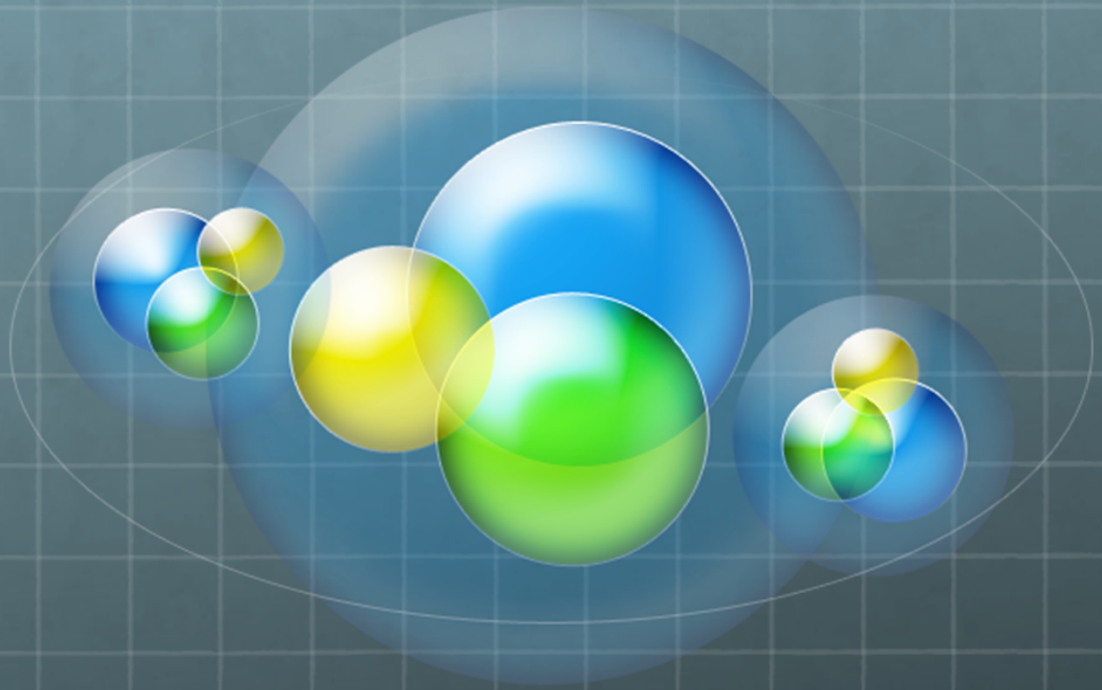


Introduction to the Linux Command Line

Ken Weiss
HITS Computational Research Consulting Division

A word from our sponsor…

- This class is brought to you courtesy of:

Advanced Research Computing – Technical Services

ARC-TS

- For more information please click on:

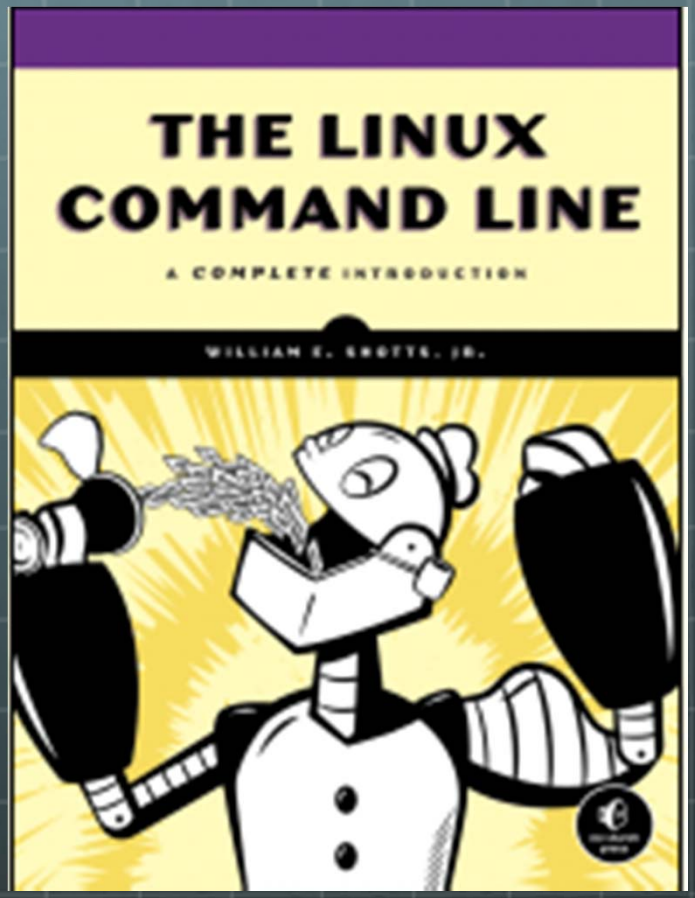
http://arc-ts.umich.edu

Roadmap

- The command shell
- Navigating the filesystem
- Basic commands & wildcarding
- Shell redirection & pipelining
- Editing text files
- Permissions
- Processes
- Environment variables and customizing your session

