TEXAS ADVANCED COMPUTING CENTER

WWW.TACC.UTEXAS.EDU

TEXAS
The University of Texas at Austin

Introduction to Linux

A General Overview of Tools and Tips for the Command Line

PRESENTED BY:

Virginia Trueheart, MSIS

Manager, HPC Frontline

vtrueheart@tacc.utexas.edu

Overview

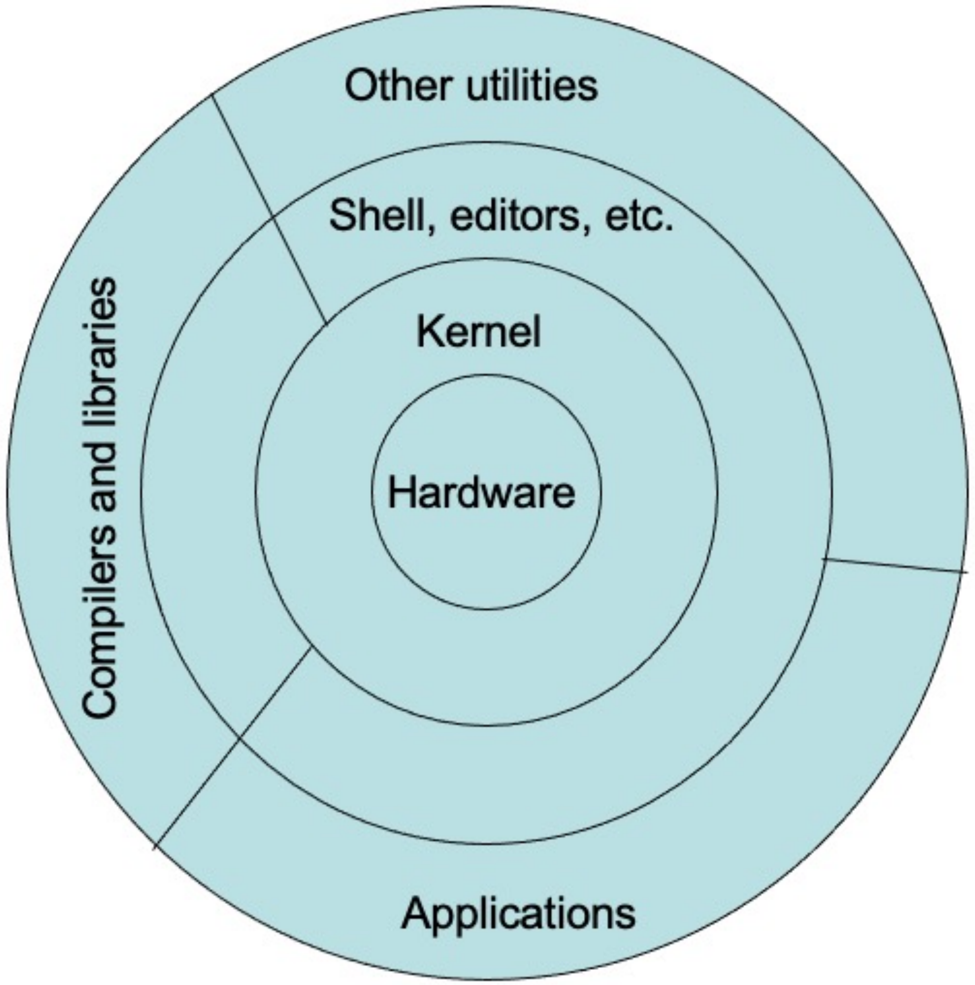
- What is an operating system
- Accessing Linux
- Basic Commands
- Modifying Your Environment

What is an Operating System

- A software interface between the human and the computer
- Controls the execution of other software and code
- Manages all computer resources such as CPU, memory, display, keyboard, etc.
- Linux is one example. See also: Windows, OSX, etc.

