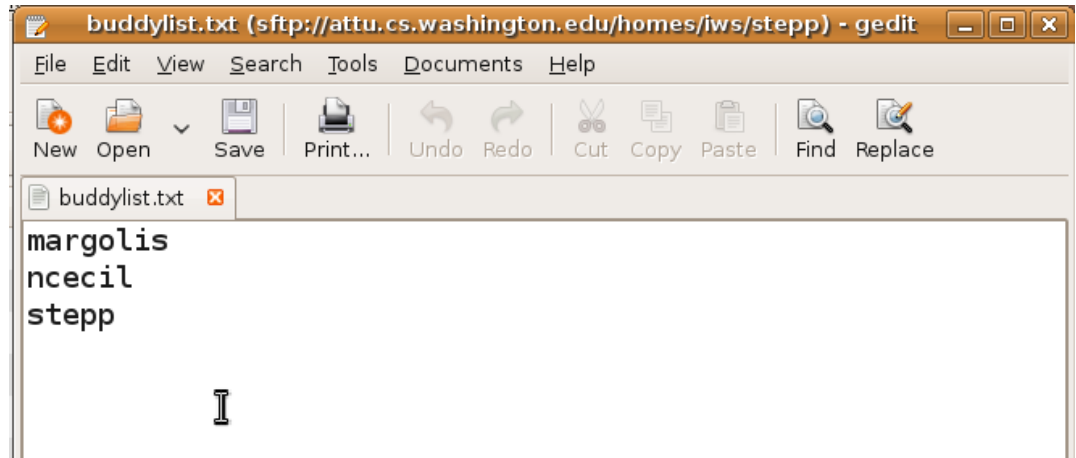
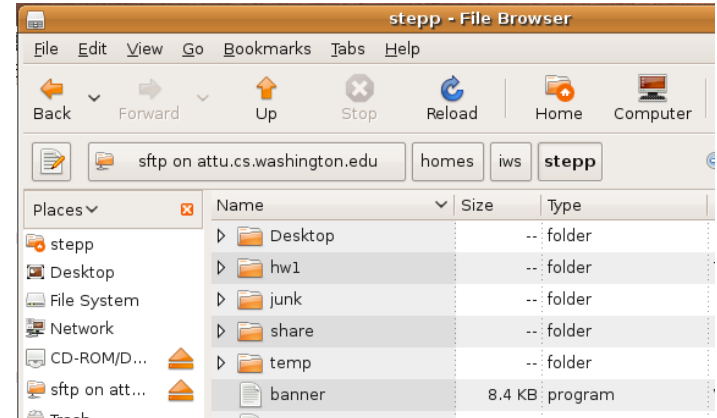
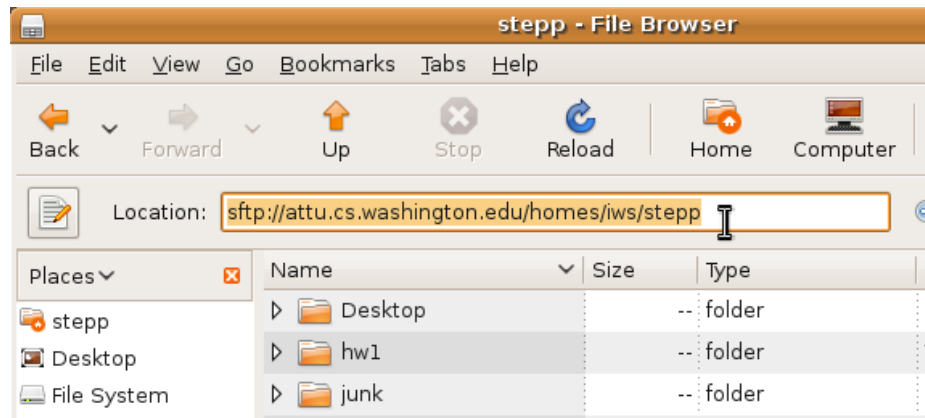
Mounting cse homedir on VM

<https://www.cs.washington.edu/lab/software/homeVMs/linuxVM#install>

- Create a directory in your home directory, called csehomedir:
 - `cd`
 - `mkdir csehomedir`
- Now to use that directory as a “link” to your CSE files on your VM:
 - `sshfs username@attu: ~/csehomedir` **OR**
 - `sshfs username@attu.cs.washington.edu:/homes/iws/username ~/csehomedir/`
- It is a good idea to back up your files from your VM regularly.
 - Actually keep your files on your CSE home directory
 - Regularly move files from your VM to another location
 - If you need to get a fresh VM image, you can save the files from your old VM using this procedure: **"My VM Seems Broken. How Do I Recover?"**
- <https://www.cs.washington.edu/lab/software/homeVMs/linuxVM#faq>

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



My VM is Broken!

<https://www.cs.washington.edu/lab/software/homeVMs/linuxVM#install>

- If your VM is misbehaving, first try a reboot of the VM and also of your machine. If that doesn't work, often it is easiest just to get a fresh VM image and start over (maybe you saved the .zip file you downloaded previously?)
- BEFORE you delete your current copy of the VM, you can save the files from your current copy of the VM using this procedure:
 - See "My VM Seems Broken. How Do I Recover?" here:
<https://www.cs.washington.edu/lab/software/homeVMs/linuxVM#faq>

Basic Emacs Commands

- C- = control key M- = meta/alt key
- read a file into Emacs: C-x C-f
- save a file back to disk: C-x C-s
- exit Emacs permanently: C-x C-c
- search forward: C-s search backward: C-r
- scroll to next screen: C-v scroll to previous screen: M-v
- Undo: C-x u

entity to move over	backward	forward
character	C-b	C-f
word	M-b	M-f
line	C-p	C-n
go to line beginning/end	C-a	C-e
go to buffer beginning/end	M-<	M->

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