Course Text

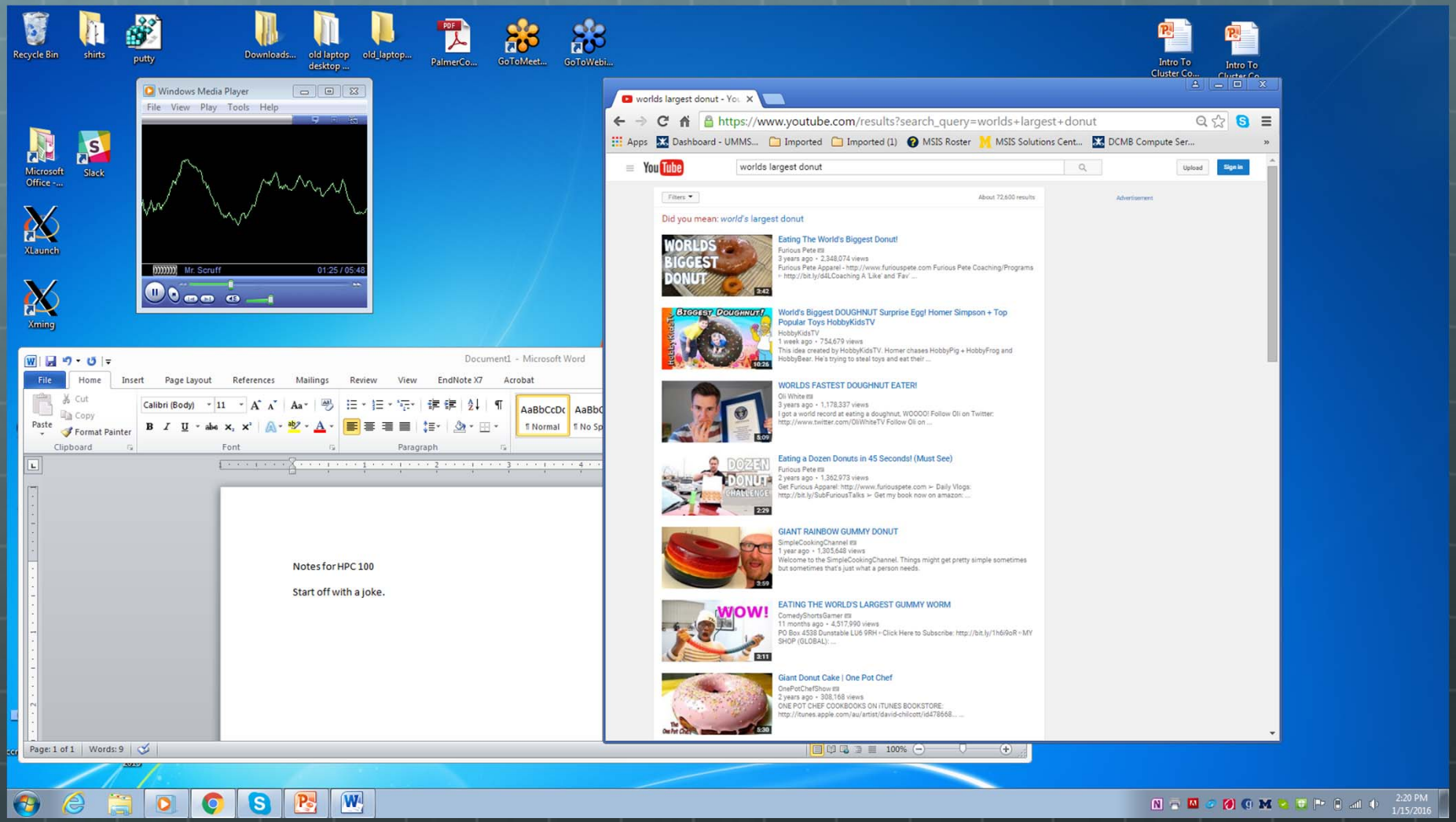
William E Shotts, Jr.,
"The Linux Command Line: A Complete Introduction,"
No Starch Press, January 2012.



