Introduction to Linux
HPC Workshop

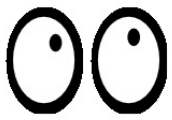
Erin Shaw and Cesar Sul

Advanced Cyberinfrastructure Research and Education Facilitation

USC Center for High-Performance Computing

Welcome to HPC

- Log in if you have not done so
- Add –X argument to ssh to enable remote viewing
  - X-Win32 for PC is pre-configured
  - Mac users type ssh –X hpc-login2.usc.edu



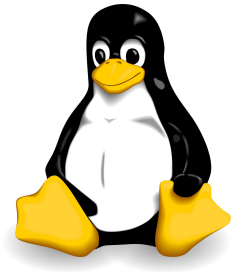
Then type "xeyes" at the command line.

- Interactive “eyes” appear if remote viewing is enabled
  - *The xeyes program is running on HPC and displayed on your PC*
- Move your cursor to interact with the program
  - *Ctrl-C in the xterm to end*

Introduction

- Show New User slides


Linux

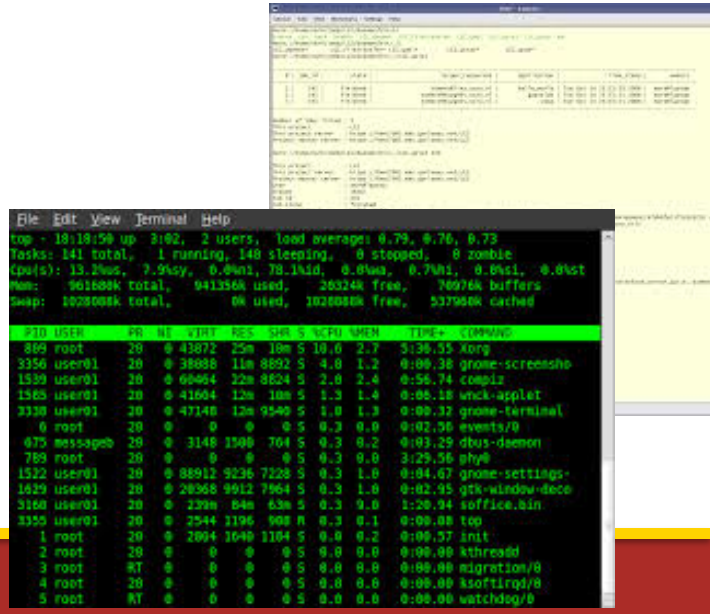


What is Linux?

- Linux is an Operating System (OS)
  - Software that enables computer devices and applications to perform tasks
- Linux is an open OS (unlike &)
  - Created in 1991 as a variant of Unix for PCs
    - *Thousands of developers contribute*
    - *www.linuxfoundation.org*
  - Most web-services today are run under Linux

GUIs and CMIs

- GUI: Graphical User Interfaces
  - Linux has these, too
  - But HPC does not support
- CMI: Command Line Interfaces/Interpreter
  - Called a ‘shell’ in Unix/Linux
  - The only way to communicate with HPC nodes
  - PCs & Macs also have CMIs
    - *Accessed via terminals/xterms on PCs (DOS) & Macs (MacOSX)*



What is a Unix/Linux Shell?

- A command interpreter and processor
- HPC supports two shells
  - bash and tcsh ("t" shell)
    - *Bash is default shell on HPC nodes, considered more functional*
    - *Bash = Bourne-again shell (sh), developed by Bourne in 1977*
- To verify and change your shell

```
$ echo $0      #show shell
-bash
$ tcsh         #change shell to
               tcsh
$ echo$0
tcsh
$ bash
```


See http://www.faqs.org/faqs/unix-faq/shell/shell-differences/ for a history and discussion of Unix shells.

Comparison of Shells

Function	sh	csh	ksh	bash	tcsh	zsh
Job control	N	Y	Y	Y	Y	Y
Aliases	N	Y	Y	Y	Y	Y
Shell functions	Y	N	Y	Y	N	Y
I/O redirection	Y	N	Y	Y	N	Y
Directory stack	N	Y	Y	Y	Y	Y
Command history	N	Y	Y	Y	Y	Y
Command line editing	N	N	Y	Y	Y	Y
Vi Command line editing	N	N	Y	Y	Y	Y
Emacs Cmd line editing	N	N	Y	Y	Y	Y
Rebind Cmd line editing	N	N	N	Y	Y	Y
User name look up	N	Y	Y	Y	Y	Y
Login/Logout watching	N	N	N	N	Y	Y
Filename completion	N	Y	Y	Y	Y	Y
Username completion	N	Y	Y	Y	Y	Y
Hostname completion	N	Y	Y	Y	Y	Y
History completion	N	N	N	Y	Y	Y
Full Completion	N	N	N	N	Y	Y
Mh Mailbox completion	N	N	N	N	N	N
Co Processes	N	N	Y	N	N	Y
Builtin artithmetic eval	N	Y	Y	Y	Y	Y
Can follow sym links	N	N	Y	Y	Y	Y
Periodic cmd execution	N	N	N	N	Y	Y

Function	sh	csh	ksh	bash	tcsh	zsh
Custom Prompt (easily)	N	N	Y	Y	Y	Y
Sun Keyboard Hack	N	N	N	N	N	Y
Spelling Correction	N	N	N	N	Y	Y
Process Substitution	N	N	N	Y	N	Y
Underlying Syntax	sh	csh	sh	sh	csh	sh
Freely Available	N	N	N	Y	Y	Y
Checks Mailbox	N	Y	Y	Y	Y	Y
Tty Sanity Checking	N	N	N	N	Y	Y
Can cope w large arg lists	Y	N	Y	Y	Y	Y
Has non-interac. startup file	N	Y	Y	Y	Y	Y
Has non-login startup file	N	Y	Y	Y	Y	Y
Can avoid user startup files	N	Y	N	Y	N	Y
Can specify startup file	N	N	Y	Y	N	N
Low level cmd redefinition	N	N	N	N	N	N
Has anonymous functions	N	N	N	N	N	N
List Variables	N	Y	Y	N	Y	Y
Full signal trap handling	Y	N	Y	Y	N	Y
File no clobber ability	N	Y	Y	Y	Y	Y
Local variables	N	N	Y	Y	N	Y
Lexically scoped variables	N	N	N	N	N	N
Exceptions	N	N	N	N	N	N

