Introduction to UNIX Basic Commands

CS2600 Systems Programming

Dr. Amar Raheja

Lecture 4

Metacharacters

Symbol	Meaning
>	Output redirection, (see File Redirection)
>>	Output redirection (append)
<	Input redirection
*	File substitution wildcard; zero or more characters
?	File substitution wildcard; one character
[ ]	File substitution wildcard; any character between brackets
`cmd`	Command Substitution
$(cmd)	Command Substitution
|	The Pipe (|)
;	Command sequence, Sequences of Commands
||	OR conditional execution
&&	AND conditional execution
( )	Group commands, Sequences of Commands
&	Run command in the background, Background Processes
#	Comment
$	Expand the value of a variable
\	Prevent or escape interpretation of the next character
<<	Input redirection (see Here Documents)

Output Direction Metacharacter

> output redirection: overwrites what is in a file if it exists

$ echo hi > file .... Stores output of echo in *file*

$ cat > foo

hello, how are you

^D //end of file character

>> redirection: appends the output to file specified

```
[Amars-MacBook-Pro:~] amarraheja% cat foo.txt
Mon Feb  6 14:10:31 PST 2017
14:10:31
[Amars-MacBook-Pro:~] amarraheja% date -u > foo1.txt
[Amars-MacBook-Pro:~] amarraheja% cat foo1.txt >> foo.txt
[Amars-MacBook-Pro:~] amarraheja% cat foo.txt
Mon Feb  6 14:10:31 PST 2017
14:10:31
Thu May 14 17:14:11 UTC 2020
[Amars-MacBook-Pro:~] amarraheja%
```

Output Direction

Input redirection Metacharacter

< input redirection reads from standard input

```
$ cat < foo.txt
$ mail joe < foo.txt
```

File Substitution Metacharacters

File substitution wildcards

*	matches 0 or more characters
?	matches any single character
[…]	matches any character between brackets

$ls –l foo*

#list all files that begin with the word foo followed by anything else

$ ls –l foo?

#list all files that begin with the word foo followed by any single character

$ ls –l foo[1-3]

#list all files that begin with the word foo followed by any single character between number 1 and 3 (inclusive)

$ ls –l foo[23]

#list all files that begin with the word foo followed by any single character which is either 2 or 3

$ ls [!f-z]???

#list all files that begin with the characters a through e and followed by any three characters

Command Substitution

`*command*` replaced by the output of *command* run at the prompt

`` is called the grave accents

$echo `date`

```
[[Amars-MacBook-Pro:~] amarraheja% echo date
date
[[Amars-MacBook-Pro:~] amarraheja% echo `date`
Thu May 14 10:41:30 PDT 2020
[Amars-MacBook-Pro:~] amarraheja%
```

$ echo `date` > foo

```
[Amars-MacBook-Pro:CS2600 amarraheja$ more foo
[Amars-MacBook-Pro:CS2600 amarraheja$ echo `date` >foo
[Amars-MacBook-Pro:CS2600 amarraheja$ more foo
Thu May 14 15:08:51 PDT 2020
Amars-MacBook-Pro:CS2600 amarraheja$
```

How to Avoid Shell Interpretation

- Sometimes we need to pass metacharacters to the command being run and do not want the shell to interpret them. Three options to avoid shell interpretation of metacharacters.
  - Escape the metacharacter with a backslash (\). (See also Escaped Characters) Escaping characters can be inconvenient to use when the command line contains several metacharacters that need to be escaped.
  - Use single quotes (' ') around a string. Single quotes protect all characters except the backslash (\).
  - Use double quotes (" "). Double quotes protect all characters except the backslash (\), dollar sign (**$**) and grave accent (`).
- Double quotes is often the easiest to use because we often want environment variables to be expanded.

