UT Southwestern Medical Center | BioHPC

The Linux Command Line & Shell Scripting

[web] portal.biohpc.swmed.edu
[email] biohpc-help@utsouthwestern.edu

David Trudgian - Updated for 2015-02-18

Overview

The Linux command line is provided by a *shell.*

By far the most common shell on linux is *bash* (Bourne Again SHell)

The shell lets us run other programs or commands. It also has a lot of built-in commands, and is a programing language in itself (bash script).

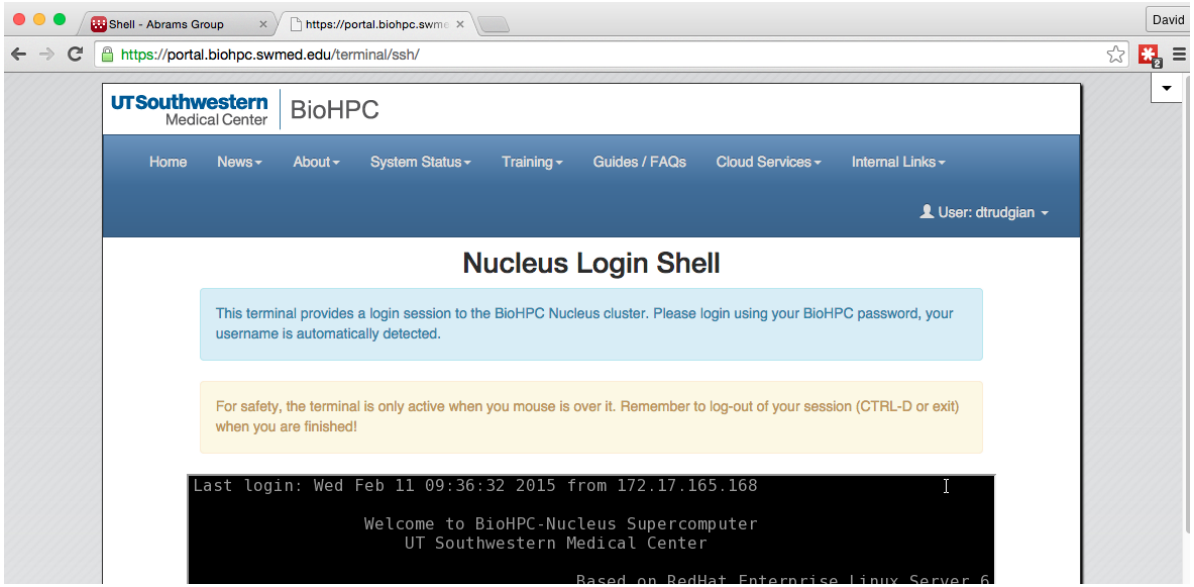
The shell is scary, but a little knowledge can make difficult things easy, and time-consuming things quick.

Follow Along…

Today is mostly demos. Slides and cheat sheets for reference. Stop me with questions!

You can follow along using the Nucleus Web terminal on the BioHPC portal:

