

Advanced UNIX Class

The purpose of this class is to assist and familiarize the user with apple OS X

By: Eric Adint

Unix Commands

- cp copies a file
 - cp <source path>/<filename> <target path>/<filename>
- some common modifiers are
 - -r for recursive (used to copy whole directories)
 - -f force (don't ask me if I want to do it just do it)
 - -P preserve permissions

Unix Commands 007

- PS, &, and kill are used to manages jobs that are running in the background.
 - `<command> &` executes a command and sends it to the background, this allows you to continue with something else while you are waiting for it to finish
- `ps -aux` reports what processes are running in the background.
- `kill -9 <process id>` kills the backgrounded process (warning only kill process that belong to you)
- `ps` by itself only reports processes that are yours and in your current session.

Mother May I

- UNIX uses the concept of permissions and ownership to determine who can access a file or directory
- There are three types of permissions
 - read allows you to read the contents of a file
 - write allows you to write to a file or delete it
 - x or execute allows you to run a file as a script or a program.
- There are three classes of permissions.
 - User or Owner is the person who created the file
 - Group is the group that the owner belongs to
 - Other is everyone else in the world.

Unix Commands

- `chmod` changes permissions on a file
 - `chmod +x <filename>` this is commonly used to make a script executable
- Some common modifiers are
 - a (all) u (user) o (other) g (group) and r (read) w (write) x (execute)
 - these combine to form `chmod g+rw` (add read and write permissions to the group) `o-rwx` (remove read write, execute permissions from other<filename> where + means add permissions

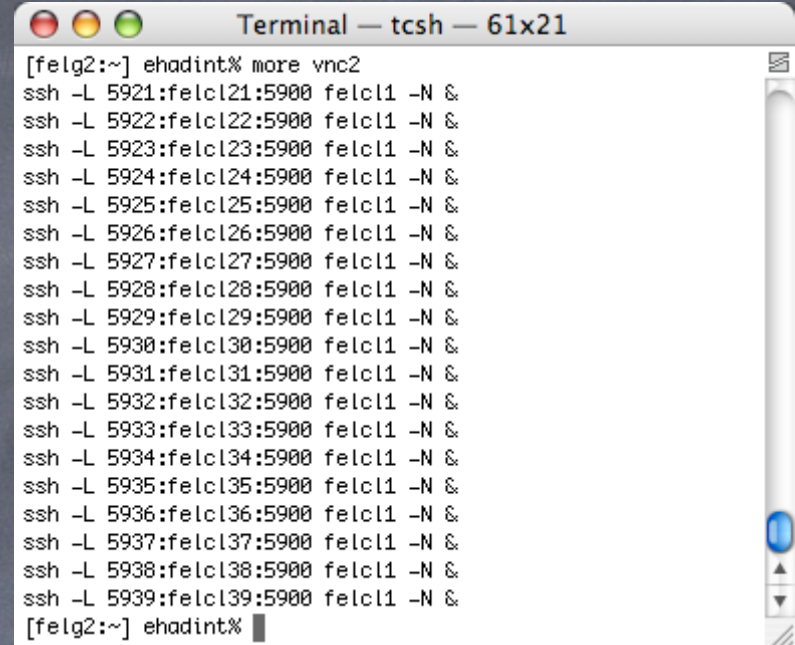
Unix Commands

- chown is used to change the ownership of a file or directory.
 - for example `chown <user>:<group> <filename>` changes the ownership of the file or directory
- some common modifiers
 - `-R` recursively changes ownership on all the elements in a directory.
 - `-f` forces the operation, without asking for permission first.

Unix Commands

- grep performs a textual search for a word or pattern.
- grep <pattern> <File or directory>
- grep test * searches all of the files in the current directory and returns any files that contain the word test
- grep uses regular expressions for searching patterns. you can find out more about regular expressions at <http://www.regular-expressions.info/>

Shell Scripting

A terminal window titled "Terminal — tcsh — 61x21" showing a list of ssh commands. The prompt is [felg2:~] ehadint%. The user has entered the command more vnc2, which has displayed a list of 19 ssh -L commands. Each command is of the form ssh -L :felcl:5900 felcl1 -N &. The list of ports ranges from 5921 to 5939. The terminal ends with the prompt [felg2:~] ehadint% and a cursor.

```
[felg2:~] ehadint% more vnc2
ssh -L 5921:felcl21:5900 felcl1 -N &
ssh -L 5922:felcl22:5900 felcl1 -N &
ssh -L 5923:felcl23:5900 felcl1 -N &
ssh -L 5924:felcl24:5900 felcl1 -N &
ssh -L 5925:felcl25:5900 felcl1 -N &
ssh -L 5926:felcl26:5900 felcl1 -N &
ssh -L 5927:felcl27:5900 felcl1 -N &
ssh -L 5928:felcl28:5900 felcl1 -N &
ssh -L 5929:felcl29:5900 felcl1 -N &
ssh -L 5930:felcl30:5900 felcl1 -N &
ssh -L 5931:felcl31:5900 felcl1 -N &
ssh -L 5932:felcl32:5900 felcl1 -N &
ssh -L 5933:felcl33:5900 felcl1 -N &
ssh -L 5934:felcl34:5900 felcl1 -N &
ssh -L 5935:felcl35:5900 felcl1 -N &
ssh -L 5936:felcl36:5900 felcl1 -N &
ssh -L 5937:felcl37:5900 felcl1 -N &
ssh -L 5938:felcl38:5900 felcl1 -N &
ssh -L 5939:felcl39:5900 felcl1 -N &
[felg2:~] ehadint%
```