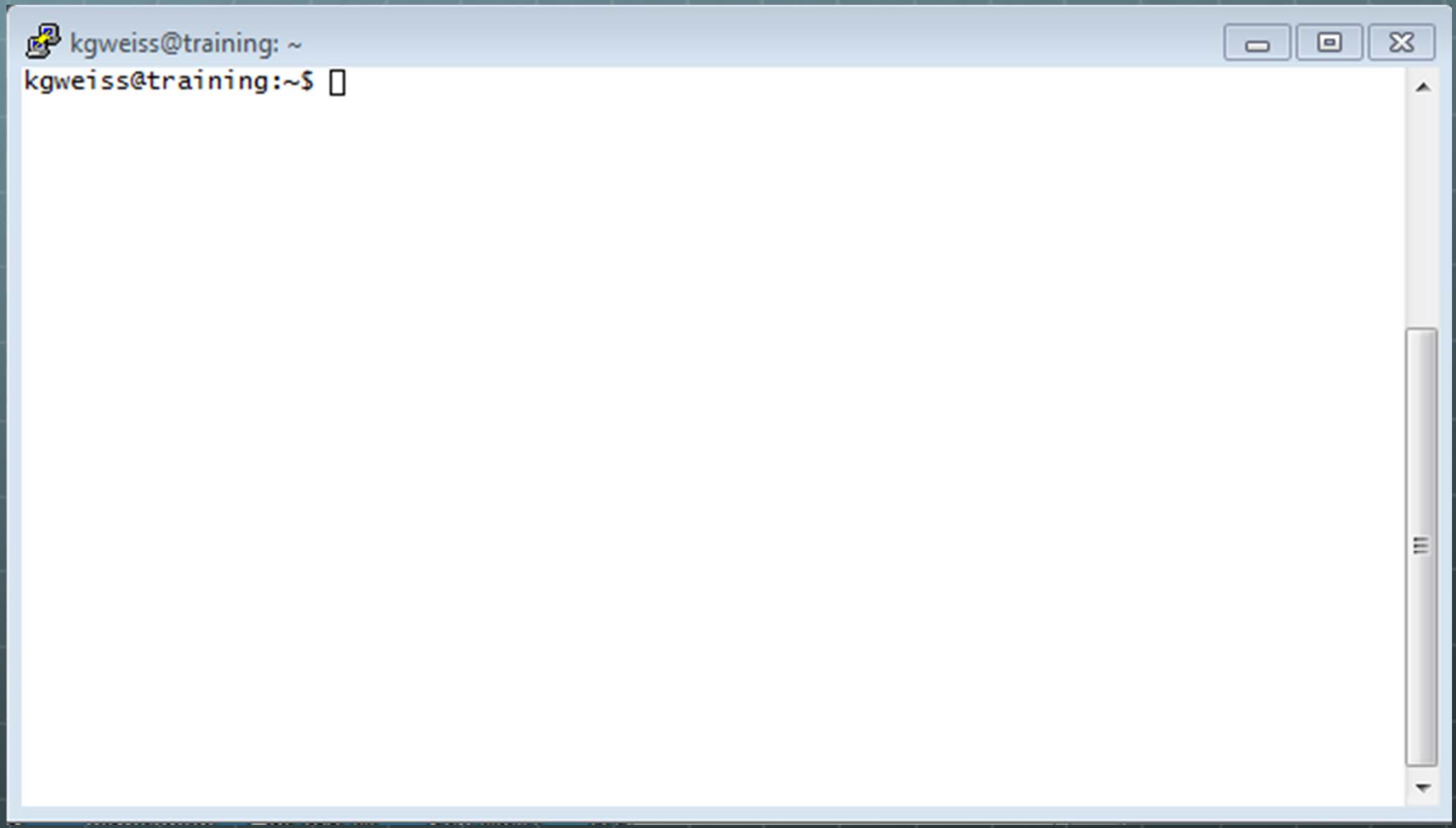
Download Creative Commons Licensed version at
http://downloads.sourceforge.net/project/linuxcommand/TLCL/13.07/TLCL-13.07.pdf.

