NLP Halloween* Hacks

* Also work on non-Halloween

Why do you need command line magic?

Magic for NLP

1. Preprocessing: parsing, extracting, chunking, cleaning text
2. Scheduling
3. Remote Servers

Part 1: Command Line Hacks

Pre-Reqs

```
cd mv ls pwd
mkdir touch echo
vim emacs nano
man
chmod chown
sudo su
git
```

How do you:

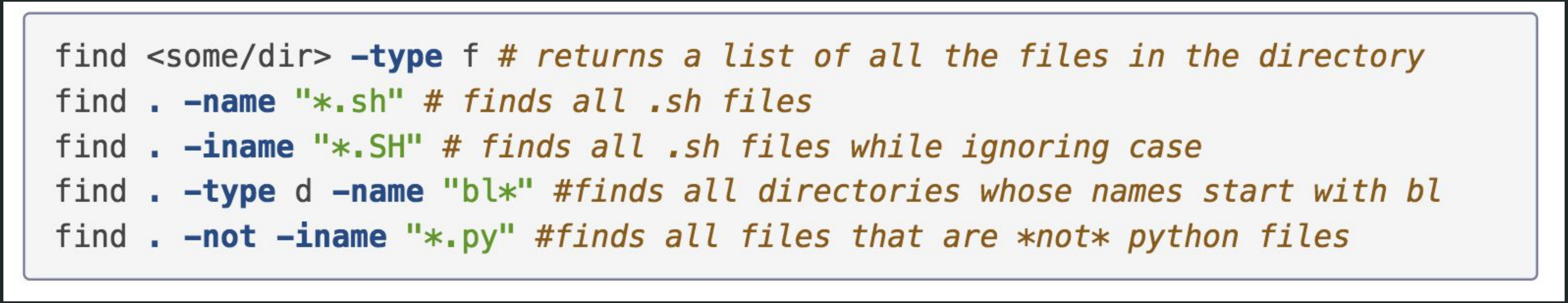
How do you:
copy a file into all subdirectories?

How do you:

copy a file into all subdirectories?

```
find . -type d -exec cp template.txt {} \;
```

Find



```
find <some/dir> -type f # returns a list of all the files in the directory
find . -name "*.sh" # finds all .sh files
find . -iname "*.SH" # finds all .sh files while ignoring case
find . -type d -name "bl*" #finds all directories whose names start with bl
find . -not -iname "*.py" #finds all files that are *not* python files
```

find + exec

How do you:

figure out if your teammate didn't do any work?

How do you:

figure out if your teammate didn't do any work?

```
diff   -y   my_file.py  their_file.py
```

How do you:

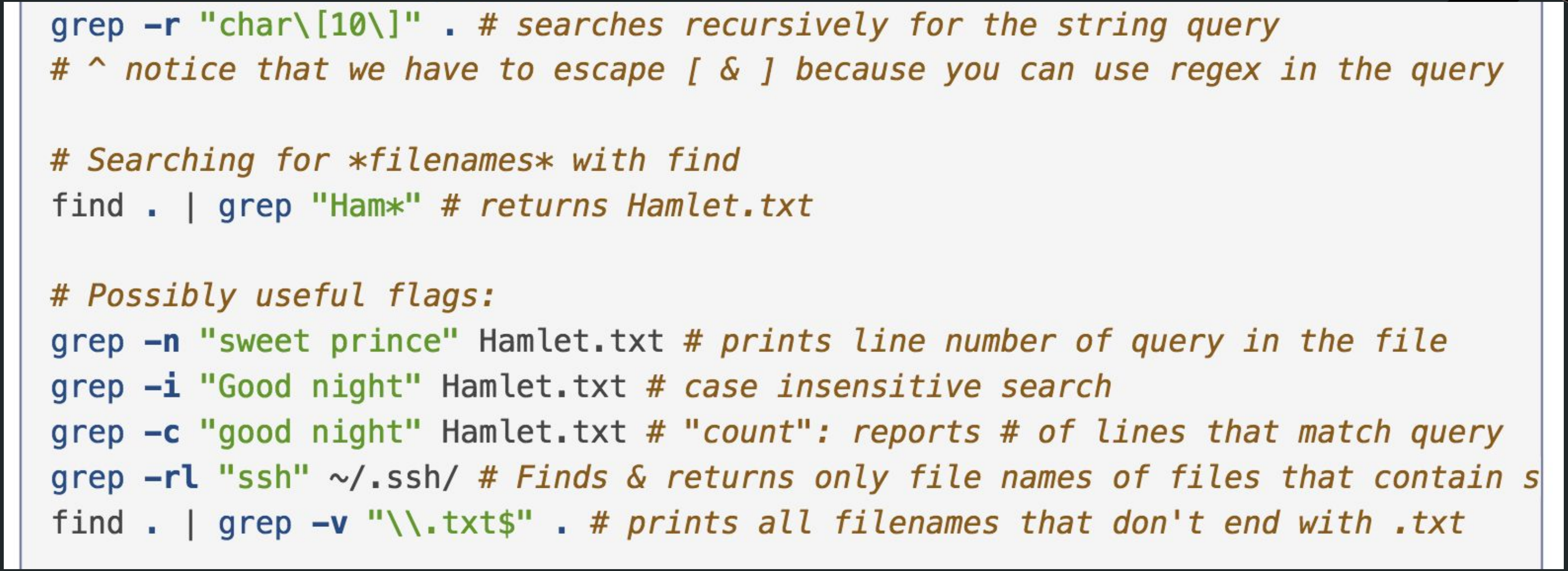
find all files that contain q1 && q2?