- If you find yourself entering a series of commands frequently UNIX supports shell scripting.
- shell scripting is a simple text file with a list of commands as you would enter them in a terminal.
- shell scripting also supports input variables, loops and conditional operation

Shell Example

Here is an example of a script used to start the renderfarm

```
Terminal — ssh — 121x23
[felcl1:~] ehadint% more snerve
#!/bin/tcsh
set num=$2
set end=$3
set snum = 1
while ( $num )
    @ snum = $num - 2
    echo -n "felcl${snum}:"
if (${1} == "shutdown") then;
    echo ssh felcl${num} sudo killall LWSN-${snum}<pss
    ssh felcl${num} sudo killall LWSN-${snum}<pss
endif
if (${1} == "startup") then
    echo ssh felcl${num} sudo -u administrator open -a """/Volumes/BHD1TB/LightWave [8]/programs/LWSN-${snum}""'<pss
    ssh felcl${num} sudo -u administrator open -a """/Volumes/BHD1TB/LightWave [8]/programs/LWSN-${snum}""'<pss
endif
    @ num++
    if ($num >$end) then
        exit (0)
    endif
end
[felcl1:~] ehadint%
```

Programming and Editing

- Creating a folder for programs
- Editing a program with Textedit
- Editing a program with Textwrangler
- Helloworld.c Example

Compiling And running a program

- Opening a terminal and navigating to the programming directory
- Compiling HelloWorld.c
- Running HelloWorld
 - . errors.
- Workflow.

Redirection Examples.

- Creating a program with redirection
- Printit.c
 - Typing input
 - redirecting input.
 - Redirecting output
 - Redirecting input and output.

VI

(return of the text editor)

- VI is the oldest and most common UNIX text editor in existence it has support for advanced editing and pattern matching search and replace.
- VI has three modes view, insert, and command.
 - view allows you to view files without modifying the text.
 - insert allows you to edit the file.
 - command allows you to save quit search replace and all other text editor functions.
- vi is very efficient at performing complex text editing and manipulation operations.
- you have an overwhelming desire to use vi (announcer waives his hand)

VI Basics (View)

- vi <filename> to open a file for editing
- ←↑→↓ move the cursor left up right and down. also j,k,h,l accomplish the same thing
- x or delete delete the letter over the cursor
- dd deletes a line
- dw deletes the word
- yy yanks a line into the buffer, you can also use mouse select and ⌘c on the mac
- pp paste the line at the cursor, and ⌘v on the mac
- /<pattern> performs a regular expression search for a certain word ie /test↵ will place the cursor at the first instance of the word
- cw allows you to change the word under the cursor (press esc when done)
- r allows you to change the character under the cursor.

VI Basics (Insert)

- i,I puts vi into insert (edit) mode at/before the cursor
- o, O puts vi into insert mode above/below the current line (it creates a new line)
- a,A inserts text at the beginning/ending of the current line
- to exit insert mode hit the esc key.

VI Basics (Command)

- to enter command mode use `:` in view mode or `esc` : in insert mode (I will use `:<command>` notation to signify command mode)
- Basic Command options
 - `:q` Quit vi, `:w` save the current file, `:qw` save the file and quit vi, `:q!` force quite vi without writing changes
 - `:/<pattern>` find selected pattern in file
 - `:^` go to the beginning of the file
 - `:$` go to the end of the current file.
 - `:<number>` got to the line number in the file.

VI Basics (Command)

- VI also has advanced search and replace capabilities.
 - `:<start>,<end>s/<pattern>/<replace pattern>/<gis>`
 - `<start>` (optional) starting line to search
 - `<end>` (optional) ending line to search.
 - `s/` tells vi you want to search and replace
 - `<pattern>` this is the regular expression pattern that you want to search for
 - `<replace pattern>` this is the text you wish to replace any matches with.
 - `<gis>` (any combination, optional) search modifiers `g` (global, replaces all instances of a match) `i` (Case insensitive) `s` ignore carriage return