https://portal.biohpc.swmed.edu/terminal/ssh/



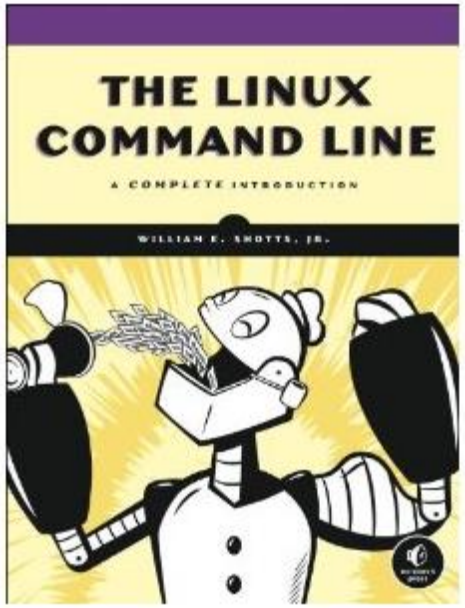
A Free Book

Free, Creative-Commons PDF

On the portal

Training -> Slides & Handouts

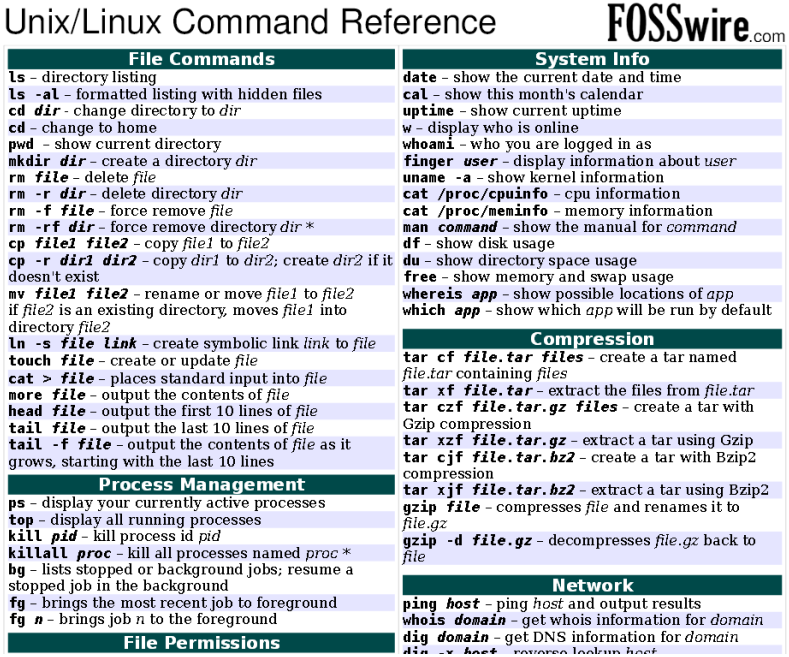
http://linuxcommand.org/tlcl.php



Cheat Sheets

On the portal

Training -> Slides & Handouts



Unix/Linux Command Reference — FOSSwire.com

File Commands

ls – directory listing
ls -al – formatted listing with hidden files
cd *dir* - change directory to *dir*
cd – change to home
pwd – show current directory
mkdir *dir* – create a directory *dir*
rm *file* – delete *file*
rm -r *dir* – delete directory *dir*
rm -f *file* – force remove *file*
rm -rf *dir* – force remove directory *dir* *
cp *file1 file2* – copy *file1* to *file2*
cp -r *dir1 dir2* – copy *dir1* to *dir2*; create *dir2* if it doesn't exist
mv *file1 file2* – rename or move *file1* to *file2*
if *file2* is an existing directory, moves *file1* into directory *file2*
ln -s *file link* – create symbolic link *link* to *file*
touch *file* – create or update *file*
cat > *file* – places standard input into *file*
more *file* – output the contents of *file*
head *file* – output the first 10 lines of *file*
tail *file* – output the last 10 lines of *file*
tail -f *file* – output the contents of *file* as it grows, starting with the last 10 lines

Process Management

ps – display your currently active processes
top – display all running processes
kill *pid* – kill process id *pid*
killall *proc* – kill all processes named *proc* *
bg – lists stopped or background jobs; resume a stopped job in the background
fg – brings the most recent job to foreground
fg *n* – brings job *n* to the foreground

File Permissions

System Info

date – show the current date and time
cal – show this month's calendar
uptime – show current uptime
w – display who is online
whoami – who you are logged in as
finger *user* – display information about *user*
uname -a – show kernel information
cat /proc/cpuinfo – cpu information
cat /proc/meminfo – memory information
man *command* – show the manual for *command*
df – show disk usage
du – show directory space usage
free – show memory and swap usage
whereis *app* – show possible locations of *app*
which *app* – show which *app* will be run by default

Compression

tar cf *file.tar files* – create a tar named *file.tar* containing *files*
tar xf *file.tar* – extract the files from *file.tar*
tar czf *file.tar.gz files* – create a tar with Gzip compression
tar xzf *file.tar.gz* – extract a tar using Gzip
tar cjf *file.tar.bz2* – create a tar with Bzip2 compression
tar xjf *file.tar.bz2* – extract a tar using Bzip2
gzip *file* – compresses *file* and renames it to *file.gz*
gzip -d *file.gz* – decompresses *file.gz* back to *file*