Environment Variables

- Variables – a name and a value – defined within the Linux environment
  - Set and used by shells, users, scripts, programs
  - Names uppercase by convention, referenced with a "$"
- Display all environment variables

```
$ env
HOSTNAME=indigo.usc.edu
TERM=xterm
SHELL=/bin/bash
USER=avalonjo
MAIL=/var/spool/mail/avalonjo
PATH=/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin
PWD=/home/avalonjo/workshop
```

Environment Variables

- Example: $PATH holds a user's search path

```
$ echo $PATH
/usr/bin:/usr/local/bin:/bin:/sbin
```

  - *When you type a command, bash looks searches for it in the directories listed: First in /usr/bin, then in /usr/local/bin, etc.*

- Set environment variables

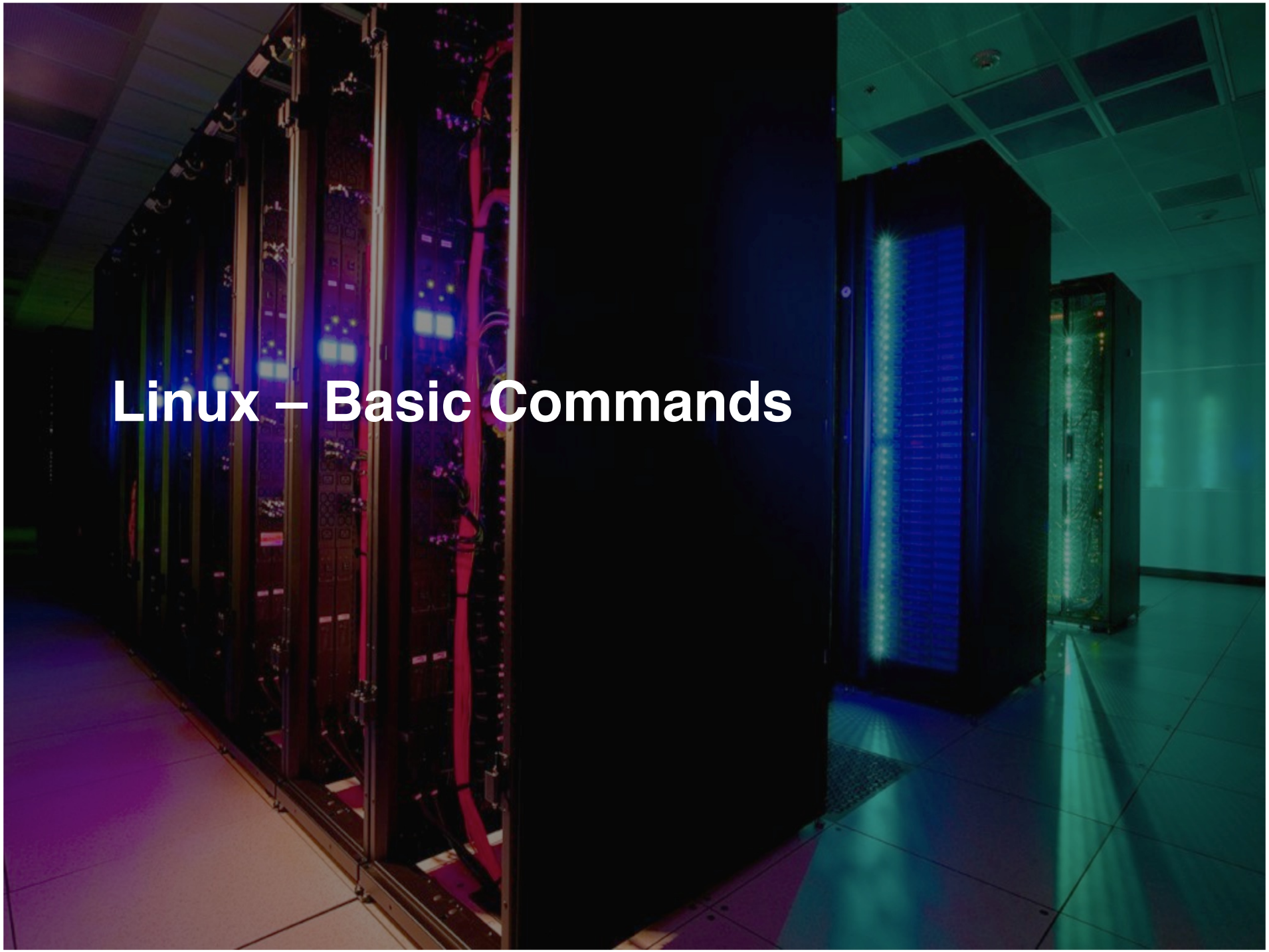
```
$ JAVA_HOME='/usr/usc/java/default'
```

  - *Many programs set (or require that you set) a 'home' variable*

```
$ export /usc/usc/matlab/default:${PATH}
```

  - *Add MATLAB directory to path (gets searched before other directories)*

```
$ echo $PATH
```


Linux – Basic Commands

Basic Linux Commands

ls	List file and/or directory names
pwd	Print working (current) directory

```
$ touch a.a
$ touch b.b
$ ls
a.a   b.b
$ ls –lt        #list -long form, -sort by time
total 0
-rw-rw-r--. 1 avalonjo  avalonjo   0 Mar 24  13:44 b.b
-rw-rw-r--. 1 avalonjo  avalonjo   0 Mar 24  13:44 a.a
$ pwd
/home/avalonjo/workshop
```

Basic Linux Commands

cd	Change directory
mkdir/rmdir	Print working (current) directory

```
$ ls
a.a   b.b
$ mkdir dir1
$ ls
a.a   b.b   dir1
$ cd dir1
$ pwd
/home/avalonjo/workshop/dir1
```

```
$ cd ..
$ rmdir dir1
$ ls
a.a b.b
```

Other Special Characters

- ~ home directory
- . current directory
- .. parent directory
- * any number of any character (wild card)

```
$ cd ~                  #go home
$ pwd
/home/avalonjo
$ ls .bash*             #show bash
                         dot files
```

```
$ cd ~                  #go home
$ pwd
/home/avalonjo
$ cd ..                  #go up
$ pwd
/home
$ cd .                  #go current
$ pwd
/home
$ cd ~
$ pwd                   #go home
/home/avalonjo
```