How does Linux Work

- Linux has a kernel and one or more shells
- The shell is the command line interface through which the user interacts with the OS. The most commonly used shell is “bash”
- The kernel sits on top of the hardware and is the core of the OS; it receives tasks from the shell and performs them



TACC

Shells

Standard examples:

- bash
- ksh

C language examples:

- csh
- tcsh

Comparison of Shells:
https://en.wikipedia.org/wiki/Comparison_of_command_shells

Cheat Sheets

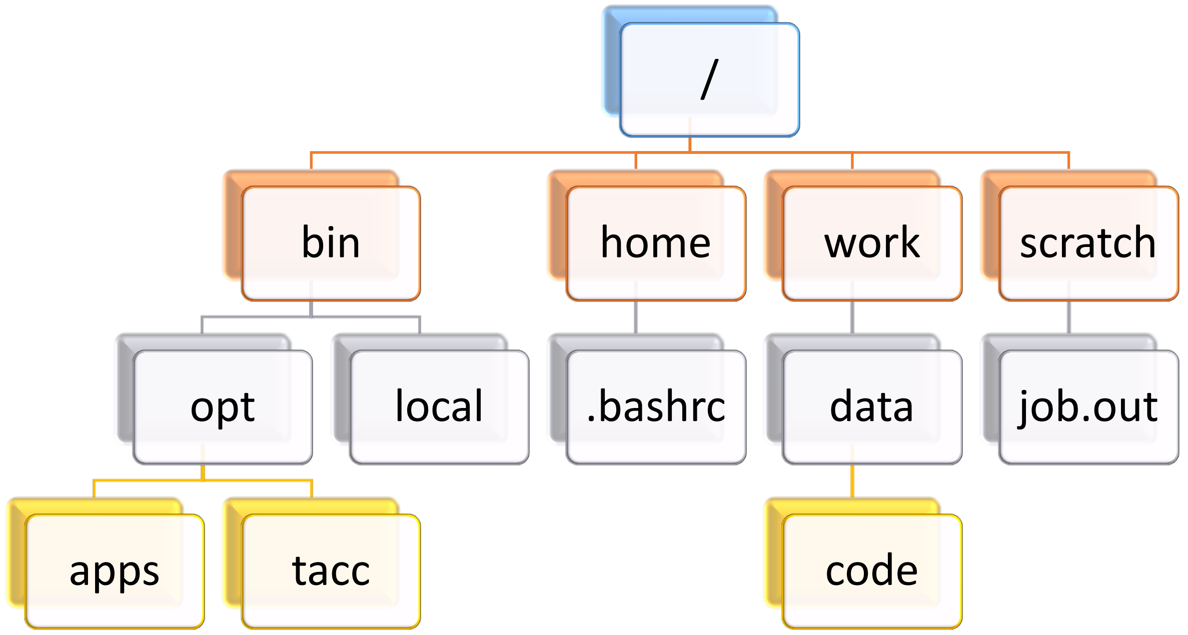
No one memorizes everything.