Protection of Metacharacters

- Escape examples

  $echo 5\>3

  $echo I am printing a \$ sign or two \$\$ signs

- Another example

  $echo the amount is \$5\.30

```
[Amars-MacBook-Pro:~] amarraheja% echo 5\>3
5>3
[Amars-MacBook-Pro:~] amarraheja% echo I am printing a \$ sign or two \$\$ signs
I am printing a $ sign or two $$ signs
[Amars-MacBook-Pro:~] amarraheja%
```

Single and Double Quotes

- Single and double quotes protect each other

  $ echo 'Hi "Intro to Unix" Class' Hi "Intro to Unix" Class

  $ echo "Hi 'Intro to Unix' Class" Hi 'Intro to Unix' Class

```
[Amars-MacBook-Pro:~] amarraheja% echo 'Hi "Intro to Unix" Class' Hi "Intro to Unix" Class
Hi "Intro to Unix" Class Hi Intro to Unix Class
[Amars-MacBook-Pro:~] amarraheja% echo "Hi 'Intro to Unix' Class" Hi 'Intro to Unix' Class
Hi 'Intro to Unix' Class Hi Intro to Unix Class
[Amars-MacBook-Pro:~] amarraheja%
```

$ echo 5>3 ; more 3

$echo 5\>3

5>3

$ echo '5\>3’

5\>3

Expand the value of a Variable

- Environmental Variables using *env*
- Caution: Case sensitivity

$echo $SHELL

/bin/bash

$echo $PATH

/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin

$file_to_kill=“foo1 foo2”

$echo $file_to_kill

foo1 foo2

```
Amars-MacBook-Pro:CS2600 amarraheja$ env
TERM_PROGRAM=Apple_Terminal
SHELL=/bin/bash
TERM=xterm-256color
TMPDIR=/var/folders/jm/3wyk7ct96bgcfz_zwsd59m9w0000gr/T/
Apple_PubSub_Socket_Render=/private/tmp/com.apple.launchd.Y2748WWGVt/Render
TERM_PROGRAM_VERSION=421.2
TERM_SESSION_ID=580AA94A-4CA2-4245-977B-1A71EB55C850
USER=amarraheja
SSH_AUTH_SOCK=/private/tmp/com.apple.launchd.dkWeOf1jUt/Listeners
PATH=/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin
PWD=/Users/amarraheja/Documents/junk/CS2600
LANG=en_US.UTF-8
XPC_FLAGS=0x0
XPC_SERVICE_NAME=0
SHLVL=1
HOME=/Users/amarraheja
LOGNAME=amarraheja
SECURITYSESSIONID=186ab
_=/usr/bin/env
OLDPWD=/Users/amarraheja/Documents
Amars-MacBook-Pro:CS2600 amarraheja$
```

What happens when if you try file-to-kill ?
What happens if you try not to put quotes around foo1 and foo2?
Try it

Unexpected special characters in filenames

- What happens when someone puts a **star** (i.e. **asterisk**) into a filename? You're aware of what happens when you do rm * – the **star** acts as a wildcard, grabbing *every* file and deleting it

```
Amars-MacBook-Pro:CS2600 amarraheja$ filename="* foo1 foo2 LOL"
Amars-MacBook-Pro:CS2600 amarraheja$ touch "$filename"
Amars-MacBook-Pro:CS2600 amarraheja$ ls
* foo1 foo2 LOL
Amars-MacBook-Pro:CS2600 amarraheja$
```

- Notice how rm "$filename" affects *only* the file that is named, * foo1 foo2 LOL.

```
Amars-MacBook-Pro:CS2600 amarraheja$ touch $(seq 1 5)
Amars-MacBook-Pro:CS2600 amarraheja$ ls
* foo1 foo2 LOL 2                4
1                3                5
Amars-MacBook-Pro:CS2600 amarraheja$ rm "$filename"
Amars-MacBook-Pro:CS2600 amarraheja$ ls
1       2       3       4       5
Amars-MacBook-Pro:CS2600 amarraheja$ rm $filename
rm: foo1: No such file or directory
rm: foo2: No such file or directory
rm: LOL: No such file or directory
Amars-MacBook-Pro:CS2600 amarraheja$ ls
Amars-MacBook-Pro:CS2600 amarraheja$
```

- So the main takeaway here is: **double-quote your variable references whenever possible**.

Background vs. Foreground process

- All processes started at the prompt run in foreground and the prompt is not available till the process is complete.