Basic Linux Commands

cp/mv/rm	Copy/move/remove file or directory
alias	Create an alias for a command

```
$ ls
a.a  b.b

$ cp b.b c.c
$ ls
a.a  b.b  c.c

$ mv c.c c.new
$ ls
a.a  b.b  c.new
```

Basic Linux Commands

cp/mv/rm	Copy/move/remove file or directory
alias	Create an alias for a command

```
$ ls
a.a    b.b    c.new
$ rm a.a
$ ls
b.b    c.new

$ alias rm 'rm –i'
$ rm b.b
rm: remove regular empty file `b.b'? n
```

```
$ mkdir dir2
$ ls
b.b   c.new   dir2

$ alias ls 'ls -CF'
$ ls
b.b   c.new   dir2/
```

Basic Linux Commands

cat/more/less* Display file contents

> Redirect output from command line to file

```
$ ls
b.b  c.new dir2/

$ ls > ls.out
$ cat ls.out
b.b  c.new  ls.out  dir2/

$ less ls.out
```

*Less is a program similar to more (1), but which allows backward movement in the file as well as forward movement. Also, less does not have to read the entire input file before starting, so with large input files it starts up faster…

Summary

ls	List file and/or directory names
pwd	Print working (current) directory
mkdir/rmdir	Create/remove directory
cd cp/mv/rm	Change directory
cat/more/less	Copy/move/remove file or directory
man	Display file contents
	Display online manual
chmod/chown	Change permission/ownership

Command History***

history Display a history* of your commands

↑ ↓ Previous command, next command, in order

!# Rerun a command

```
$ history
…
32 mkdir dir2
33 ls
34 alias ls 'ls -CF'
35 ls

$!35      #re-run the 35th command
```

*Shell keeps a log of your commands -- it's always a good idea to review this if you forget what you typed

Command-line Completion***

<tab> Auto-complete the cmd/file/dir name

<tab><tab> Show candidates for the cmd/file/dir name

It can substantially reduce the number of keystrokes

```
$ ls /usr/usc/mat      <tab> to autocomplete (fails)
                       <tab> to show candidates
$ ls /usr/usc/matl     <tab> to autocomplete (succeeds)
                       <tab><tab> to show candidates
$ ls /usr/usc/matlab/2     <tab> etc.
```

Linux Commands & Concepts

Environment

bash shell, .bashrc

environment variables ($PATH)

Keyboard navigation

<up>/<down> : show prev/next cmd

<ctl-a>/<ctl-e> : mv to beg/end of line

<alt-f>/<alt-b> : mv forwd/back a word

<tab>, <tab-tab>: autocomplete

Special characters

"*": wild card

"/", "~" : root dir, home dir

".", ".." : current dir, parent dir

">", "<" : redirect output/input

"|" : pipe output to input of next cmd

Navigation

$ pwd

$ cd

$ ls (-alh)

$ cp/mv

$ touch/ rm (-i)

$ mkdir/rmdir

$ history

Reading/Editing files

$ cat, $ less

$ nano ($vi, $emacs)

Permissions

$ chmod

$ chgrp

Information

$ mybalance -h

$ myquota

$ top

$ du –h

$ man

$ echo [$PATH]

$ wc

$ sort

Tools

$ alias

$ wget

$ for (loop)

Linux Commands & Concepts (applied)

$ mybalance -h
$ myquota
$ top
$ pwd
$ ls, ls -l, ls -F --color
$ man ls
$ echo hello
$ ↑↑↑↓↓↓
$ echo $PATH
$ cd .., pwd, ls

$ cd /home/rcf-proj/<myproj>/<mydir>
$ mkdir test, ls, rmdir test, ls
$ mkdir workshop, ls
$ cd workshop, pwd
$ touch a.a, ls
$ cp a.a b.b, ls
$ mv b.b c.c, ls
$ rm c.c, ls
$ alias rm='rm -i'
$ rm a.a, ls

Linux Commands & Concepts (applied)

$ alias cdp='cd /path/to/proj'
$ cd ~, pwd
$ cdp, pwd
$ cd ~, ls –alh .bash*
$ cat .bashrc
$ cp .bashrc .bashrc_ori
$ nano ~/.bashrc

alias rm='rm-i'

alias cdp='cd /path/to/proj'

alias ll='ls -hlt'