Printable: https://files.fosswire.com/2007/08/fwunixref.pdf

Interactive: https://devhints.io/bash

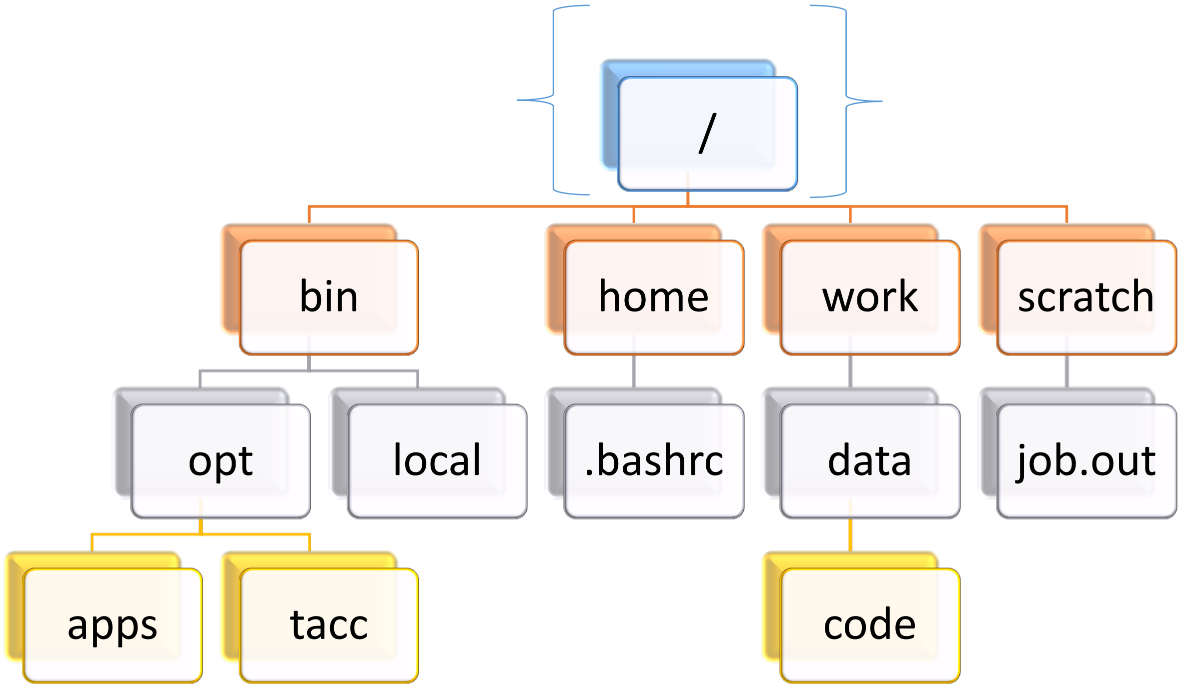
Linux File Systems

- A file is a basic unit of storage and should have a name associated with it
  - Linux is case-sensitive
- Files are organized into directories and sub-directories



Linux File Systems

- In Linux, paths begin at the root directory which is the top-level of the file system and is represented as a forward slash (/)
  - Relative vs. Absolute Path
- Forward slash is used to separate directory and file names



Accessing Linux Locally

On Mac open the Terminal application

On Windows 10 open the Windows Terminal application

Accessing Linux Remotely

- On a Mac or Windows 10 you can use the respective Terminal applications and use secure shell commands to access a TACC machine
- For older versions of Windows you can install an emulator like PuTTY: https://www.putty.org
- Secure shell is made up of **`ssh`**, **`slogin`**, and **`scp`**
- To access a TACC machine you would use an **`ssh`** command

Interacting with a Shell

- Execute commands to the shell by pressing `Enter`
- The shell will start a new process for executing the requested command and display any output generated by the command
- When the process completes, the shell will once again display the prompt line and be ready for further commands
- Additional information can be passed to a shell using arguments, typically indicated by a `-`
- A shell is killed by `exit` or `CTRL-D`

Navigating the File System

When you first start/login, you current working directory is your home directory.

Find out your current directory type:

```
$ pwd
```

List what is in your directory type:

```
$ ls
```

`ls` lists only those ones whose name does not begin with a dot (.) Files beginning with a dot (.) are known as *hidden* files and usually contain important program configuration information.