$ find ~ –name foo –print #what is find?

find: /Users/amarraheja/Library/Caches/CloudKit/com.apple.Safari: Operation not permitted

find: /Users/amarraheja/Library/Caches/com.apple.Safari: Operation not permitted

/Users/amarraheja/foo

/Users/amarraheja/Documents/junk/CS2600/foo

/Users/amarraheja/Documents/junk/foo

/Users/amarraheja/Documents/courses/260/foo

- How to use prompt and start a process in background?

Shifting from bg to fg and vice versa

```
$ find ~ -name foo > foo2 &
[1] 75837
$ ls ; pwd; date
[1]+  Done        find ~ -name foo > foo2
$
```

```
Amars-MacBook-Pro:CS2600 amarraheja$ sleep 10
^Z
[1]+  Stopped                 sleep 10
Amars-MacBook-Pro:CS2600 amarraheja$ ps
  PID TTY           TIME CMD
68468 ttys000    0:00.26 -bash
77502 ttys000    0:00.00 sleep 10
Amars-MacBook-Pro:CS2600 amarraheja$
```

Example of bring a background process to foreground using fg

```
$ sleep 5 &
[1]      75944
$ fg
sleep 5
```

command	description
&	(special character) when placed at the end of a command, runs that command in the background
^Z	(hotkey) suspends the currently running process
fg , bg	resumes the currently suspended process in either the foreground or background

Sequence of Commands

- Two ways: using ; or ()

$ date; pwd ; ls

```
[Amars-MacBook-Pro:~] amarraheja% date; pwd;ls
Thu May 14 10:46:12 PDT 2020
/Users/amarraheja
Applications                    OneDrive - Cal Poly Pomona
Desktop                         Pictures
Documents                       Public
Downloads                       Sites
Dropbox                         dwhelper
Google Drive                    foo
Library                         foo.txt
Movies                          foo1
Music                           foo1.txt
[Amars-MacBook-Pro:~] amarraheja%
```

- (date; pwd; ls) > out.txt

```
[Amars-MacBook-Pro:CS2600 amarraheja$ (date; pwd; ls) > foo.txt
[Amars-MacBook-Pro:CS2600 amarraheja$ ls
foo      foo.txt foo2
[Amars-MacBook-Pro:CS2600 amarraheja$ cat foo.txt
Thu May 14 15:05:50 PDT 2020
/Users/amarraheja/Documents/junk/CS2600
foo
foo.txt
foo2
Amars-MacBook-Pro:CS2600 amarraheja$
```

Conditional Execution Metacharacters

- The following two examples use the Exit code of the first command to conditionally execute the second command. A logical **AND** operator, **&&**, executes the second command only if the first command completed successfully.

$ gcc hello.c && ./a.out

```
Amars-MacBook-Pro:CS2600 amarraheja$ gcc hello.c && ./a.out
Hello, World!
Amars-MacBook-Pro:CS2600 amarraheja$
```

$ gcc hello.c || echo compilation failed

```
Amars-MacBook-Pro:CS2600 amarraheja$ gcc hello.c || echo compilation failed
Amars-MacBook-Pro:CS2600 amarraheja$ gcc hello.c || echo compilation failed
hello.c:4:10: error: expected expression
  printf(%s, "Hello, World!\n") ;
         ^
hello.c:4:11: error: use of undeclared identifier 's'
  printf(%s, "Hello, World!\n") ;
          ^
2 errors generated.
compilation failed
Amars-MacBook-Pro:CS2600 amarraheja$
```

Other Useful Commands (1 of 3)

$ cd .. # change directory to one directory level higher

$ cd . # changes to current working directory, so no change

$ file * # determines and prints file type

```
Amars-MacBook-Pro:CS2600 amarraheja$ file *
a.out:     Mach-O 64-bit executable x86_64
foo:       ASCII text
foo.txt:   ASCII text
foo2:      ASCII text
hello.c:   c program text, ASCII text
hello.c~:  c program text, ASCII text
Amars-MacBook-Pro:CS2600 amarraheja$
```