Network

ping *host* – ping *host* and output results
whois *domain* – get whois information for *domain*
dig *domain* – get DNS information for *domain*
dig -x *host* – reverse lookup *host*

v1.3 — steve-parker.org #! — Sept 2012

Shell Scripting Cheat Sheet for Unix and Linux

Online: http://steve-parker.org/sh/sh.shtml
Book: http://steve-parker.org/shellscripting

File Redirection

> file	create (overwrite) file
>> file	append to file
< file	read from file
a | b	Pipe 'a' as input to 'b'

Common Constructs

$ while read f > do > echo "Line is $f" > done < file	read text file line by line note : "$" prompt becomes ">"
$ grep foo myfile afoo foo foobar	find lines in myfile containing the text "foo"
$ cut -d: -f5 /etc/passwd Steve Parker	get 5th field delimited by colon
$ cmd1 || cmd2	run cmd1; if fails, run cmd2
$ cmd1 && cmd2	run cmd1; if it works, run cmd2
case $foo in a) echo "foo is A" ;; b) echo "foo is B" ;;	act upon the value of a variable note that ";;"

Test Operators

if ["$x" -lt "$y"]; then # do something fi	

Numeric Tests

lt	less than
gt	greater than
eq	equal to
ne	not equal
ge	greater or equal
le	less or equal

File Tests

nt	newer than
d	is a directory
f	is a file
x	executable
r	readable
w	writeable

String Tests

=	equal to
z	zero length
n	not zero length

Logical Tests

&&	logical AND

Variable Substitution

${V:-default}	$V, or "default" if unset
${V:=default}	$V (set to "default" if unset)
${V:?err}	$V, or "err" if unset

Conditional Execution

cmd1 || cmd2	run cmd1; if fails, run cmd2
cmd1 && cmd2	run cmd1; if ok, run cmd2

Files

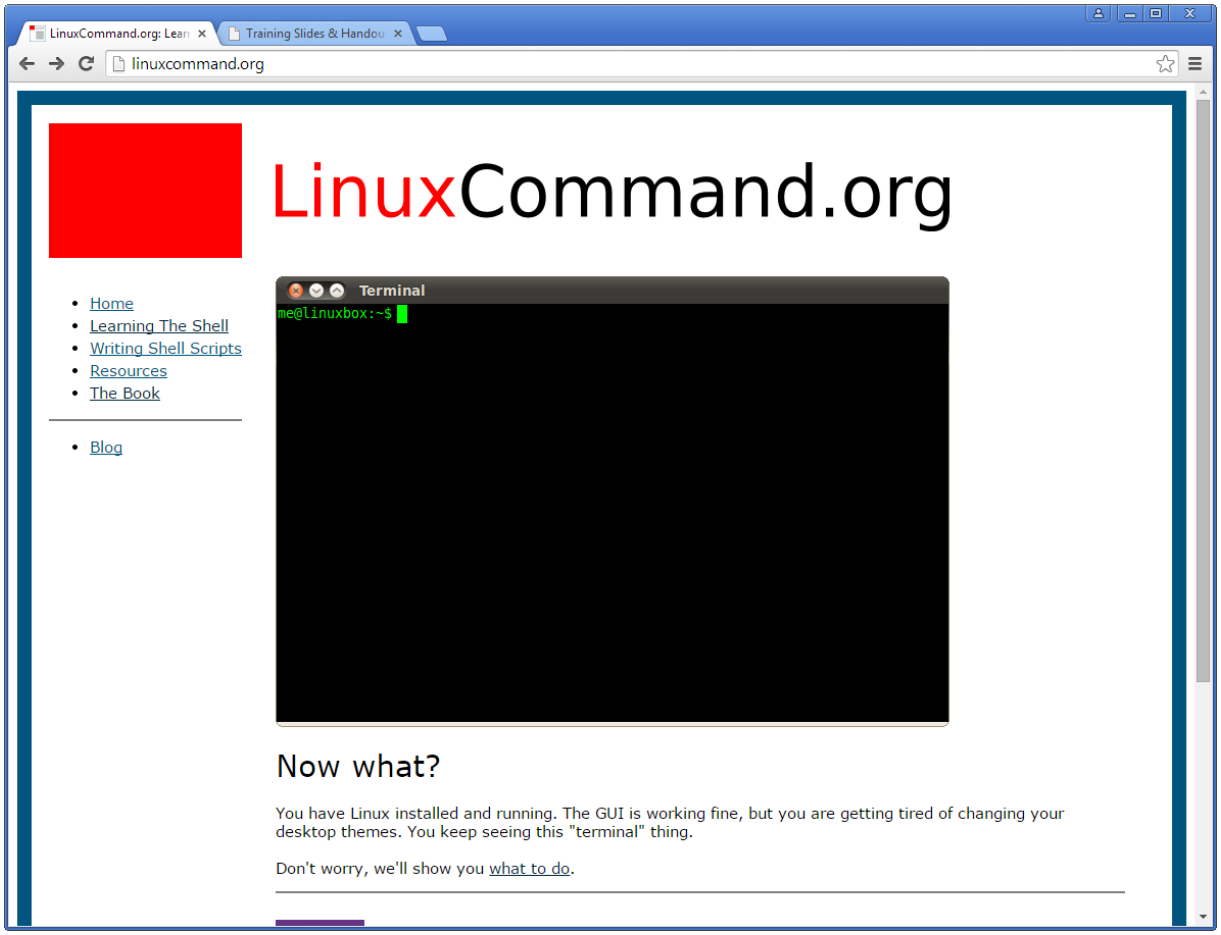
mv /src /dest	move /src into /dest
ls a*	list files beginning with "a"
ls *a	list files ending with "a"
ls -ltr	list oldest first, newest last
ls -lSr	list smallest first, biggest last
ls -a	list all files, including hidden
find /src -print \ | cpio -pudvm	copy /src into current directory, preserving links, special devices, etc.

