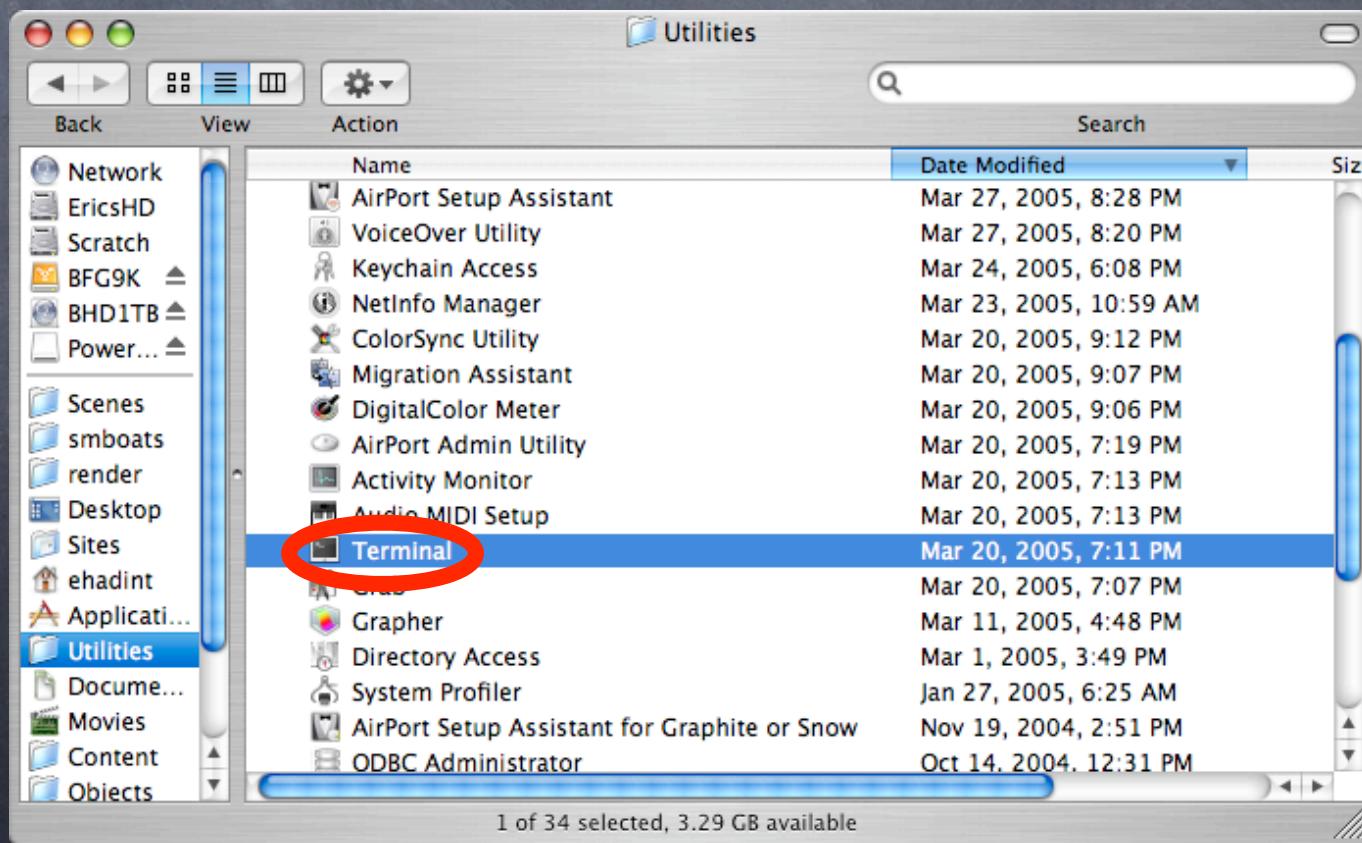


UNIX FUNDAMENTALS

THIS CLASS IS DESIGNED TO GIVE THE USER A
FUNDAMENTAL OVERVIEW OF THE UNIX
SUBSYSTEM IN OS X

OPENING A TERMINAL

- To access the UNIX subsystems in OS X you need to open the terminal application. It is located in the Utilities folder under applications. It is highly recommended that you put this application on the dock



NAVIGATING IN THE TERMINAL

- The ← → keys move the cursor left and right
- the ↑ ↓ keys page through the command history.
- you can use the mouse to select text and ⌘c and ⌘v to cut and paste text
- Delete deletes text behind the cursor the ⌘ key deletes text after the cursor.