Other Useful Commands (2 of 3)

$ du #display disk usage statistics

64 . #current directory . using 64 bytes

$ df #display free disk space

```
Amars-MacBook-Pro:CS2600 amarraheja$ df
Filesystem    512-blocks      Used Available Capacity iused               ifree %iused  Mounted on
/dev/disk1s1   976490576 742296200 182622096    81% 1771083 9223372036853004724    0%   /
devfs                371       371         0   100%     642                   0  100%   /dev
/dev/disk1s4   976490576  50166264 182622096    22%      23 9223372036854775784    0%   /private/var/vm
map -hosts             0         0         0   100%       0                   0  100%   /net
map auto_home          0         0         0   100%       0                   0  100%   /home
map -fstab             0         0         0   100%       0                   0  100%   /Network/Servers
Amars-MacBook-Pro:CS2600 amarraheja$
```

Other Useful Commands (3 of 3)

- Word count using wc
  - The order of output always takes the form of line, word, byte, and file name. The default action is equivalent to specifying the -c, -l and -w options.

```
Amars-MacBook-Pro:CS2600 amarraheja$ cat foo2
/Users/amarraheja/foo
/Users/amarraheja/Documents/junk/CS2600/foo
/Users/amarraheja/Documents/junk/foo
/Users/amarraheja/Documents/courses/260/foo
Amars-MacBook-Pro:CS2600 amarraheja$ wc foo2
       4       4     147 foo2
Amars-MacBook-Pro:CS2600 amarraheja$ wc -c foo2; wc -l foo2; wc -w foo2
     147 foo2
       4 foo2
       4 foo2
Amars-MacBook-Pro:CS2600 amarraheja$
```

Unix system status commands

$ uname # print name of operating system

Darwin

$ w

$ who

```
[raheja@abbott 513 ~]$ w
 16:55:40 up 190 days,  3:13,  5 users,  load average: 0.00, 0.00, 0.00
USER     TTY        LOGIN@   IDLE   JCPU   PCPU WHAT
zalindsa pts/0     15:22   20.00s  0.70s  0.01s nano RationalApp.cpp
gapierce pts/1     29Apr20  3:49m  0.01s  0.00s tmux attach -t 0
bmwallac pts/4     16:09   20:27   0.01s  0.01s nano
bimoya   pts/5     13:52    4.00s  0.46s  0.37s nano DirectedGraph.java
raheja   pts/6     16:53    0.00s  0.02s  0.00s w
[raheja@abbott 514 ~]$ who
zalindsay pts/0        2020-05-14 15:22 (172.88.217.1)
gapierce pts/1        2020-04-29 08:00 (134.71.250.50)
bmwallace pts/4        2020-05-14 16:09 (71.93.129.0)
bimoya   pts/5        2020-05-14 13:52 (10.104.198.118)
raheja   pts/6        2020-05-14 16:53 (172.249.37.82)
[raheja@abbott 515 ~]$
```

$who am i

raheja pts/6 2020-05-14 16:53 (174.249.38.89)

$ hostname

```
[raheja@abbott 510 ~]$ hostname
abbott
[raheja@abbott 511 ~]$ hostname -i
2620:df:8000:ff14:0:1:246:203
[raheja@abbott 512 ~]$ hostname -d
unx.cpp.edu
[raheja@abbott 513 ~]$
```

User id command

id displays the user id and all group names and ids

[raheja@abbott 516 ~] $ id

uid=24633(raheja) gid=1012(cpp) groups=1012(cpp),1119(cfa),1422(faculty),2607(cs_faculty),3767(sci_faculty),13142(coe_cos_team),17984(polyquanta-admin),17985(polyquanta),18475(senate_sci),20795(campus_techs),21053(acad_advisors),23358(employees),23359(members),26872(pending_instructors),27371(tenure_track_faculty),27677(unit03_employees),29184(cs_tenure_track_faculty),29185(cs_tenured_faculty),29274(sci_tenure_track_faculty),29275(sci_tenured_faculty),29276(tenured_faculty)