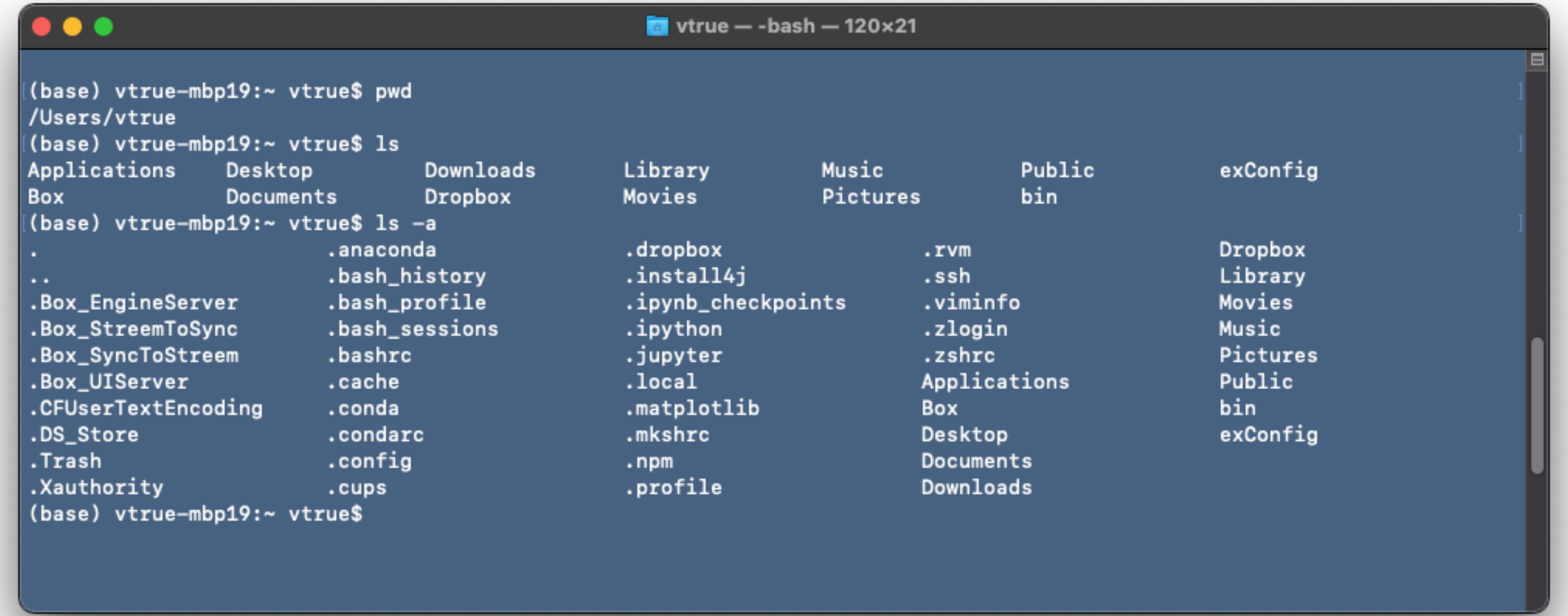
List all the files:

```
$ ls -a
```

TIP!

-a is only one of many command options. Find out more about any command with the 'man' command. Ex: man ls

Example:



vtrue — -bash — 120×21

```
(base) vtrue-mbp19:~ vtrue$ pwd
/Users/vtrue
(base) vtrue-mbp19:~ vtrue$ ls
Applications    Desktop         Downloads       Library         Music           Public          exConfig
Box             Documents       Dropbox         Movies          Pictures        bin
(base) vtrue-mbp19:~ vtrue$ ls -a
.                       .anaconda               .dropbox                .rvm                    Dropbox
..                      .bash_history           .install4j              .ssh                    Library
.Box_EngineServer       .bash_profile           .ipynb_checkpoints      .viminfo                Movies
.Box_StreemToSync       .bash_sessions          .ipython                .zlogin                 Music
.Box_SyncToStreem       .bashrc                 .jupyter                .zshrc                  Pictures
.Box_UIServer           .cache                  .local                  Applications            Public
.CFUserTextEncoding     .conda                  .matplotlib             Box                     bin
.DS_Store               .condarc                .mkshrc                 Desktop                 exConfig
.Trash                  .config                 .npm                    Documents
.Xauthority             .cups                   .profile                Downloads
(base) vtrue-mbp19:~ vtrue$
```

Moving through Directories

The `cd` command allows you to change directories. Pick a directory name from the ls output and `cd` to it.

```
vtrue$ cd Applications
```

To move back up the directory tree you can use `..`

```
vtrue$ cd ..
```

This works for `ls` too

```
vtrue$ ls ..
```

Creating Directories

To make a new directory, use the `mkdir` command

```
login4$ mkdir LinuxIntro
```

To change your working directory, use the `cd` command

```
login4$ cd LinuxIntro
```

TIP!

If you begin typing the name of your directory and then press Tab, the shell will autocomplete the name provided there are no other competing file names

Creating Files

To create a new file use the vim command:

```
login1$ vim test.txt
```

Press `i` to begin inserting text. Ex: `i Hello World!!`

To save and quit, press `Esc` key, and enter `:wq!`

To quit without saving, press `Esc` key, and enter `:q!`

To read your file you can use the `cat` command

Copying Files

Copy the contents of one file to another with the `cp` command

```
cp test.txt copytest.txt
```

You can also use this to copy files between directories

```
mkdir junk
cp copytest.txt ./junk/test2.txt
cd junk
ls
```

Removing Files and Directories

To remove a file use the **`rm`** command

```
rm copytest.txt
```

To remove a directory and its contents you need to use the recursive flag

```
rm -r junk
```

You can also use `rmdir` to remove a directory if it is empty

Moving Files

If you want to move a file rather than create a copy of it, you can use the `mv` command

```
mv test.txt ./junk
```

You can also use this to rename files

```
mv test.txt newTest.txt
```

Try It Out

- Create a file in LinuxIntro called hello.txt that has your name in it
- Create a subdirectory in LinuxIntro called MyDirectory
- Create 2 files inside MyDirectory
- Copy one of your new files from MyDiretory up to LinuxIntro
- Rename the remaining file in MyDirectory

EXTRA *You can use diff to compare two files. Ex: diff test.txt newtest.txt*

Viewing Previous Commands

Want to see what you've been doing in your terminal? Try this `history` command:

```
vtrue$ history
512   cd junk
521   rm -r junk
522   ls
523   history
```

Reviewing Files

If the contents of a display are more than one page, you can use the `more` or `less` commands to page through text a screenful at a time

```
more test.txt
less test.txt
```

`less` allows for both forward and backwards movement

Searching Files

Search using `less` with `/`

```
less test.txt
/World
```

The searched term will be highlighted and you can use `n` to go to the next instance in the file.