The command shell

What you are used to using



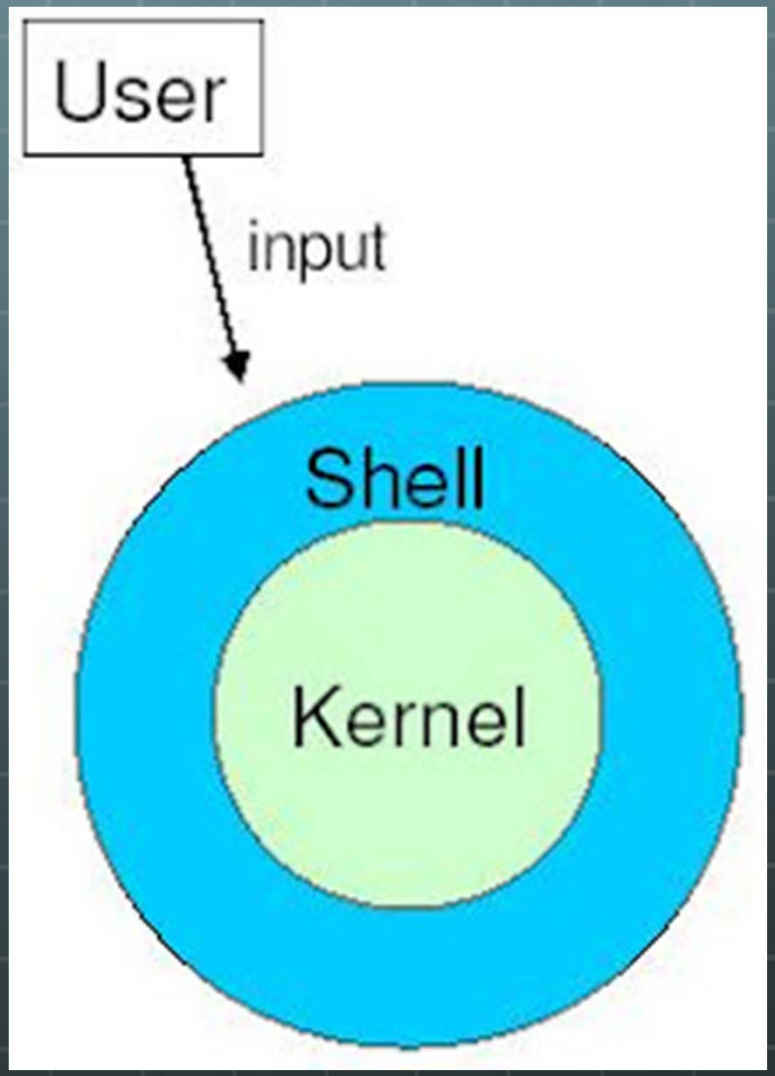
What you will be using



The command shell

- The *command shell* is an application that reads *command lines* from the keyboard and passes them to the Linux operating system to be executed.
- When you login to a remote Linux system, using a tool like `ssh`, you will automatically be connected to a shell.
- Your computing session is kept separate from other user's computing sessions, because they are "enclosed" in a "shell".
- On your desktop, laptop, or tablet, you may have to find and execute a *terminal emulator* application to bring up a shell in a window.

The command shell



http://askubuntu.com/questions/161511/are-the-linux-utilities-parts-of-the-kernel-shell

The command line

- A basic way of interacting with a Linux system
  - Execute commands
  - Create files and directories
  - Edit file content
  - Access the web
  - Copy files to and from other hosts
  - Run HPC jobs

  - … do things you can't do from the conventional point-and-click *Graphical User Interface* (GUI)

Why command line?

1. Linux was designed for the command line
2. You can create new Linux commands using the command line, without programming
3. Many systems provide only the command line, or poorly support a GUI interface
   - Such as most HPC systems
4. Many things can be accomplished only through the command line
   - Much systems administration & troubleshooting
5. You want to be cool

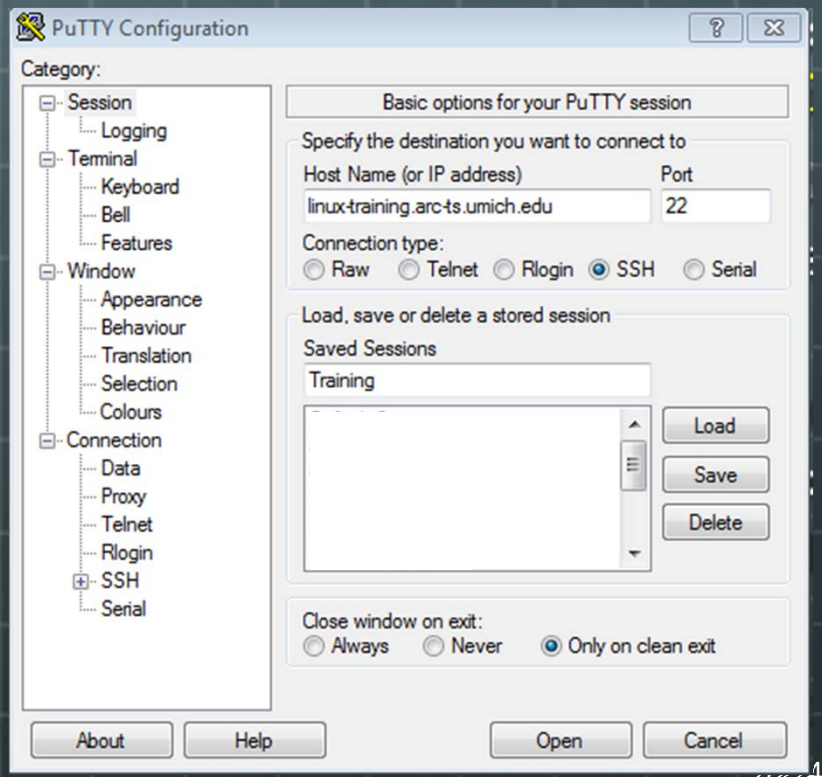
Connecting via ssh

- Terminal emulators
  - Linux and Mac OS X
    - Start Terminal
    - Use `ssh` command
  - Windows
    - U-M Compute at the U (Get Going)
      http://its.umich.edu/computing/computers-software/compute
    - PuTTY
      http://www.chiark.greenend.org.uk/~sgtatham/putty/

Logging in to a host

We will be using the host: linux-training.arc-ts.umich.edu for our class.

- For Mac or other Linux workstation, from a terminal window type: ssh *uniqname*@linux-training.arc-ts.umich.edu
- On a PC, start PuTTY. In the "Host Name (or IP address)" box type: linux-training.arc-ts.umich.edu Click on the "Open" button. Once connected, you will see: "login as:". Type in your uniqname and press: enter



Logging in to a host

- You will be prompted:

  “uniqname@linux-training.arc-ts.umich.edu's password:”

  Enter your Level 1 password and press enter.
- You are now logged into a shell on the linux-training host
- Your shell prompt looks like this:
  uniqname@training:~$

The shell prompt

- The "*uniqname*`@training:~$`" is the shell prompt
  - This means the shell is waiting for you to type something
  - Format can vary, usually ends with "$" , "%" or "#"
    - If $ or %, you have a normal shell
      - This shell has your privileges
    - If #, you have a so-called "root shell"
      - This shell has administrator privileges 
      - You can do a great deal of irreversible damage