Simple UNIX trick

- Unix tricks is ^x^y, which will take the last command and replace the first instance of "x" with "y"

```
Amars-MacBook-Pro:CS2600 amarraheja$ dare
-bash: dare: command not found
Amars-MacBook-Pro:CS2600 amarraheja$ ^r^t
date
Thu May 14 22:08:58 PDT 2020
Amars-MacBook-Pro:CS2600 amarraheja$ ls -af foo
foo
Amars-MacBook-Pro:CS2600 amarraheja$ ^f^l
ls -al foo
-rw-r--r--  1 amarraheja  staff  29 May 14 15:08 foo
Amars-MacBook-Pro:CS2600 amarraheja$
```

```
Amars-MacBook-Pro:CS2600 amarraheja$ echo "dog cat dog"
dog cat dog
Amars-MacBook-Pro:CS2600 amarraheja$ ^dog^cat^:&
echo "cat cat cat"
cat cat cat
Amars-MacBook-Pro:CS2600 amarraheja$ echo "dog cat dog dog"
dog cat dog dog
Amars-MacBook-Pro:CS2600 amarraheja$ ^dog^cat^:&
echo "cat cat cat dog"
cat cat cat dog
Amars-MacBook-Pro:CS2600 amarraheja$ echo "dog cat dog dog"
dog cat dog dog
Amars-MacBook-Pro:CS2600 amarraheja$ ^dog^cat^:&:&
echo "cat cat cat cat"
cat cat cat cat
Amars-MacBook-Pro:CS2600 amarraheja$
```

Auto Completion in Shells

- Most shells can complete a filename, command name, username or shell variable that you have begun to type if enough has been typed to uniquely identify it.
- While typing on the command line, press *TAB* to see if BASH can complete the word you are typing. If it can not identify a completion, nothing appears on the command line, but by pressing *TAB* a second time, a list of possible matches is displayed.

```
Amars-MacBook-Pro:CS2600 amarraheja$ ls foo
foo       foo.txt  foo2
Amars-MacBook-Pro:CS2600 amarraheja$ ls foo
```

How to find if an executable/command exists?

which command that looks for executable in your PATH

```
Amars-MacBook-Pro:CS2600 amarraheja$ pwd
/Users/amarraheja/Documents/junk/CS2600
Amars-MacBook-Pro:CS2600 amarraheja$ ls
a.out           foo             foo.txt         foo2            hello.c         hello.c~
Amars-MacBook-Pro:CS2600 amarraheja$ which ls
/bin/ls
Amars-MacBook-Pro:CS2600 amarraheja$ which gcc
/usr/bin/gcc
Amars-MacBook-Pro:CS2600 amarraheja$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin
Amars-MacBook-Pro:CS2600 amarraheja$ which a.out
Amars-MacBook-Pro:CS2600 amarraheja$
```

Other Useful Commands

clear command that clears all the content visible in the terminal

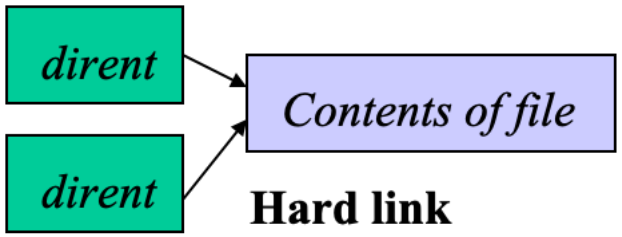
ln command to create a link

- Directories are a list of files and directories.
  - Each directory entry *links* to a file on the disk
  - Two different directory entries can link to the same file
    - In same directory or across different directories
  - Moving a file does not actually move any data around.
    - Creates link in new location
    - Deletes link in old location
  - Each file has a link count

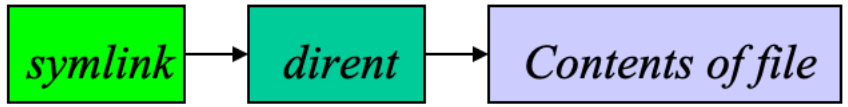
Hard vs. Soft Links

Hard Links	Soft link (symbolic link)
Target must exist	Target may already exist, but does not have to
Allowed within file systems only	Allowed between different file systems
Links directly to the place the file is stored	Links to the entry in the file system table (node)
Removing the link means removing the whole file	Removing the link means removing the link to the node, not the file itself