Preset Variables

$SHELL	what shell am I running?
$RANDOM	provides random numbers
$$	PID of current process
$?	return code from last cmd
$!	PID of last background cmd

Generally Useful Commands

Tutorial Website



man pages

Get a command's help page: man <command>

```
dtrudgian@biohpcws064:~/cli_scripts
10:41 AM $ man ls
```

```
LS(1)                          User Commands                          LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about  the FILEs (the current directory by default).
       Sort entries alphabetically if none of -cftuvSUX nor --sort.

       Mandatory arguments to long options are  mandatory  for  short  options
```

Press q or Ctrl-C to exit the man page

The Basics – Listing files, changing, creating and removing directories

Command	What does it do?
`pwd`	Print Working Directory – where am I?
`ls`	List files in current directory
`ls -lah`	List files with detail (l = long format, a = all files including hidden, h = human readable file sizes)
`mkdir cli_training`	Make a directory called cli_training
`mkdir –p cli_training/ex1/oops`	Make a directory path structure, all directories necessary if they don't already exist.
`rmdir cli_training/ex1/oops`	Remove the cli_training/ex1/oops directory (if empty!)
`ls –lah cli_training`	List files inside the cli_training directory
`cd cli_training`	Change directory into cli_training
`cd ..`	Change directory to .. which is the parent directory – i.e. up one level.
`rm –r cli_training`	Remove recursively (-r) the cli_training directory (the directory and everything inside it).

The Basics – Copying and Moving Files and Directories

Command	What does it do?
`mkdir cli_training`	Make the directory cli_training
`touch cli_training/file1`	Create an empty file called file1 inside cli_training
`mkdir new_training`	Make the directory new_training
`cp cli_training/file1 new_training/`	Copy file1 from cli_training into new_training
`cp –r new_training newer_training`	Copy new_training recursively (directory and everything inside) to a directory called newer_training)
`mv new_training old_training`	Move/rename new_training directory to old_training
`mkdir all_training`	Make a directory all_training
`mv newer_training old_training all_training/`	Move the newer_training and old_training directories inside all_training

Wildcards

* Match any number of characters

`ls *`	Any file
`ls notes*`	Any file beginning with notes
`ls *.txt`	Any file ending in .txt
`ls *2015*`	Any file with 2015 somewhere in its name

? Match a single character

`ls data_00?.txt`	Matches data_001, data002, data_00A etc.