`grep` will return a word if it is found in a file

```
grep world test.txt
```

To search for a case specific instance use the `-i` option

```
grep -i world test.txt
```

Searching Continued

Search for a phrase by using quotes:

```
grep "Hello World" test.txt
```

`wc` (with options) will tell you know how many words or lines a file has

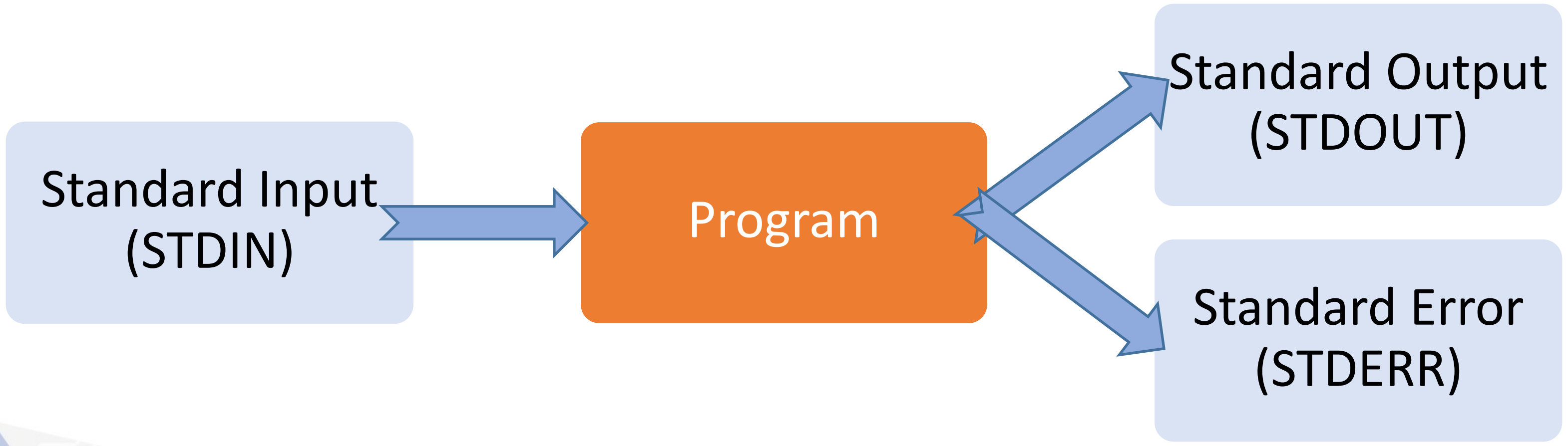
```
wc -w test.txt
wc -l test.txt
```

Summary of the Basics

command	meaning
`cp <file1> <file2>`	copy file1 to file2
`mv <file1> <file2>`	move/rename file1 to file2
`rm`	remove a file
`cat`	display a file
`more/less <file>`	display a file one screen at a time
`head/tail <file>`	display first/last 10 lines of file
`grep 'word' <file>`	search a file for ‘word’
`wc <file>`	count number of word/lines/chars

Redirecting Output

- Typically, in a Linux environment you will type in the name of a program and some command line options
- The shell establishes 3 separate I/O streams: standard input, standard output, and standard error
- Most processes initiated by Linux commands write to standard output (i.e. the screen) and take their input from standard input (the keyboard).



Redirecting Output Cont.

- The shell can attach things other than your keyboard to standard input:
    - A File (the contents of the file are fed to a program as if you typed it)
    - A pipe (the output of another program is fed as input as if you typed it)

- The shell can also attach things other than your screen to standard output:
    - A File (the output of a program is stored in a file)
    - A pipe (the output of another program is fed to the input of another program)

Commands for Redirecting Output

To tell the shell to store the output of your program in a file, follow the command line for the program with `>` followed by the filename:

```
$ ls /LinuxIntro > job.log
$ cat job.log
```

You can concatenate two files using `cat`, and redirect the output to a new file:

```
$ cat test.txt job.log > textblob.txt
```

`>` will create (or overwrite) the file, if you want to append to and existing file use `>>`

```
$ whoami >> job.log
$ cat job.log
```

Redirecting Input

The `sort` command will sort alphabetically or numerically, it takes input from standard input (the keyboard). To see how it works, type sort followed by a list of three fruits and then the key combo CTRL+D to end the input.

```
$sort

mango

tomato

apple

<[Ctrl][d]>
```