Typing into the shell

- Basic input line editing commands
  - Backspace erases previous character
  - Left and right arrow move insertion point on the line
  - Control-c interrupts whatever command you started and returns you to the shell prompt (usually)
  - Control-u erases the line from the beginning to the cursor
  - Control-k erases the line from the cursor to the end
  - Enter executes the line you typed
  - Up and down arrow will access your *command history*
  - Type "`exit`" and press Enter without the quotes to exit the shell
  - Click the red "close" icon at the top of the Terminal window to close it (on a Mac)

Lab 1

Task: Enter some basic commands

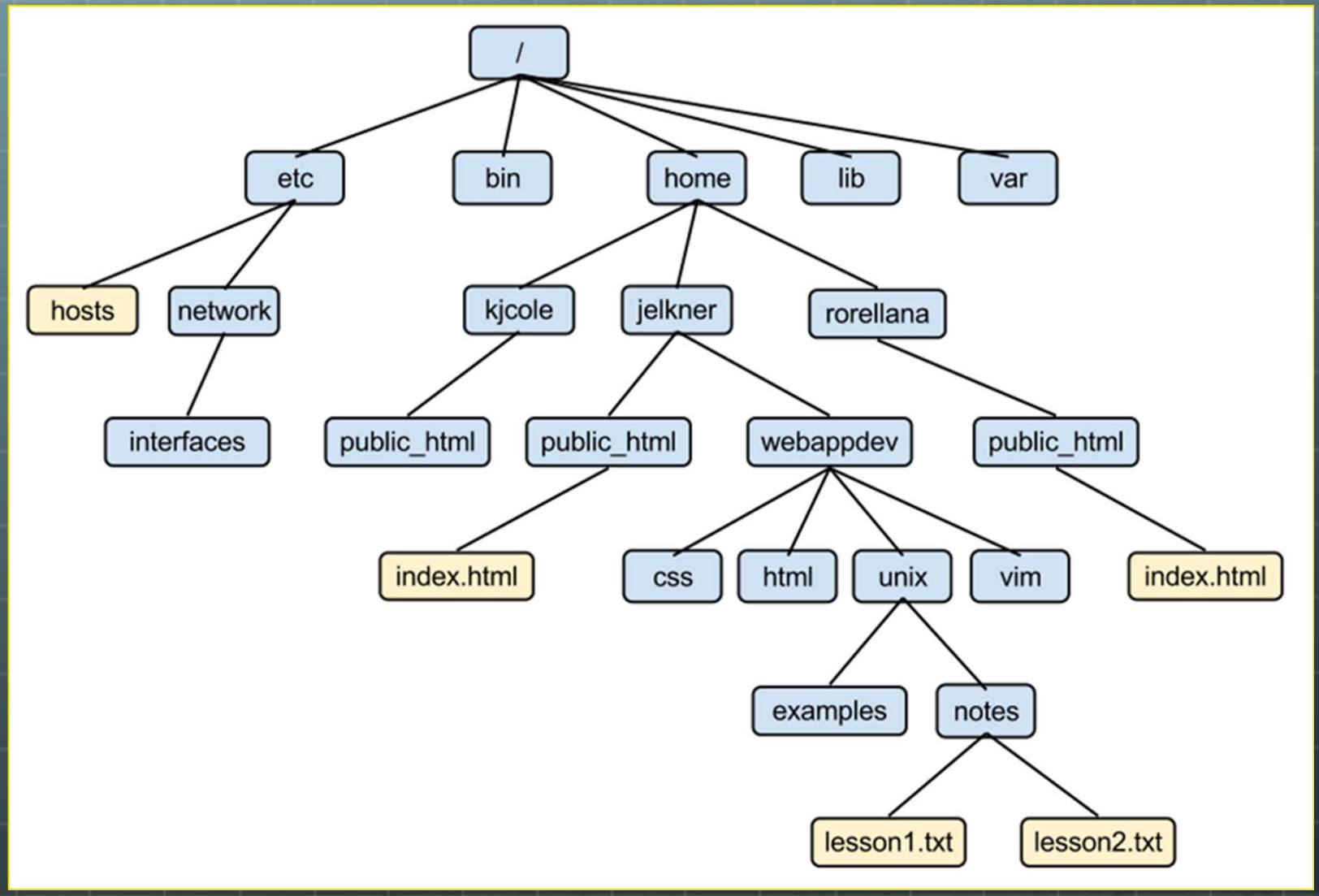
```
~$ date
~$ id
~$ ps
~$ df -kh
~$ who
~$ top      # type Control-c or q to exit
~$ history
```

Navigating the filesystem

Linux Filesystem Concepts

- Files are stored in a *directory* (think: folder)
- Directories may contain other directories as well as files
- A hierarchy of directories is called a *directory tree*
- A directory tree (a connected graph with no cycles) has a single, topmost *root* directory
- A directory tree, rooted at the *system root directory* "/", is called a *filesystem*

A Linux Filesystem



http://www.openbookproject.net/tutorials/getdown/unix/lesson2.html

Linux Filesystem Concepts

- A file is accessed using its *path name*
- Absolute path name
  - */dir1/dir2/…/dirn/filename*
  - */usr/X11R6/bin*
- Relative path name
  - *current-working-directory/filename*
  - *bin*
- Every shell maintains a notion of a *current working directory*
  - Initialized at login to your *home directory*
  - Changed via cd command
- Two special directories
  - . refers to the current directory
  - .. refers to the current directory's *parent directory*
- Many ways to get "home"
  - *~ refers to your home directory*
  - *$HOME is a synonym for ~*
  - *~username refers to a user's home directory*