- Can be thought of as a directory entry that points to the ***name*** of another file.
- Does not change link count for file
  - When original deleted, symbolic link remains
- They exist because:
  - Hard links don't work across file systems
  - Hard links only work for regular files, not directories



Hard link



Symbolic Link

Soft Links and Examples

- Soft links can be created using the –s option for ln

$ ln -s /usr/bin/python link_to_python

$ ls -la link_to_python

lrwxrwxrwx 1 amarraheja faculty 15 Oct 5 14:25 link_to_python -> /usr/bin/python

- Another example on Cal Poly UNIX server

```
[raheja@abbott 522 ~]$ which sh
/bin/sh
[raheja@abbott 523 ~]$ which bash
/bin/bash
[raheja@abbott 524 ~]$ ls -l /bin/sh
lrwxrwxrwx 1 root root 4 Apr  3  2019 /bin/sh -> bash
[raheja@abbott 525 ~]$
```

Popular Commands: cal

cal calendar for any year and any month

$cal 2025 #it will print calendar for 2025

```
Amars-MacBook-Pro:CS2600 amarraheja$ cal
      May 2020
Su Mo Tu We Th Fr Sa
                   1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
Amars-MacBook-Pro:CS2600 amarraheja$ cal 8 1970
    August 1970
Su Mo Tu We Th Fr Sa
                      1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

Popular Commands: bc

bc poor man's command line calculator

bc command supports the following features:

- Arithmetic operators
- Increment or Decrement operators
- Assignment operators
- Comparison or Relational operators
- Logical or Boolean operators
- Math functions
- Conditional statements
- Iterative statements

```
Amars-MacBook-Pro:CS2600 amarraheja$ echo "12+5" | bc
17
Amars-MacBook-Pro:CS2600 amarraheja$ echo "10^2" | bc
100
Amars-MacBook-Pro:CS2600 amarraheja$ x=`echo "12+5" | bc`
Amars-MacBook-Pro:CS2600 amarraheja$ echo $x
17
Amars-MacBook-Pro:CS2600 amarraheja$ echo "1==2" | bc
0
Amars-MacBook-Pro:CS2600 amarraheja$ echo "10>5" | bc
1
Amars-MacBook-Pro:CS2600 amarraheja$
```

Popular Commands: touch

- touch updates the time stamp on a file that is touched.
  empty file created if it doesn't exist.