Common Navigation Nomenclature

- `.` means this directory or right here
- `..` means the directory right above you
- `~` means your home directory
- `/` the root directory
- `~<username>` the home directory of the user
- for example `cp /local/fel/test .` will copy test to your current directory

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

- mv moves a file from one directory to another. it can also be used for renaming files
- mv <source dir>/<filename> <target dir>/<file>
- 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

- `cd` changes your current working directory.
 - `cd <directory to change to>`
- also the command `pwd` tells you which directory you are currently in.

Unix Commands

- mkdir creates a new directory
- mkdir <directory name> or /<ABS path>/
<directory name>

Unix Commands

- ls lists the contents of a directory
 - ls or ls <directory name> lists the current directory or the specified directory
- Some common modifiers.
 - -l lists all the information about the files
 - -lt lists all information in chronological order
 - -al list all the information including hidden files.

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/>

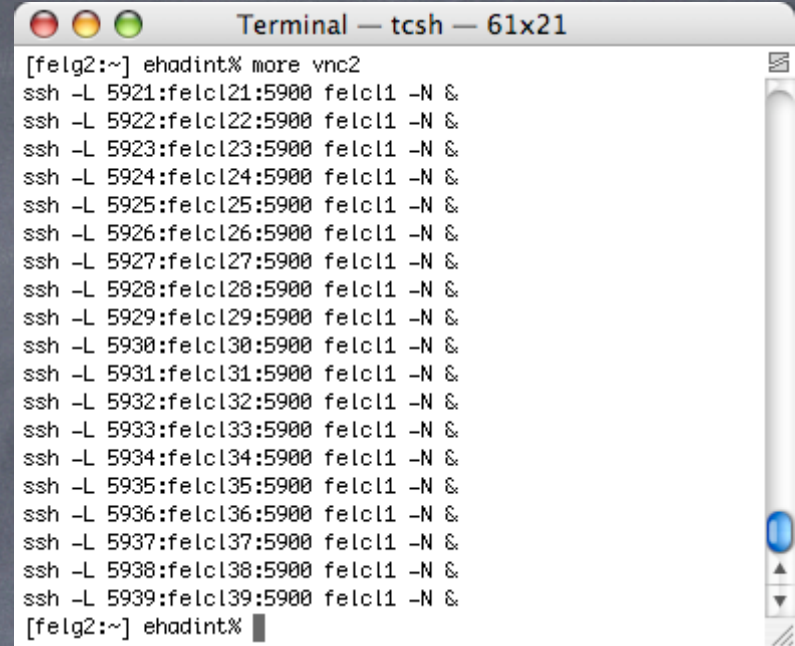
UNIX REDIRECTION

- UNIX allows to you information form many sources.
 - `<` `>` input output redirection
 - `{command} < {filename}` takes the information in filename and uses it as input to the `{command}`
 - `{command} > {filename}` takes the output from the command and puts it into the filename
 - `>` creates the file
 - `>>` appends to the file
 - for example `ls -al > dirinfo.txt` creates the file `dirinfo.txt` and puts the directory information in it.
- | or pipe redirection
 - `{command1} | {command2}` takes the output of `command1` and makes it the input of `command2`
 - for example `ls -al | grep <somefile>` will only return something if the file is in the output.

Whose da MAN and --h

- UNIX has a built in help system or collection of manual pages that can be accessed by the command `man`
 - `man <command name>` will give you a complete manual page on the command.
- also for a brief summary of a commands usage `<command name> --h` or `-help`