Basic commands

Prerequisites

Some fundamental commands:

```
~$ file file          # what kind of file is file?
~$ cat file           # display contents of text file
~$ less file          # paginate text file
~$ man command        # get info about command
```

Exercise: figure out how to make the `date` command display the date in Coordinated Universal Time (UTC)

Navigating the filesystem

Some fundamental commands:

```
~$ pwd              # print working directory
~$ cd dir           # make dir the current working directory
~$ cd               # cd to your home dir
~$ cd ~cja          # cd to cja's home dir
~$ mkdir dir        # create directory dir
~$ rmdir dir        # remove (empty) directory dir
~$ rm -fR dir       # remove directory dir (empty or not)
~$ tree             # display dir tree
```

Lab 2

Task: navigate the file system

Commands:

```
~$ cd                  # make your home directory
                         the current working directory
~$ pwd                 # print working directory
~$ mkdir foo           # create directory foo
~$ cd foo              # cd to the foo directory
~$ mkdir bar           # create directory bar
~$ cd ..               # go up one level in the directory tree
~$ tree foo            # display foo's directory tree.
                         (Use tree -A in PuTTY)
```

Listing info on files

ls – list information about files

```
~$ ls                  # list contents of cur dir
~$ ls dir              # list contents of dir
~$ ls -l               # list details of files in cur dir
                         including access, owner & group,
                         size, and last-modified time
~$ ls -t               # list newest files first
~$ ls -R dir           # list all files in tree dir
~$ ls -lt dir          # options can be combined
~$ ls -hl dir          # list all files in human readable
                         format
```

Working with files

These commands manipulate files

```
~$ mv big large       # rename file big to large
~$ cp big large       # copy file big to large
~$ cp -r dir1 dir2    # copy dir tree dir1 to dir2
~$ cp f1 f2 dir       # copy file1 and file2 to directory dir
~$ mkdir dir          # create empty directory dir
~$ rmdir dir          # remove empty directory dir
~$ rm file            # remove file file
~$ rm -r dir          # remove directory tree dir
```

Lab 3

Exercise:

Create a directory named `tutorial` in your home directory. In that directory, create a directory named `sample` and a directory named `test` . Create a file named `msg` in directory `test` that contains a copy of the file `/etc/os-release`.

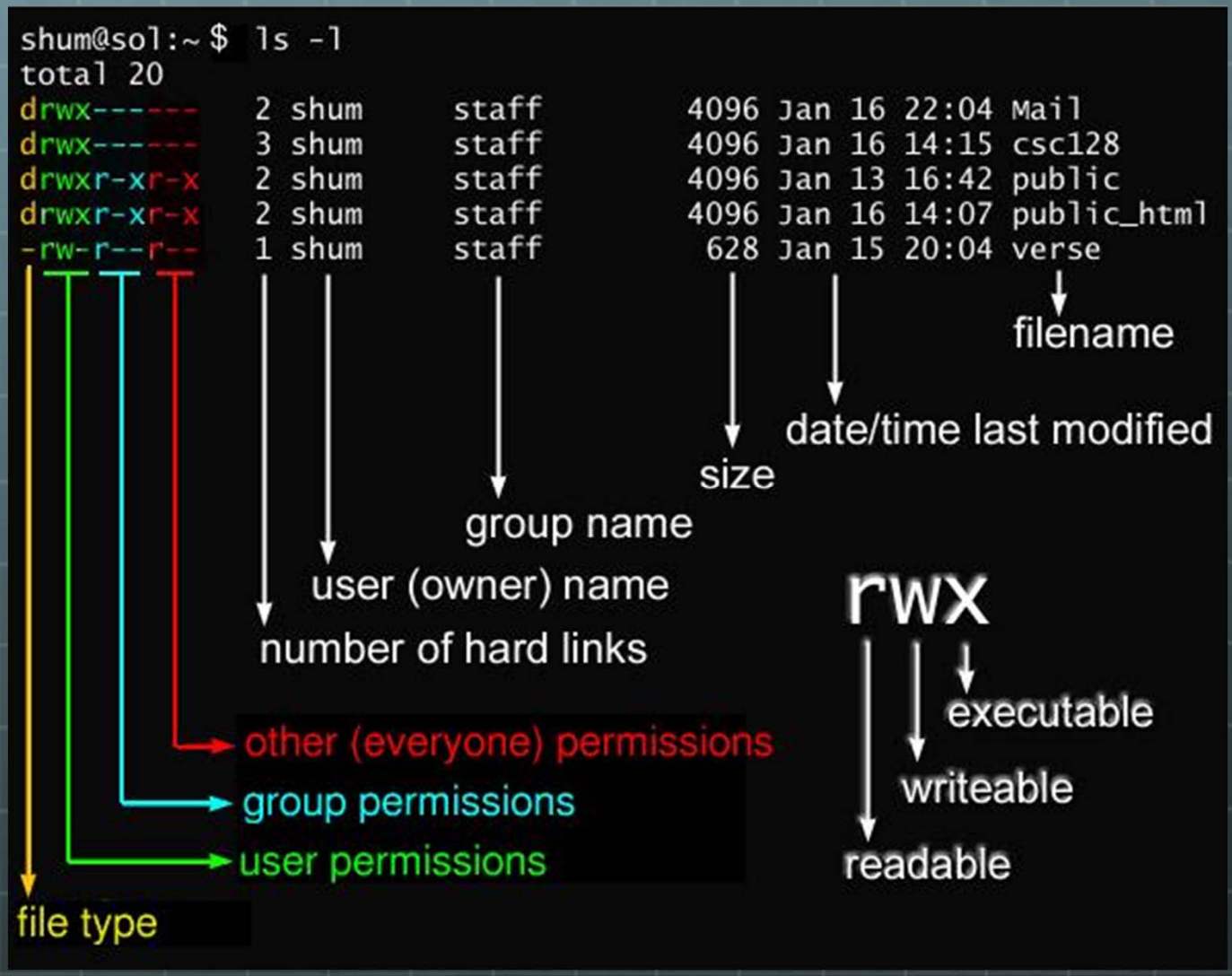
Extra credit: Make the last-modified time of your copy identical to that of `/etc/os-release`. *Hint: look at the options of the copy command*