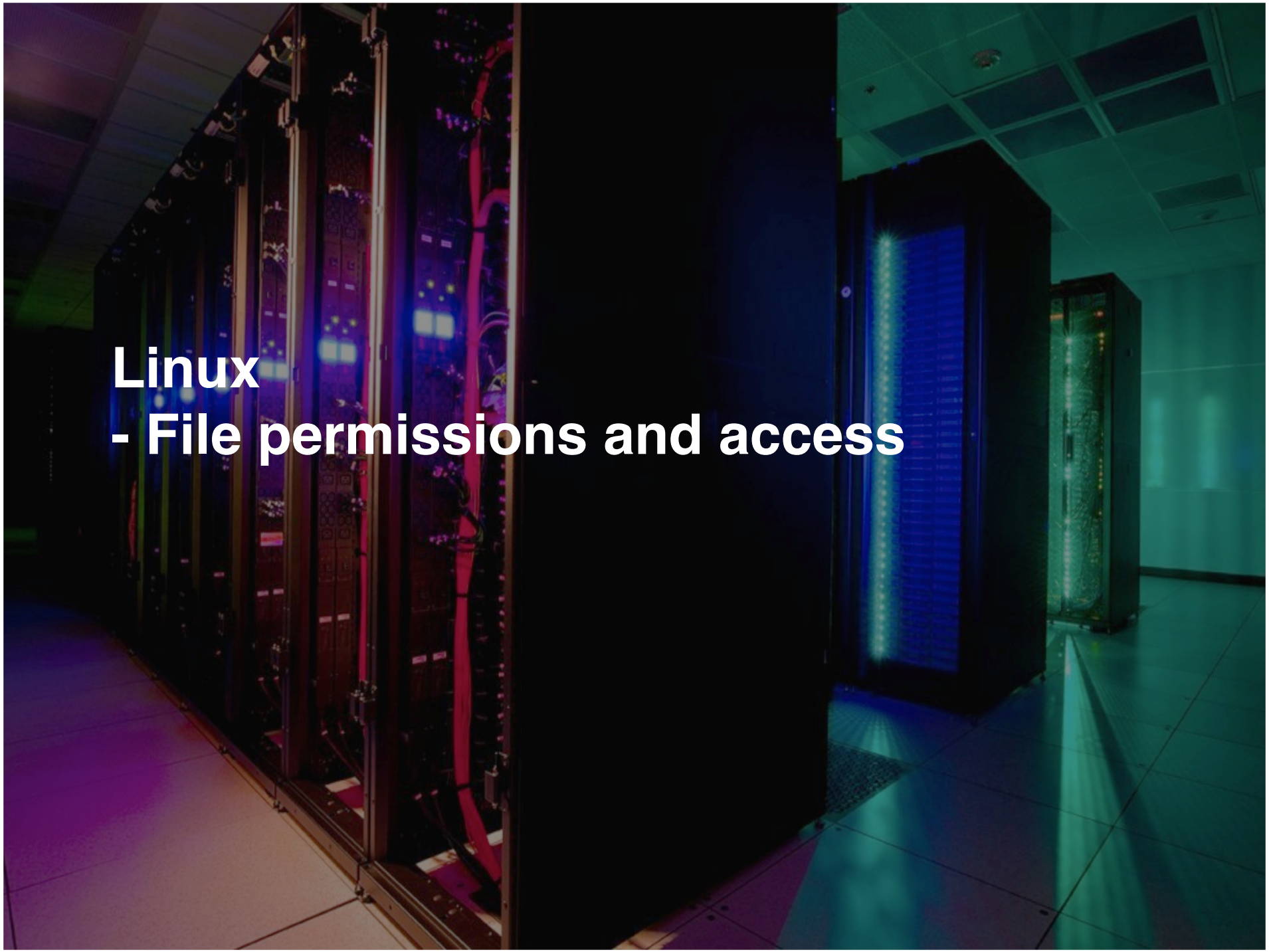
$ cdp
$ ll workshop
$ chmod g+w workshop, ls -l
$ chmod g-rw workshop, ls –l
$ cd workshop
$ wget http://hpcc.usc.edu, ls
$ wc -l index.html
$ cat index.html | grep @
$ cat index.html | grep jpg

Linux Commands & Concepts (applied)

```
$ history
$ history >> history.out
$ less history.out
$ less history.out | grep wget
$ ls -t /usr/usc/ > ls.out
$ less ls.out
$ less ls.out | sort
$ less ls.out | sort -f | head –n 5
$ du –h * | sort –n
$ du –h --summarize
```

```
$ ls /usr/usc/mat
  <tab> to autocomplete (fails)
  <tab><tab> to show candidates
$ ls /usr/usc/matl
  <tab> to autocomplete (succeeds)
  <tab><tab> to show candidates
$ ls /usr/usc/matlab/2
    :

$ ls *
$ for i in *; do echo $i; done
```


Linux
- File permissions and access

File Ownership and Permissions

- Permissions are associated w/ every file & directory
  - $ls –lg # list files (-l long) and groups (-g)
    # This will display permissions
- Permissions – three types
  - read, write, execute
  - **r**=read, **w**=write, **x**=execute
- Ownership – three types
  - owner, group, others
  - **u**=user (owner), **g**=group, **o**=others (also **a**=all)

File Ownership and Permissions

chmod file Change file permission

chmod a+w file Add write permissions for all users

chmod 750 file Set rwx permission for user, r-x permission for group and no permission for others

rwx can each be set to 0 or 1,

so range is 000-111 (binary) and 0-7 (decimal)

i.e., r=100 (4), w=010 (2), x=001 (1)

111 101 000 => rwx r--x -----

7 5 0 => u g o

File Ownership and Permissions

groups — Displays the group ID of all groups you belong to. The first ID is your primary group ID. By default, all files you create will have this group ID.

chgrp <grp> mydir — Change group ID of this directory (or file)

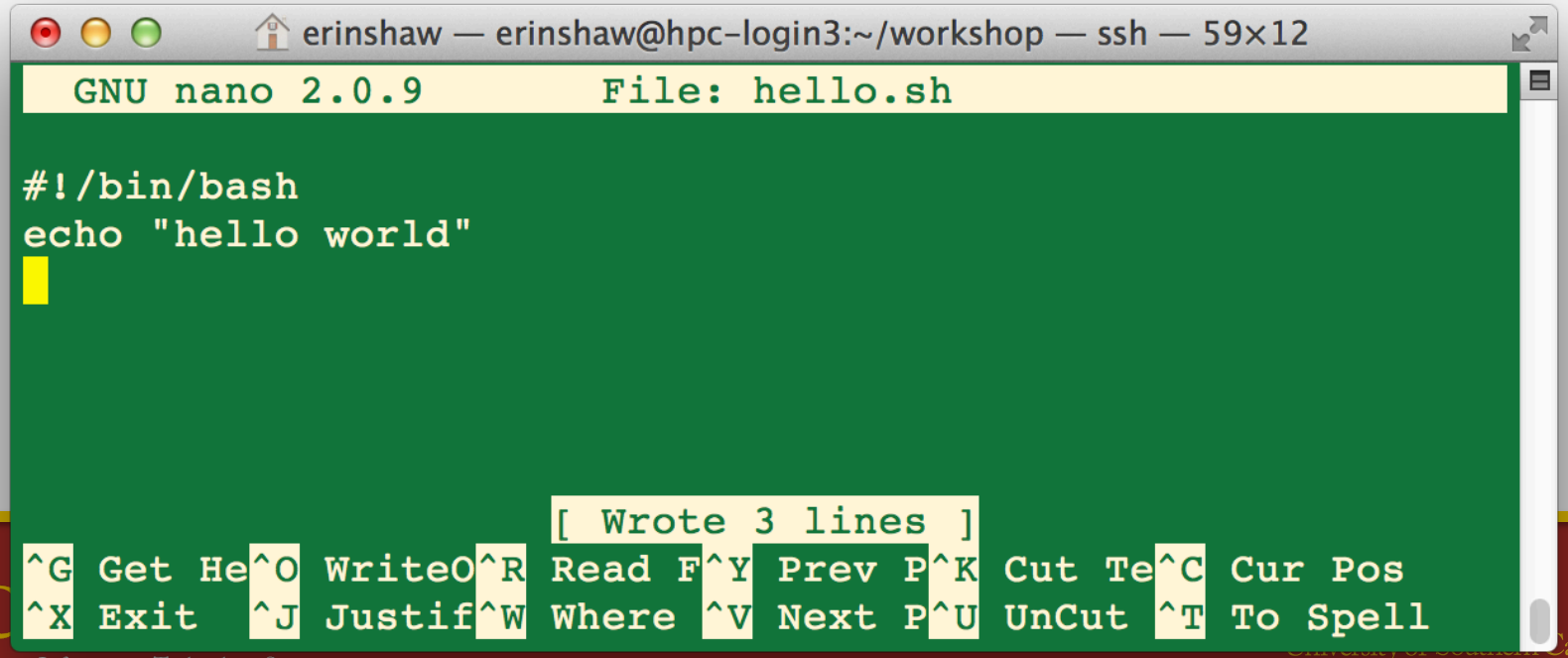
chmod g+s mydir — Set group ID (also *setgid*) of mydir so that all new files and subdirectories within mydir will inherit the group ID of the directory, rather than the primary group ID of the user who create the file.