How do you:

find all files that contain q1 && q2?

```
grep -rl "q1" ~/.ssh/
    | xargs grep -r "q2"
```

Grep



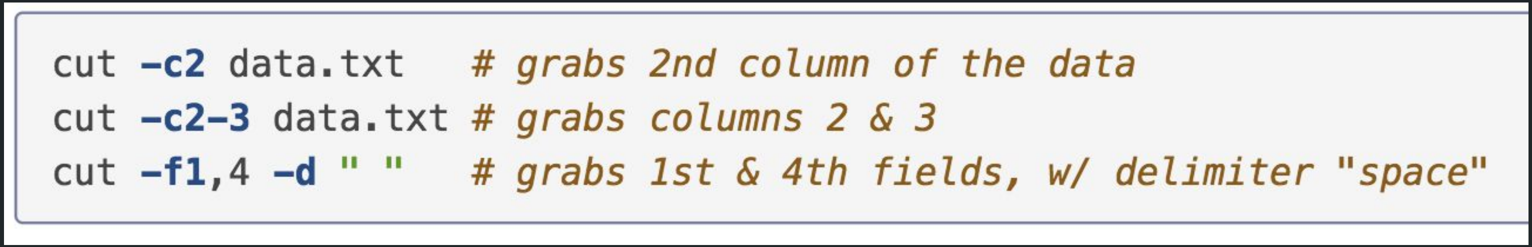
```
grep -r "char\[10\]" . # searches recursively for the string query
# ^ notice that we have to escape [ & ] because you can use regex in the query

# Searching for *filenames* with find
find . | grep "Ham*" # returns Hamlet.txt

# Possibly useful flags:
grep -n "sweet prince" Hamlet.txt # prints line number of query in the file
grep -i "Good night" Hamlet.txt # case insensitive search
grep -c "good night" Hamlet.txt # "count": reports # of lines that match query
grep -rl "ssh" ~/.ssh/ # Finds & returns only file names of files that contain s
find . | grep -v "\\.txt$" . # prints all filenames that don't end with .txt
```

grep + xargs ; find + grep

How do you:
deal with files?

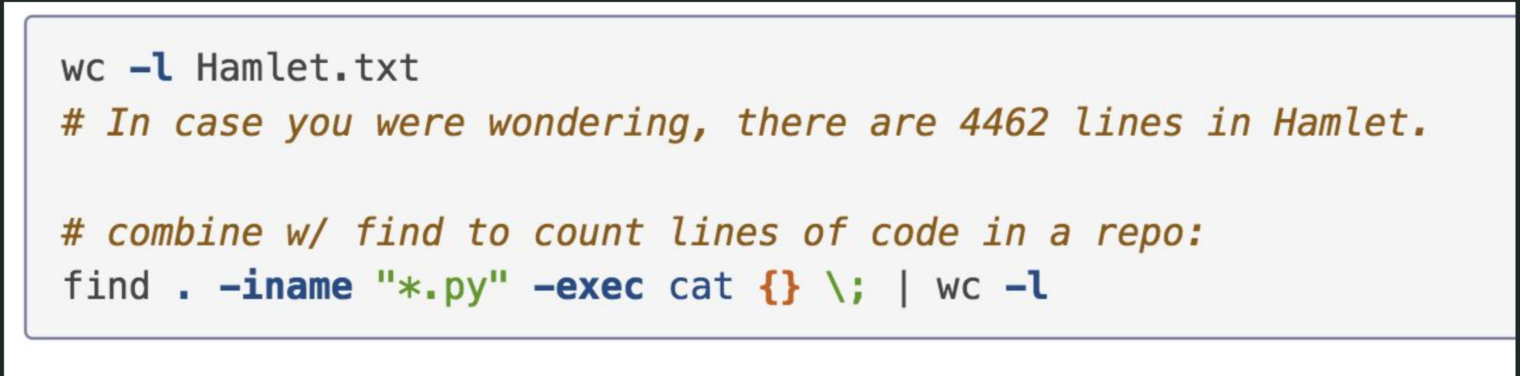
head / tail ; cat / tac
wc; sort ; shuf ; cut; uniq

Dealing with files



```
cut -c2 data.txt   # grabs 2nd column of the data
cut -c2-3 data.txt # grabs columns 2 & 3
cut -f1,4 -d " "   # grabs 1st & 4th fields, w/ delimiter "space"
```

find + wc -l



```
wc -l Hamlet.txt
# In case you were wondering, there are 4462 lines in Hamlet.

# combine w/ find to count lines of code in a repo:
find . -iname "*.py" -exec cat {} \; | wc -l
```

How many tests did I fail?

```
seq 10 | xargs -Iz ./runtests
       | tee testResults.txt

grep "fail" testResult.txt | wc -l
```

```
seq + xargs + tee
grep + wc
```

How do you:

zip all "c" files?

```
find . -name "*.[ch]"
 | zip source -@
```

unicode is hard.

unicode is hard.