Permissions

File Permissions

- Three permission bits, aka *mode bits*
  - Files: Read, Write, EXecute
  - Directories: List, Modify, Search
- Three user classes
  - User (File Owner), Group, Other
- `man chmod`

File Permissions



http://www.csit.parkland.edu/~smauney/csc128/fig_permissions.jpg

File Permissions, examples

`-rw------- cja lsait 40 Oct  1 12:03 foo.bz2`

file read and write rights for the owner, no access for anyone else

chmod u=rw,g=r,o= *file*

`-rw-r----- cja lsait 40 Oct  1 12:03 foo.bz2`

file read and write rights for the owner, read for members of the lsait group and no access for others

`drwxr-x--x cja lsait 4096 Oct  1 12:15 bar`

list, modify, and search for the owner, list and search for group, and execute only for others

Lab 4

Task: copy sample files for further exercises

Commands:

```
~$ cd                  # make your home directory
                       the current working directory

~$ pwd                 # print working directory to verify you
                       are in your home directory

~$ mkdir training      # create directory training

~$ cd training         # cd to the training directory

~$ cp -rf /data/examples/IntroLinux/. .

                       # copies sample files to training directory
```

Compression, archiving & wildcards

Compressing and archiving

These commands compress and archive files

```
~$ gzip foo              # compress foo to foo.gz
~$ gunzip foo            # uncompress foo.gz to foo
~$ bzip2 foo             # better compress foo to foo.bz2
~$ bunzip2 foo           # uncompress foo.bz2 to foo

~$ tar -cf foo.tar bar   # archive subtree bar in file foo.tar
~$ tar -xf foo.tar       # restore archive from file foo.tar
~$ tar -tf foo.tar       # list files in archive file foo.tar
~$ tar -zcf foo.tgz bar  # archive and compress
~$ tar -jcf foo.tjz bar  # archive and compress better
```

Exercise: Archive and compress the files in the *training* directory to a file named examples_train.tgz

Wildcards

The shell accepts *wildcarded* arguments

This is also called "shell globbing"

Wildcards:

?	Matches a single character
*	Matches zero or more characters
[*chars*]	Matches any of the *chars*
[c_1-c_2]	Matches chars 'c_1' through 'c_2'
[^chars]	Matches any but the *chars*

```
~$ ls foo.?          # match files named foo.x, where x
                       is any character

~$ echo *.[cs]       # echo files that end in .c or .s

~$ mv [o-z]* save    # move files starting with o through
                       z to directory save

~$ echo [^A-Z]?      # ???
```

Shell redirection & pipelining

Shell redirection

A Linux command can have its inputs and outputs *redirected*

```
~$ ls >myfiles          # put list of files in current
                          directory into file myfiles

~$ ls >>filelist        # add list of files in current
                          directory to end of file filelist

~$ sort <grocery.list   # sort lines from file grocery.list

~$ sort <<EOF           # sort lines entered at keyboard
whiskey                 # (this is a "here document")
bravo
tango
EOF

~$ wc -l </etc/os-release >~/mycounts
                        # count number of lines from file
                          /etc/os-release and put result in
                          file mycounts in my home directory
```

More Linux commands

More useful Linux *tool* commands

```
~$ grep string          # show lines of input containing
                          string
~$ tail                 # show last few lines of input
~$ head                 # show first few lines of input
~$ sort                 # sort the input
~$ du -sh               # report the size of the current directory
~$ du -sh dir           # report the size of directory dir
~$ who                  # gives a list of the users currently
                          logged in
~$ cut -cxx-yy          # keep the output from a command starting
                          at the xx character in the line and ending
                          at the yy character
```

Shell pipelining

A Linux command can have its output connected to the input of another Linux command

```
~$ ls | wc -l          # count files in current
                         directory

~$ last | grep reboot # when did we reboot?
```

Exercises:

How many users are currently logged in?