(The exception is if you mv a file to mydir.)


Linux
– Editing & Scripting

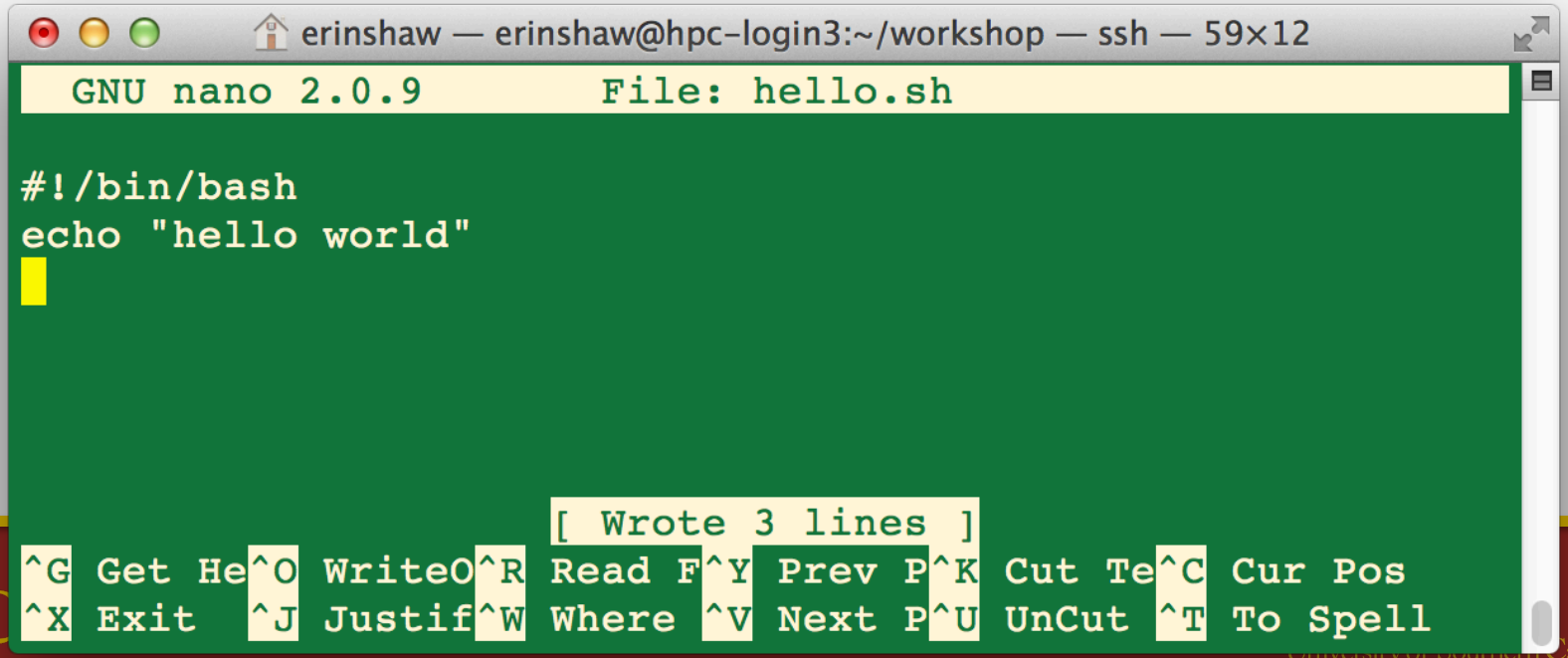
Editors

- There are several editors installed on HPC
  - "vi" is native to Unix/Linux operating systems
  - "vim", "emacs", and "nano" are also installed on HPC
- We teach nano because it is easiest to use



Editors

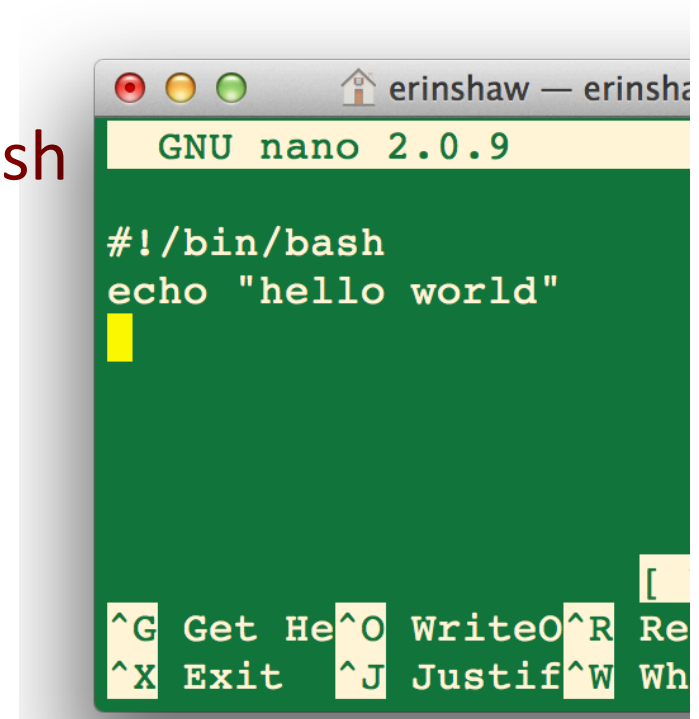
- The most useful commands are displayed along bottom
  - ^o indicates press Ctrl-o to write out (save) text
  - ^k/^u = cut/uncut, ^a/^e = move to beg/end of line
  - ^v/^y = page forward/backward, ^w = search, ^x = exit



Shell Scripting

- Now that you can edit, let's create a script!
  - cd to your workshop directory
- Create a bash script named hello.sh
  - Open nano (nano hello.sh)
  - Type in the lines ==========>
  - Save the file (^o), enter/return,

    Then exit (^x)
- Run the script!
  - Type hello.sh
  - Hmmm…. what's wrong?