Type `who` to see who is currently logged into the system, and redirect the output from the screen to a file

```
$ who

$ who > users.log
```

redirect the input of users.log to the command `sort`

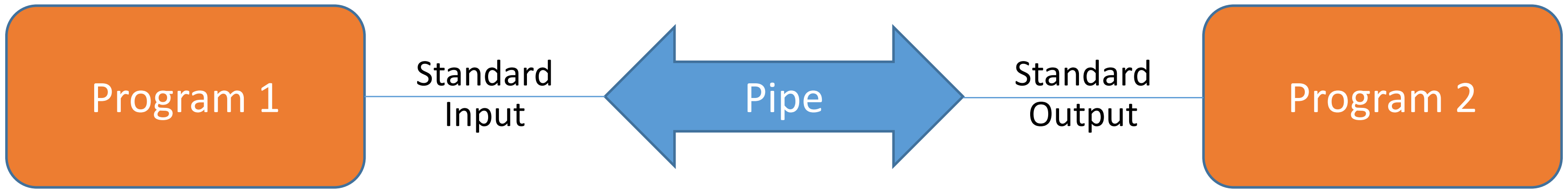
```
$ sort < users.log
```

redirect the sorted output from standard out to a file

```
$ sort < users.log > sorted_users.log
```

Pipes

- A pipe is a holder for a stream of data
- A pipeline is a set of processes chained by their standard streams, so that the output of each process (stdout) feed directly as input (stdin) of the next one
- This is handy for using multiple commands together to perform a task



Examples of Pipes

- How many times does the word “you” appear in any of your files?

```
$grep –oi “you” test.txt | wc –l
```

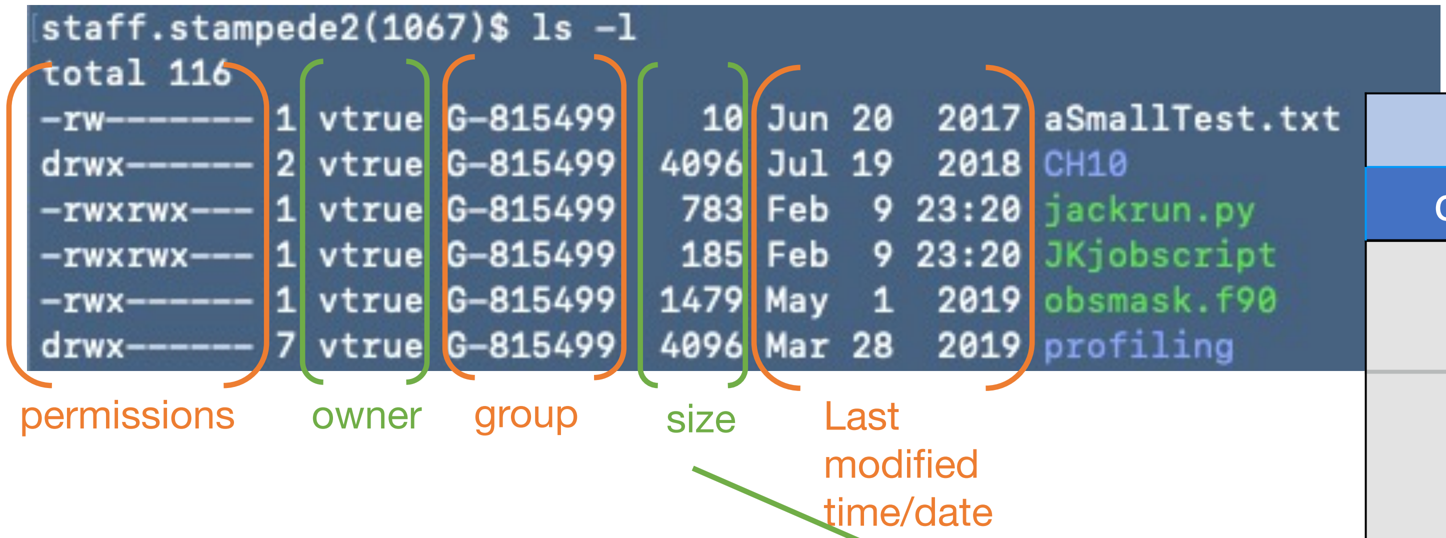
- `| less` is useful in many cases when you have more than a screenful of text, `| grep` is useful when you are looking for something specific:

```
$ history | grep grep
$ ls -lat | less
$ who | sort
```