How many unique user IDs are currently logged in?

Editing text files

Editing text files

- Simple editor
  - nano
    - "What you see is what you get" editor
    - Simple to learn if you want to get started quickly
    - No mouse support. Arrow keys for navigation
- Supported editors
  - vi or vim
  - emacs
    - Powerful but more complex
    - If you have time and inclination to become proficient, spend time here

Text files

- Watch out for source code or data files written on Windows systems
  - Use these tools to analyze and convert source files to Linux format
    - `file`
    - `dos2unix`
    - `unix2dos`

File Transfers

File Transfers

Eventually, you will need to move/copy files to and from your computer and a server/workstation/cluster. You can do this using the secure copy command (scp) in a terminal window.

- To transfer files (i.e. foobar.txt) FROM your local host TO a remote host use:

  ~$ scp foobar.txt your_username@remotehost.edu:/some/remote/directory

- To transfer files (i.e. foobar.txt) FROM a remote host TO your local host use:

  ~$ scp your_username@remotehost.edu:foobar.txt /some/local/directory

- To copy a directory, repeat as above adding the -r flag. (~$ scp -r …)

- Graphical, drag-and-drop scp programs are available for Windows and Mac platforms. (WinSCP – Windows, Cyberduck – Mac)

Demonstration

I will Copy the file headtail.txt from the training directory to /home/kgweiss directory using SCP on the remote host flux-xfer.arc-ts.umich.edu

Processes

Processes

Modern operating systems are usually *multitasking*, meaning that they create the illusion of doing more than one thing at once by rapidly switching from one executing program to another. The Linux kernel manages this through the use of *processes*. Processes are how Linux organizes the different programs waiting for their turn at the CPU.

- On Linux, every program runs in a process
- You can examine these processes

```
man ps
ps
ps ax
top
```

Processes

- You can signal a running process 
  - To stop it, or "kill" it

```
man kill
```

Additional commands

```
~$ quota -Q -s $USER                  # show disk quota for $USER
~$ grep “sometext” somefile           # find & print sometext if found
                                        in somefile
~$ history                            # displays last n commands entered
                                        (execute again with !###)
~$ clear                              # clears the screen
~$ diff -w file1 file2                # compare file1 with file2
                                        ignoring all white space
~$ which command                      # prints the full path to command
~$ acommand | tee filename            # takes the results from acommand and
                                        prints them to the screen and to
                                        the file filename
                                      (Use tee –a to append to filename)
~$ source filename                    # reads and executes the commands contained
                                        in filename
```

Environment Variables and Customizing Your Session

Environment Variables

An environment variable is a named object that contains data used by one or more applications. In simple terms, it is a variable with a name and a value.

- The convention in Linux is for the environment variable to be all uppercase letters.
- To use an environment variable, prefix the variable name with $
- You can see the value of an environment variable by typing:

  ~$ echo $*VARIABLENAME*

- You can see all the environment variables defined for your session by typing: ~$ env
- You can set an environment variable by typing:

  ~$ export *VARIABLENAME = value*

Common Environment Variables

```
USER            # $USER --> user login name
PATH            # $PATH --> a list of
                  directories to look into for programs
                  and files
HISTSIZE        # $HISTSIZE --> number of commands
                  to keep in history
HOSTNAME        # $HOSTNAME --> fully enumerated host name
HOME            # $HOME --> home directory of the user
TERM            # $TERM --> terminal type
SHELL           # $SHELL --> shell type you are using
```

Customizing Your Session

You can modify certain aspects of your session to do your work in terms that are easier/more useful for you. See chapter 11 in Schott's book

- Changes can be made in ~/.bashrc file.
- It is recommended that you use an external file, (in our class, ~/training/.custom) and then source it from the .bashrc file
- You can:
  - Set an alias
  - Export an environment variable
  - Change directories
  - Execute a Linux command or an external program

Exercises

1) Modify the .custom file in the training directory creating a new alias named 'llh' that gives a directory listing in human readable file sizes
2) Create a new environment variable CLASS that points to your training directory
3) Source this file and see if your customizations worked

Any Questions?

- Ken Weiss
  Health Information Technology & Services
  kgweiss@umich.edu
  arc-workshop-instructors@umich.edu
  734 763 7503

References

1. http://en.wikipedia.org/wiki/History_of_Linux
2. http://www.openbookproject.net/tutorials/getdown/unix/lesson2.html
3. William E Shotts, Jr., “The Linux Command Line: A Complete Introduction,” No Starch Press, January 2012. Download Creative Commons Licensed version at http://downloads.sourceforge.net/project/linuxcommand/TLCL/13.07/TLCL-13.07.pdf.
4. Learning the nano editor
    1. The nano text editor - https://www.lifewire.com/beginners-guide-to-nano-editor-3859002
    2. Beginners guide to nano - http://www.howtogeek.com/howto/42980/the-beginners-guide-to-nano-the-linux-command-line-text-editor/
5. Learning the VI editor
    1. VIM Adventures - http://vim-adventures.com/
    2. Graphical cheat sheet - http://www.viemu.com/vi-vim-cheat-sheet.gif
    3. Interactive VIM tutorial - http://www.openvim.com/

Have a Nice Day