[] Match a set of characters

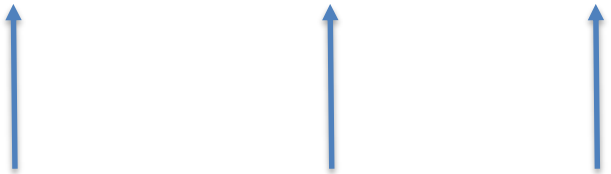
```
ls data_00[0123456789].txt
ls data_00[0-9].txt
```

Matches data_001 – data_009, not data_00A

File Permissions

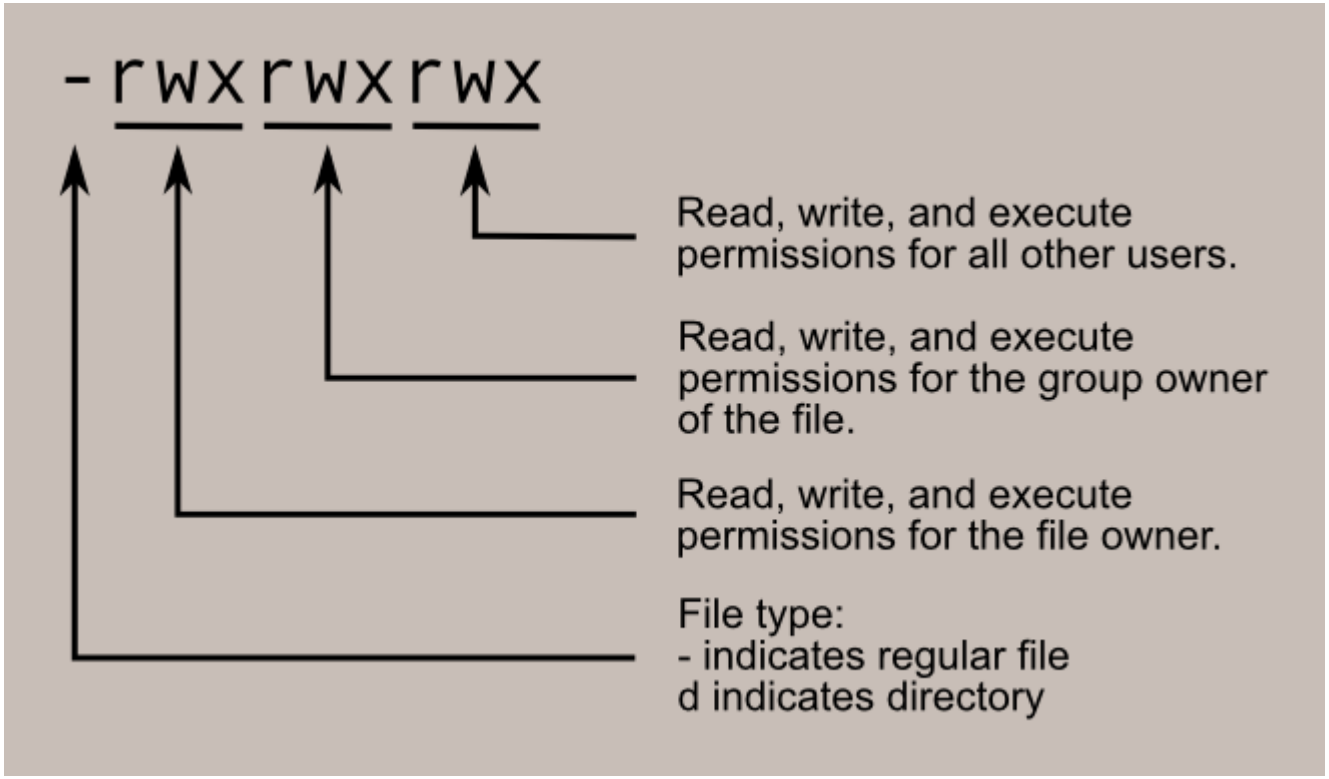
ls -l

```
drwxr-xr-x  4 dtrudgian biohpc_admin   58 Feb 16 15:13 all_training
drwxr-xr-x  7 dtrudgian biohpc_admin  140 Feb 12 10:36 Apps
drwxr-xr-x  2 dtrudgian biohpc_admin   26 Feb 16 15:12 cli_training
drwxr-xr-x  8 dtrudgian biohpc_admin 4.0K Feb 16 14:25 Cluster_Installs
drwxr-xr-x  3 dtrudgian biohpc_admin 4.0K Feb 16 11:49 Desktop
drwxr-xr-x  2 dtrudgian biohpc_admin   10 Feb 16 14:10 Documents
drwxr-xr-x  9 dtrudgian biohpc_admin  135 Feb 16 14:32 Downloads
-rw-r--r--  1 dtrudgian biohpc_admin  336 Feb 16 15:16 error.txt
drwxr-xr-x 10 dtrudgian biohpc_admin 4.0K Feb  9 12:45 Git
drwxr-xr-x 17 dtrudgian biohpc_admin 4.0K Feb 16 15:17 ownCloud
drwxr-xr-x  2 dtrudgian biohpc_admin   10 Feb 16 14:18 Pictures
drwxr-xr-x  5 dtrudgian biohpc_admin  102 Feb  4 11:19 portal_jobs
```