Shell Scripting

$ hello.sh

-bash: hello.sh: command not found

Problem: hello.sh is not in your path

- Your current directory is not in your path by default
- Solution: Use an absolute path

$ ./hello.sh

Shell Scripting

$ ./hello.sh

-bash: ./hello.sh: Permission denied

Problem: hello.sh is a text file

- It is not an executable program
- Solution: Change mode of file to executable

  You need "permission" to execute, just like you need permission to read and write files.

$ chmod +x hello.sh

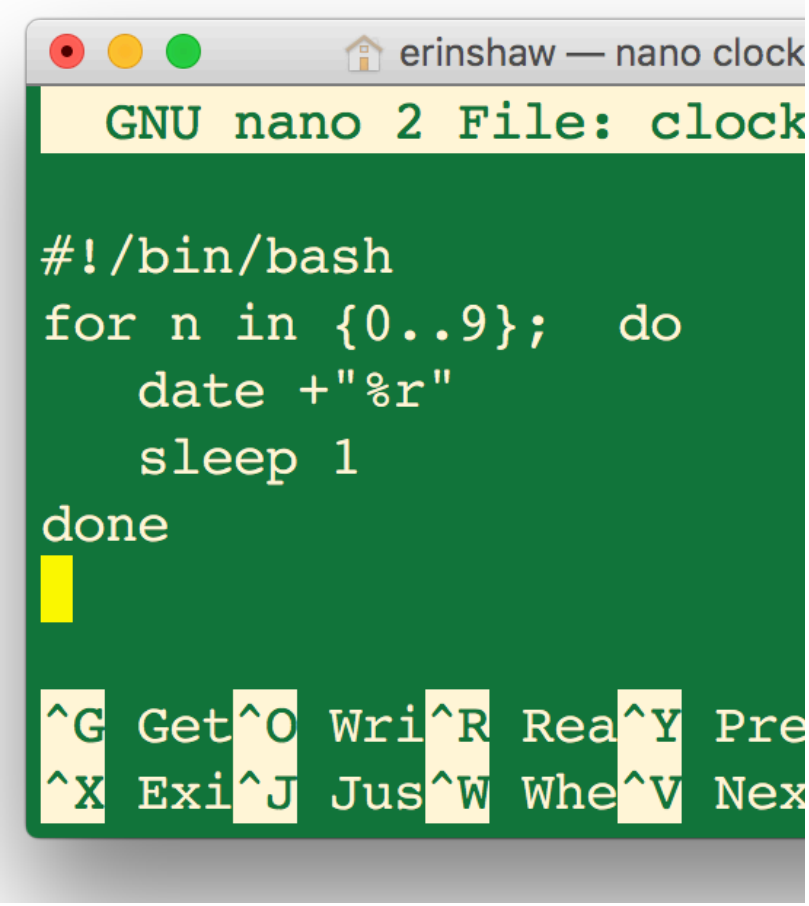
Shell Scripting

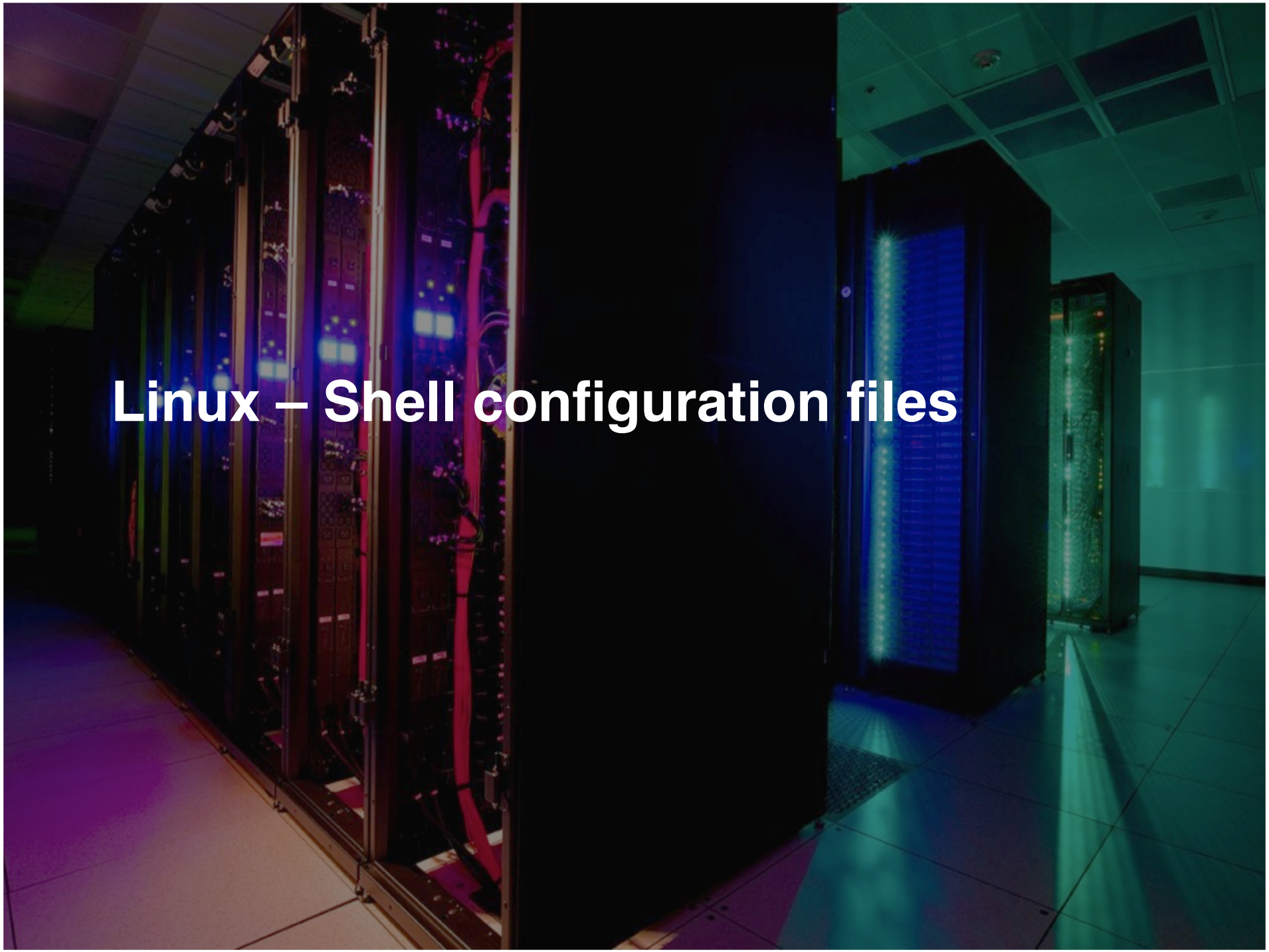
```
$ ./hello.sh
hello world
```

Bash Script #2

- Create clock.sh
  - $ nano clock.sh
  - Type in the lines
    ==========>
  - Save the file and exit:

    ^o + <enter/return> + ^x
- Run the script!
  - Add +x permission
  - Specify current directory




Linux – Shell configuration files

bash Shell Configuration Files

- Configuration files are used to set up user environments (also called *config*, *init,* and *dot* files)
  - Minimal shell configuration files come in each user's home directory by default (use "ls -a" to view)
- For bash, these are .bash_profile & .bashrc
  - .bash_profile is read by bash, hpc's default shell, when you login
  - .bash_profile sets up your hpc environment and then runs .bashrc
- Place your custom run commands, prompts, paths, aliases, and environment variables in .bashrc

tcsh Shell Configuration Files

- For tcsh, the config files are .cshrc & .login
  - .cshrc is read by tcsh, if you are using this shell, when you login
  - .cshrc sets up your hpc environment and then runs .login
- If you use tcsh, place your custom run commands, prompts, paths, aliases, and environment variables in .cshrc

Edit .bashrc script

- **Make a backup!**
  - cd ~
  - cp .bashrc .bashrc_ori
  - nano .bashrc
- Add some aliases
  - e.g., for rm, cd

    (my .bashrc file is shown)
  - Save and exit
- Test your aliases


Linux – Process managment

Process Management

- A process is a task executed by Linux
  - When you execute a command, at least one process, or job, will be created
  - Each process is assigned a unique integer called a Process ID (PID)
- Type $ top to view running jobs

```
Processes: 200 total, 2 running, 6 stuck, 192 sleeping, 922 threads                                 11:29:56
Load Avg: 1.34, 1.21, 1.03  CPU usage: 1.93% user, 4.52% sys, 93.53% idle    SharedLibs: 11M resident, 10M data, 0B linkedit.
MemRegions: 93261 total, 2125M resident, 72M private, 657M shared. PhysMem: 5745M used (1364M wired), 595M unused.