Another Quick Summary

command	meaning
`command > file`	redirect stdout to file
`command >> file`	append stdout to file
`command < file`	redirect stdin from a file
`command1 | command2`	pipe the stdout of command1 to the stdin of command2
`cat file1 file2 > file3`	concatenate file1 and file2
`sort`	sort data
`who`	list of current users logged in
`wc <file>`	count number of word/lines/chars

File Attributes and Permissions



```
staff.stampede2(1067)$ ls -l
total 116
-rw-------- 1 vtrue G-815499   10 Jun 20  2017 aSmallTest.txt
drwx------- 2 vtrue G-815499 4096 Jul 19  2018 CH10
-rwxrwx---- 1 vtrue G-815499  783 Feb  9 23:20 jackrun.py
-rwxrwx---- 1 vtrue G-815499  185 Feb  9 23:20 JKjobscript
-rwx------- 1 vtrue G-815499 1479 May  1  2019 obsmask.f90
drwx------- 7 vtrue G-815499 4096 Mar 28  2019 profiling
```

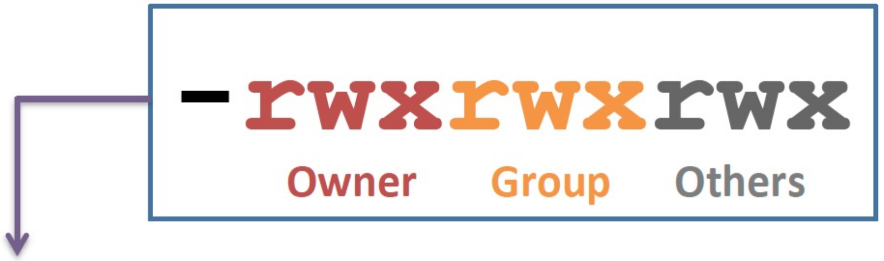
time/size attributes	
command	**meaning**
`ls -l`	when the file was last modified
`ls -lc`	when the file was last changed (change of wonder, permissions, etc)
`ls -lu`	when the file was last accessed
`ls -lt`	chronological listing
`ls -lh`	"human readable" size
`stat <filename>`	display date-related attributes

Permissions

- Each file has a set of *permissions* that control who can can access the file (and how). There are three different types of permissions:
  - read (r), write (w), execute (x)
- In Unix/Linux there are permission levels associated with three types of people that might access a file:
  - owner (you)
  - group (a group of other users that you set up or belong to)
  - world (anyone else browsing around on the file system)

Permissions Cont.

- Meaning for Files:
  - r allowed to read
  - w allowed to write
  - x allowed to execute
- Meaning for Directories:
  - r allowed to see names of the files
  - w allowed to add and remove files
  - x allowed to enter the directory (This implies that you may read files in the directory provided you have read permission on the individual files.)



Symbol	File type
-	plain file
d	directory
l	symbolic link

Changing Permissions

The `chmod` command changes the permissions associated with a file or directory. Only the owner of a file can use `chmod` to change the permissions of a file.

```
chmod [mode] <filename>
```

The mode can be specified by a symbolic representation or an octal number (both function the same)

Multiple symbolic operations can be given separated by commas.

Symbolic Representation

Symbolic mode representation has the following format:

```
[ugoa][+-=][rwxX...]
```

u user	+ add permission	r read
g group	- remove permission	w write
o other	= set permission	x execute
a all		X execute*

The X permission option is very handy, it sets to execute only if the target is a directory or already has execute permissions

Octal Representation

Number	Permission	rwx
7	read, write, execute	rwx
6	read and write	rw-
5	read and execute	r-x
4	read only	r--
3	write and execute	-wx
2	write only	-w-
1	execute only	--x
0	none	---

See also: https://chmod-calculator.com

Octals Explained

- Octal mode uses a single argument string which describes the permissions for a file (3 digits)
- Each digit of this number is a code for each of the three permission levels (user, group, world)
- Permissions are set according to the following numbers:
  - Read bit will add 4 to the total (in binary 100)
  - Write bit will add 2 to the total (in binary 010)
  - Execute bit will add 1 to the total (in binary 001)
  - Sum the individual permissions to get the desired combination. (e.g. if you want the owner only to have rwx permissions, group to have read only and world to have no access:

	Permission Level
0	no permissions
1	execute only
2	write only
3	write and execute (1+2)
4	read only
5	read and execute (4+1)
6	read and write (4+2)
7	read, write and execute (4+2+1)

owner	4(r)+2(w)+1(e)	7
group	4(r)+0(w)+0(e)	4
world	0(r)+0(w)+0(e)	0
`chmod 740 <filename>`		

Customizing Your Shell

- Each shell supports some level of customization
  - User prompt settings
  - Environment variable settings
  - aliases
- Customization takes place in start up files which are read by the shell when it starts up
  - Global files are read first. These are provided by the system administrators (e.g. /etc/profile)
  - Local files are then read in the user's home directory to allow for additional customization

Shell Startup Files

sh, ksh / bash	csh / tcsh
sh, ksh:	**csh:**
~/.profile	~/.cshrc
bash:	~/.login
~/.bash_profile	~/.logout
~/.bash_login	**tcsh:**
~/.profile	~/.tshrc
~/.bashrc	~/.cshrc
~/.bash_logout	~/.login
	~/.logout

Please note that on TACC systems we provide an alternate location for customization files to avoid overriding system defaults:

```
BASH: ~/.profile_user
CSH/TCSH: ~/.login_user
~/.cshrc_user
```

Aliases

Aliases are essentially a keyboard shortcut that allows you to complete a longer command in fewer characters. Some useful examples include:

- prevent common typo: `alias sl="ls"`
- prevent common typo protect yourself from `rm *`: `alias rm="rm -i"`
- list all files ordered by date: `alias lt="ls -alrt –color=auto"`
- use arguments: `alias histg="history | grep"`

Environment Variables

- Unix/Linux shells maintain a list of environment variables which have a unique name and a value associated with them:
  - some of these parameters determine the behavior of the shell
  - also determine which programs get run when commands are entered (and which libraries they link against)
  - provide information about the execution environment to programs

- We can access these variables:
  - set new values to customize the shell
  - find out their values to accomplish a task

Examples of Env Vars

To view ALL environment variables, use the `env` command:

```
$ env
```

If you know what you are looking for, you can use `grep`

```
$ env | grep PWD
```

Use the `echo` command to print variables (the $ prefix is required to access the value of the variable):

```
$ echo $PWD
```

You can also use the environment variable in conjunction with other commands:

```
$ ls $PWD
```

A Special Env Var: PATH

- Each time you provide the shell a command to execute, it does the following:
  - Checks to see if the command is a built-in shell command
  - If it is not a built-in command, the shell tries to find a program whose name matches the desired command
- How does the shell know where to look on the file system?
  - The PATH variable tells the shell where to search for programs

PATH

See what your $PATH is with echo:

```
$ echo $PATH
```

```
/work/03658/vtrue/Software/miniconda3/bin:/opt/apps/xalt/xalt/bi
n:/opt/apps/intel18/python3/3.7.0/bin:/opt/apps/ooops/1.4/bin:/o
pt/apps/cmake/3.16.1/bin:/opt/apps/autotools/1.1/bin:/opt/apps/g
it/2.24.1/bin:/opt/apps/intel18/impi/18.0.2/bin:/opt/intel/compi
lers_and_libraries_2018.2.199/linux/mpi/intel64/bin:/opt/apps/li
bfabric/1.7.0/bin:/opt/intel/compilers_and_libraries_2018.2.199/
linux/bin/intel64:/opt/apps/gcc/6.3.0/bin:/usr/lib64/qt-
3.3/bin:/usr/local/bin:/bin:/usr/bin:/opt/dell/srvadmin/bin:.
```

Other Important Variables

PWD	current working directory
MANPATH	where to find man pages
HOME	home directory of user
MAIL	where your e-mail is stored
TERM	what kinds of terminal you have
PRINTER	specifies the default printer name
EDITOR	used by many apps to identify editor pref (vi, emacs)
LD_LIBRARY_PATH	search path for dynamic runtime library

Setting Env Vars

The syntax you use will depend on your shell. Since we're working in bash we'll be using the `export` command.

```
$ export FRUIT=mango
$ echo $FRUIT
mango
```

Note: environment variables that you set interactively are only available in your current shell:

- if you spawn a new shell these settings will be lost
- to make permanent changes you can alter the login scripts that affect your shell e.g .profile_user

Questions

Chat is open so please ask there about anything you would like for the next few minutes and we'll try to cover as many things as we can.

Contact

Contact TACC support via the TACC User Portal here: https://portal.tacc.utexas.edu/tacc-consulting

Thanks for participating in this Introduction to Linux Tutorial!