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%
```

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

Compiling a program

- you may need to write or modify a program
gcc is the standard unix c compiler
- gcc < -lm (or other libraries)> <filename> -o
<executable to create>
- this will create an executable from c source code.

UNIX SSH

- ssh is a program used for creating terminals on remote computers. it stands for secure shell, its predecessor telnet is still available but should not be used because all data sent by telnet is unencrypted.
- ssh -l <username> <target computer>
 - <username> (optional) if ssh does not recognize your local username you may have to use this to tell it who you are.
 - <target computer> this is the computer you wish to connect to
- Example ssh -l ehadint felcl1.physics.nps.navy.mil

X11

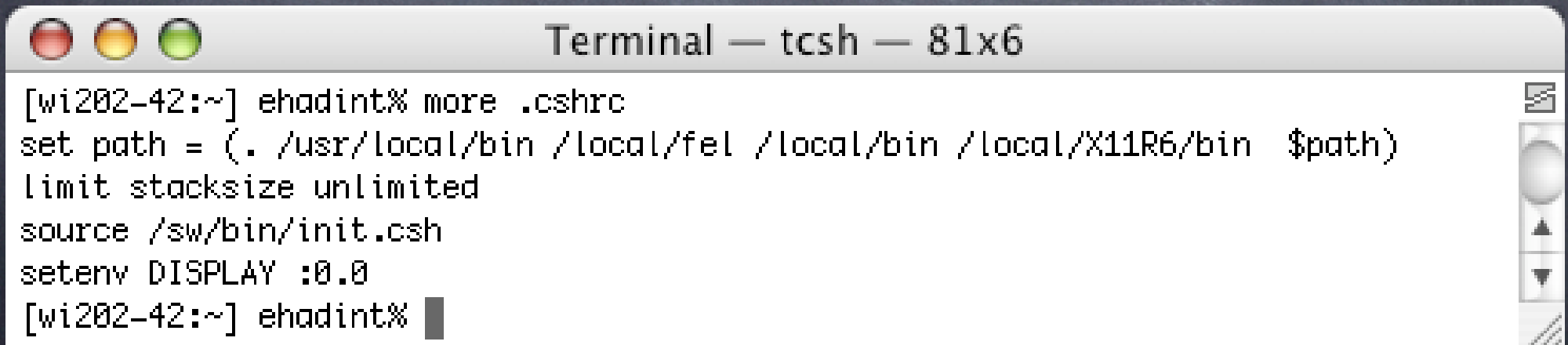
- X11 is a traditional UNIX windowing application it is a GUI based environment. X11 can be run locally or remotely
- X11 is located in the utilities folder, it is highly recommended that you add this to your dock for quick access.
- There are two settings that you need in order to use X11 from a terminal the DISPLAY environment variable, and xhosts needs to have proper permissions.
 - in tcsh you need to type `setenv DISPLAY <your computer ip>:0.0`, or if you are using X11 on localhost you can set it to `setenv DISPLAY :0.0`
 - in bash you need to type `export DISPLAY=:0.0` or your IP address for remote operations.
 - next if you have permission errors you need to type `/usr/X11R6/bin/xhost +` to enable connection to the X11 server (see example)
 - this can be configured in the profile script covered later.
- a good test of a properly configured X11 system is typing in `/usr/X11R6/bin/xeyes`

SW and fink

- many x11 applications and utilities are installed separately from OS X, these consist of gv (ghost view), imagemagick (image conversion and manipulation suite) these applications are located under the /sw/bin directory
- There are many powerful UNIX applications available through fink (Show fink commander demo)
- gv or ghostview is the primary application used for viewing FEL simulation data graphically

Profiles

- in your home directory there is a file named `.cshrc` (tcsh shell) and `.bashrc` (bash shell) this file is responsible for setting up your initial environment
- This is a typical `.cshrc` file

A terminal window titled "Terminal — tcsh — 81x6" with three colored window control buttons (red, yellow, green) on the left. The terminal shows the command `more .cshrc` and its output:

```
[wi202-42:~] ehadint% more .cshrc
set path = (. /usr/local/bin /local/fel /local/bin /local/X11R6/bin $path)
limit stacksize unlimited
source /sw/bin/init.csh
setenv DISPLAY :0.0
[wi202-42:~] ehadint% █
```

On the right side of the terminal window, there are standard OS window controls: a close button, a scroll bar, and a zoom button.

LAM maintenance

- lamclean reinitializes the user environment and ensures that there are no environment variables from a previous session, it is recommended that you run this before running any mpi program
- mpitask lets you know which nodes are performing mpi functions.
- lamhalt shuts down the cluster

mpirun

- mpirun starts the mpi program and manages all output and tasks
- mpirun -np <numprocs> <program>
 - <numprocs> is the number of processors to run the mpi task on

Connecting to the cluster file server

- you may find it easier to edit and manage text and output files on your local computer. to do this you can connect to the cluster file services and work on your files locally with a gui or in a local terminal
- (show file connecting demonstration)