VM: 489G vsize, 1066M framework vsize, 174811(0) swapins, 355954(0) swapouts.
Networks: packets: 38059781/27G in, 20628194/7089M out. Disks: 10560617/208G read, 14170246/446G written.

PID    COMMAND      %CPU TIME     #TH  #WQ  #PORT #MREGS MEM    RPRVT  PURG   CMPRS  VPRVT  VSIZE  PGRP  PPID  STATE    UID
97999  syspolicyd   0.0  00:00.06 2    1    23    40     2824K  2640K  0B     920K   53M    2412M  97999 1     sleeping 0
97926- Google Chrom 0.0  00:23.02 4    0    58    132    8356K  6752K  0B     6304K  106M   824M   97923 97923 sleeping 501
97923- Google Chrom 0.0  00:57.03 34   1    375+  442+   33M+   35M+   0B     11M    322M+  1098M+ 97923 177   sleeping 501
97477  AppleIDAuthA 0.0  00:00.01 3    2    38    40     404K   240K   0B     396K   46M    2412M  97477 177   sleeping 501
97238- Microsoft Po 0.3  11:10.39 11   2    255+  4211+  178M+  121M+  56M    41M    215M+  1527M+ 97238 177   sleeping 501
92796- dbfseventsd  0.0  00:02.79 1    0    7     27     32K    12K    0B     80K    20K    591M   248   89632 sleeping 501
92788- Dropbox      0.1  08:40.39 44   1    256+  855+   58M+   56M+   12K    47M    299M+  1006M+ 248   1     sleeping 501
92740- adb_aos      0.0  00:00.47 5    0    83    62     484K   340K   0B     732K   40M    618M   92739 1     sleeping 501
92553  Console      0.0  00:07.40 3    0    158   221    4156K  2560K  0B     33M    36M    2509M  92553 177   sleeping 501
89792  com.apple.We 0.0  05:43.72 8    3    372   1057   54M    34M    0B     20M    67M    3556M  89792 1     sleeping 501
89788  Safari       0.0  16:47.59 13   1    2530  2811   134M   82M    812K   48M    505M   4237M  89788 177   sleeping 501
89632- dbfseventsd  0.0  00:06.00 1    0    7     27     4172K  4140K  0B     96K    4148K  591M   248   89631 sleeping 0
89631- dbfseventsd  0.0  00:02.02 1    0    14    26     40K    20K    0B     136K   5280K  583M   248   1     sleeping 0
88379  com.apple.sb 0.0  00:00.02 2    0    49    42     544K   368K   0B     384K   45M    2433M  88379 177   sleeping 501
83267  helpd        0.0  00:01.07 2    0    47    46     472K   304K   0B     696K   45M    2434M  83267 177   sleeping 501
82077  usbmuxd      0.0  00:01.18 3    0    44    45     384K   256K   0B     828K   55M    2423M  82077 1     sleeping 213
```

USC

California

Process Management

You can put a process as background with & (ampersand) after a command. A background job will keep running until it finishes. This allows users to work on different tasks while the background job running. Don't forget your background jobs are consuming resources (CPU, Memory, File I/O etc).