Permissions Owner Group

File Permissions



Octal Permissions

r	=	4
w	=	2
x	=	1

Add up the permissions you need for each class, e.g.

rx = 5

rw = 6

rwx = 7

```
-rw-r--r--  1 dtrudgian biohpc_admin  336 Feb 16 15:16 error.txt
```

6 4 4

Owner can read+write

Group can read

Others can read

Changing Permissions

Command	What does it do?
`chown dtrudgian error.txt`	Change the owner of error.txt to dtrudgian
`chgrp CellBiology error.txt`	Change the group owner of error.txt to danuser
`chmod 660 error.txt`	Set permissions 660 for error.txt Owner read/write, Group read/write, Others no access.
`chown dtrudgian.biohpcadmin cli_training`	Change ownership of the cli-training directory to dtrudgian, and group to biohpcadmin (in one step!)
`chown –R biohpcadmin cli_training`	Change the group of cli-training to biohpcadmin *recursively (directory and everything inside it).*
`chmod –R 755 cli_training`	Change permissions of cli-training and everything inside it to 755 Owner read/write/execute, group read/execute, others read/execute

Easier chmod!

```
chmod   u/g/a   +/-    r/w/x   filename
```

Class:

- u = user (owner)
- g = group
- a = all

+	Add permission
-	Remove permission

r	read
w	write
x	execute

```
chmod g+rw test.txt
chmod a+x script.sh
chmod g-x script.sh
```

Add read/write permissions for the group
Add execute permission for everyone
Remove execute permission for the group

Editors

vi / vim

Cryptic commands! Cheat sheet on the portal.

Quick tutorial: http://www.washington.edu/computing/unix/vi.html

nano

Easier to use.

Quick tutorial: http://mintaka.sdsu.edu/reu/nano.html

Manipulating text files 1

Command	What does it do?
`cat story.txt`	Show the content of story.txt in the terminal
`less story.txt`	Show the content of story.txt in a pager that allows scrolling & searching
`head –n 15 story.txt`	Show the first 15 lines of story.txt
`tail –n 5 story.txt`	Show the last 5 lines of story.txt
`head –n 15 story.txt | tail –n 5`	Shows lines 11-15 of story.txt. We take the first 15 lines from story.txt using head, and extract the bottom of that selection by piping it through tail.
`grep "elephant" story.txt`	Find all lines in story.txt that contain “elephant” *case-sensitive.*
`grep –i "mouse" story.txt`	Find all lines in story.txt that contain “cat” *non-case sensitive.*
`grep –c "at" story.txt`	Count the number of lines in story.txt that contain “at”.
`grep “^The” story.txt`	Find lines beginning with “In”
`grep “water\.$” story.txt`	Find lines ending with “water”. Note that ‘.’ is a special character so we have to *escape* it.

grep searches for patterns using *regular expressions* - http://www.robelle.com/smugbook/regexpr.html