```
Amars-MacBook-Pro:CS2600 amarraheja$ ls -l datebook.txt
-rw-r--r--  1 amarraheja  staff  2059 May 25 22:13 datebook.txt
Amars-MacBook-Pro:CS2600 amarraheja$ touch datebook.txt
Amars-MacBook-Pro:CS2600 amarraheja$ ls -l datebook.txt
-rw-r--r--  1 amarraheja  staff  2059 May 27 15:47 datebook.txt
Amars-MacBook-Pro:CS2600 amarraheja$ ls
a.out           data3           datebook.txt    foo3            junk
data            database.txt    foo             foo4            test
data1           datafile.txt    foo.txt         hello.c         test2
data2           datafile.txt~   foo2            hello.c~        test2~
Amars-MacBook-Pro:CS2600 amarraheja$ touch newfile
Amars-MacBook-Pro:CS2600 amarraheja$ ls
a.out           data3           datebook.txt    foo3            junk            test2~
data            database.txt    foo             foo4            newfile
data1           datafile.txt    foo.txt         hello.c         test
data2           datafile.txt~   foo2            hello.c~        test2
Amars-MacBook-Pro:CS2600 amarraheja$
```

Popular Commands: tar

- tar creates a tape archive and can also compress using gzip

  -c: **C**reate an archive.

  -x: e**X**tract an archive (untar)

  -z: Compress the archive with g**z**ip.

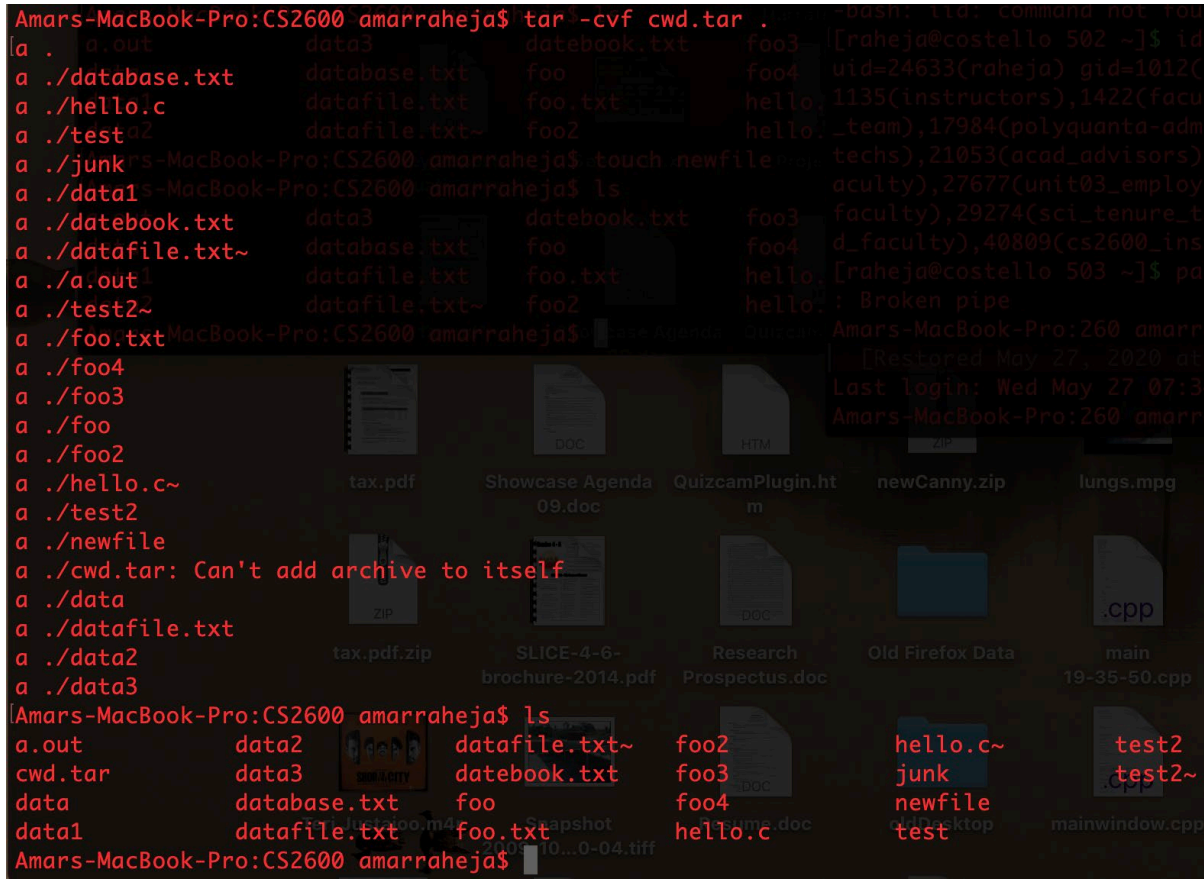
  -v: Display progress in the terminal while creating the archive, also known as "**v**erbose" mode.

  -f: Allows you to specify the **f**ilename of the archive.

Example: tar -czvf archive.tar.gz ~

How to untar or extract archive?

How to unzip?



Popular Commands: gzip and compress

- gzip Creates a zip or compressed file from the input (uses similar compression algorithm as winzip), creates .gz extension for that file
  - -d: uncompress the file
  - -r: recursively compress every file in directory
- compress Older compression utility that creates a .Z extension after file compression.
- uncompress decompresses the file compressed using compress

  removes the .Z extension
- Tape Archive and Compression are primarily used to distribute the entire software file structure in Unix