sleep	delay for a specified amount of time
&	run a process as a background job
Ctrl-z/fg	send a foreground job to background and vice versa

```
$ sleep 2
$ sleep 10 &
[1] 18506
$ fg
sleep 10
^Z
[1]+  Stopped    sleep 10
$ bg
[1]+ sleep 10 &
```

Process Management

ps	display currently running jobs
kill/killall	terminate a process (not for PBS job)

```
$ sleep 10 &
[1] 27629
$ ps
  PID TTY          TIME CMD
27362 pts/27   00:00:00 bash
27629 pts/27   00:00:00 sleep
27791 pts/27   00:00:00 ps
$ kill 27629
$ ps
  PID TTY          TIME CMD
27362 pts/27   00:00:00 bash
28248 pts/27   00:00:00 ps
[1]+  Terminated              sleep 10
```

Redirect and Pipe

A special character > redirects output from commands into different channels. Two types of outputs are commonly used, standard output and standard error.

```
[~]$ env                              …
MANPATH=/usr/share/man:
HOSTNAME=hpc-login2
TERM=xterm-256color
SHELL=/bin/bash
HISTSIZE=1000
…
[~]$ env > env.log

[~]$ cat env.log
MANPATH=/usr/share/man:
HOSTNAME=hpc-login2
TERM=xterm-256color
SHELL=/bin/bash
```

Redirect and Pipe (cont.)

A special character | pass output from one command to another command, called pipe. Many command can be daisy-chained by pipe.

grep	print lines matching a pattern
head/tail	show first/last several lines
sort	sort text alphabetically/numerically

Example: Print top 5 users who are consuming CPU except myself:

```
[~]$ ps axuw | grep -v ${USER} | sort -r -n -k 3 | head -n 5
```

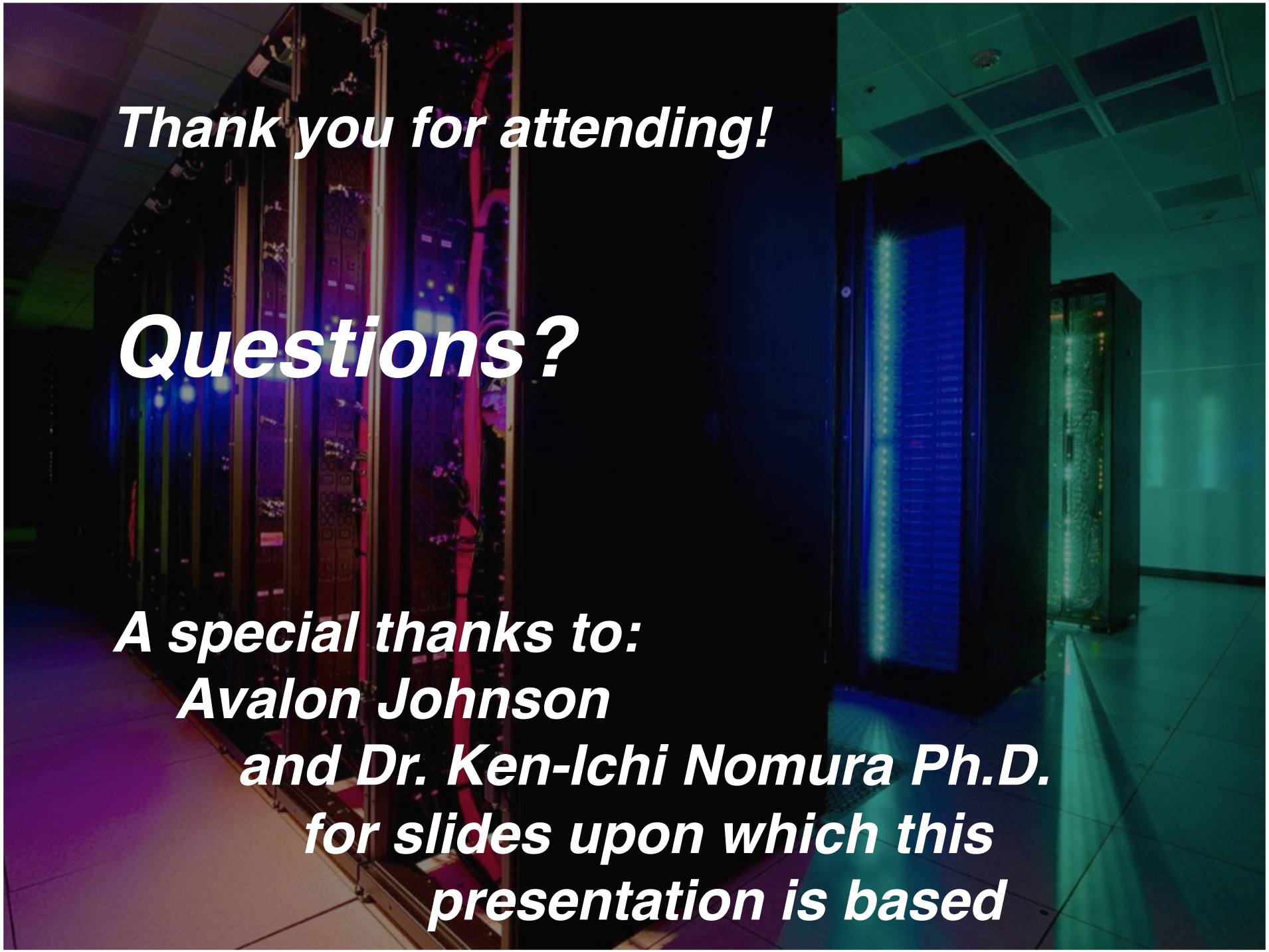
And many more!

There are many useful, even fun, unix commands and utility programs that are beyond the scope of an introductory tutorial.

grep	matches things
find	finds things
sort	sort things
sed	string substitution
awk	?

Linux References

- HPC workshop
  - Introduction to Linux, PBS & the HPC Cluster
- Lynda video (access via USC)
  - https://www.lynda.com/Linux-tutorials/Learn-Linux-Command-Line-Basics/435539-2.html
- Software Carpentry tutorial
  - http://swcarpentry.github.io/shell-novice/
- O'Reilly Books directory
  - http://www.linuxdevcenter.com/cmd/
- Many many web sites… use search



Thank you for attending!

Questions?

A special thanks to:
Avalon Johnson
and Dr. Ken-Ichi Nomura Ph.D.
for slides upon which this
presentation is based