Manipulating text files 2

Command	What does it do?
`sort gifts.txt`	Sort the lines in a file alphabetically
`sort -r gifts.txt`	Sort the lines in a file in reverse alphabetical order
`sort –nr numbers.txt`	Sort the lines in a file in reverse numerical order
`uniq repeats.txt`	Find al of the unique lines in a file
`sed "s/dog/cat/g" story.txt`	Change all instances of cat to dog, printing the result s(substitute)/old/new/g(lobal)

sed can do a lot - http://www.grymoire.com/Unix/sed.html

awk can do even more - http://www.grymoire.com/Unix/Awk.html

Redirection & Pipes

Command	What does it do?
`date > date.txt`	Send the output of the date command into date.txt **overwriting** existing content.
`date >> date.txt`	Send the output of the date command into date.txt, **appending** to the end of the file.
`which nonsense 2> error.txt`	Send the error output of the command 'which nonsense' into error.txt
`which nonsense 2> /dev/null`	Discard the error output of the command 'which nonsense'
`which nonsense 2>&1 > output.txt`	Combine the error output into the standard output, and direct into the file output.txt
`matlab < hello.m`	Send the text in hello.m to the matlab command as input
`last | less`	*Pipe* the output of the last command to the less command as input

Archives

Command	What does it do?
`gunzip mydata.txt.gz`	Extract a .gz gzipped file
`gzip mydata.txt`	Compress a file using gzip – becomes a .gz file
`tar xvf archive.tar`	Extract (x) everything from a tar archive file (f). Show verbose output (v).
`tar cvf new.tar file1 file2 folder1`	Create (c) a tar archive file (f) adding files and folders to it.
`tar zxvf archive.tar.gz`	Extract a compressed gzipped tar archive in one step
`tar jxvf archive.tar.bz2`	Extract a compressed bzipped tar archive in one step
`unzip archive.zip`	Extract a compressed ZIP archive.

My First Shell Script — script1.sh

You already wrote some, all SLURM job scripts are shell scripts – but….

Save as script1.sh

```
#!/bin/bash
clear
echo "Hello $USER"
echo –n "Today is: "
date
echo –n "Number of users logged in: "
who | wc -l
echo "Calendar"
cal
exit 0
```

Use bash to execute this script

Exit with success code

Run it: bash `myscript.sh` *-or-* `chmod +x myscript.sh` `./myscript.sh`

A variable holds information to be used later

Set them using:	`name=value`
Get their value using:	`$name`
Make them available to other programs:	`export name`

```
#!/bin/bash

MY_NAME=dave
echo Hello $MY_NAME

export MY_NAME
printenv | sort
```


Set the variable MY_NAME


Run echo using the value in MY_NAME


Put MY_NAME into our *environment*


Show our *environment variables*

We can assign the output of programs/commands into variables:

```
#!/bin/bash

NOW=$(date +%Y-%m-%d)

DATA_DIR="data_$NOW"
mkdir $DATA_DIR

module load matlab
matlab –nodisplay -nosplash < hello.m > $DATA_DIR/output.txt
```

Put the date (YY-MM-DD) into NOW

Create a directory with a name incorporating the date

Run matlab with input hello.m, and send the output into our data directory

```
#!/bin/bash

echo "This scripts checks for the dummy file."
echo "Checking...“

if [ -f "dummy" ]; then
    echo "dummy exists."
else
    echo "Couldn’t find it!”
fi
```

file exists test is -f

Read about the available tests here

http://tldp.org/LDP/Bash-Beginners-Guide/html/sect_07_01.html

This script lists all of the files in the current directory using a for loop:

```
#!/bin/bash

for i in $( ls ); do

    echo "item: $i"

done
```

Set variable i to each value in the output of the ls command

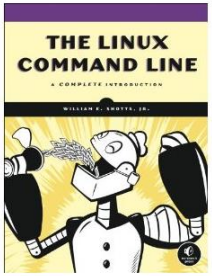
Echo the current value of $i

Read about the other loops here

http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO-7.html

Learning Shell Scripting

- Try to automate the next repetitive manual task you have to do.
- Invest some time in learning the shell to save a lot of time in future.
- Email biohpc-help@utsouthwestern.edu for advice if you're stuck.



Reminder – this book is a free PDF available from the portal website.