😱 🎃 □ □ □ ⚡ 😱 □ 🍭 😱 □

How do you:

deal with command line options?

getopt

getopt example

http://www.kfirlavi.com/blog/2012/11/14/defensive-bash-programming/

What do you do with:

expr?

How do you:

find all 'a's and replace with 'b's

awk

sed

Part 2: Scheduling & Bash

How do you:
deal with jobs?

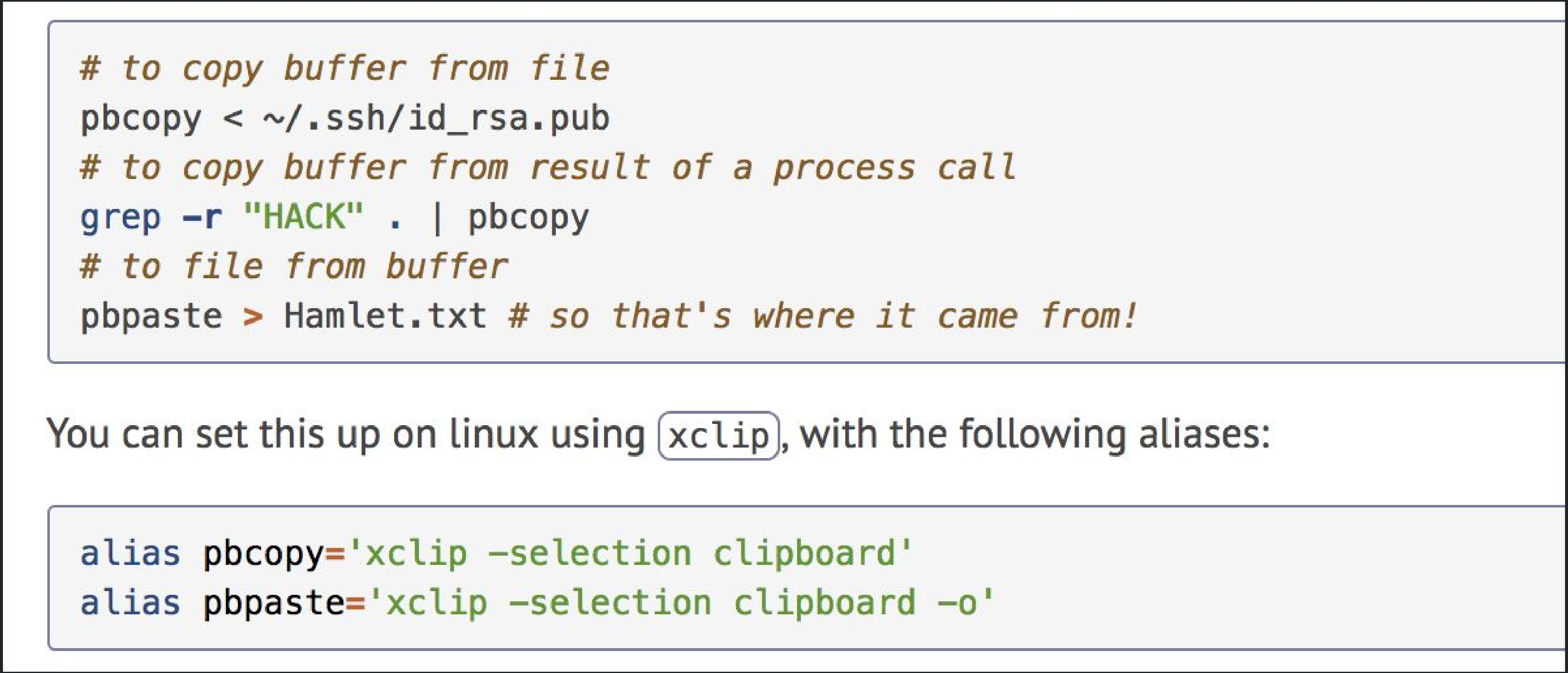
jobs	;	kill / CNTRL + C
fg	;	bg / CNTRL + Z

How do you:

dump a file to clipboard?

pbcopy / pbpaste
xclip + alias

copy & pasting



```
# to copy buffer from file
pbcopy < ~/.ssh/id_rsa.pub
# to copy buffer from result of a process call
grep -r "HACK" . | pbcopy
# to file from buffer
pbpaste > Hamlet.txt # so that's where it came from!
```

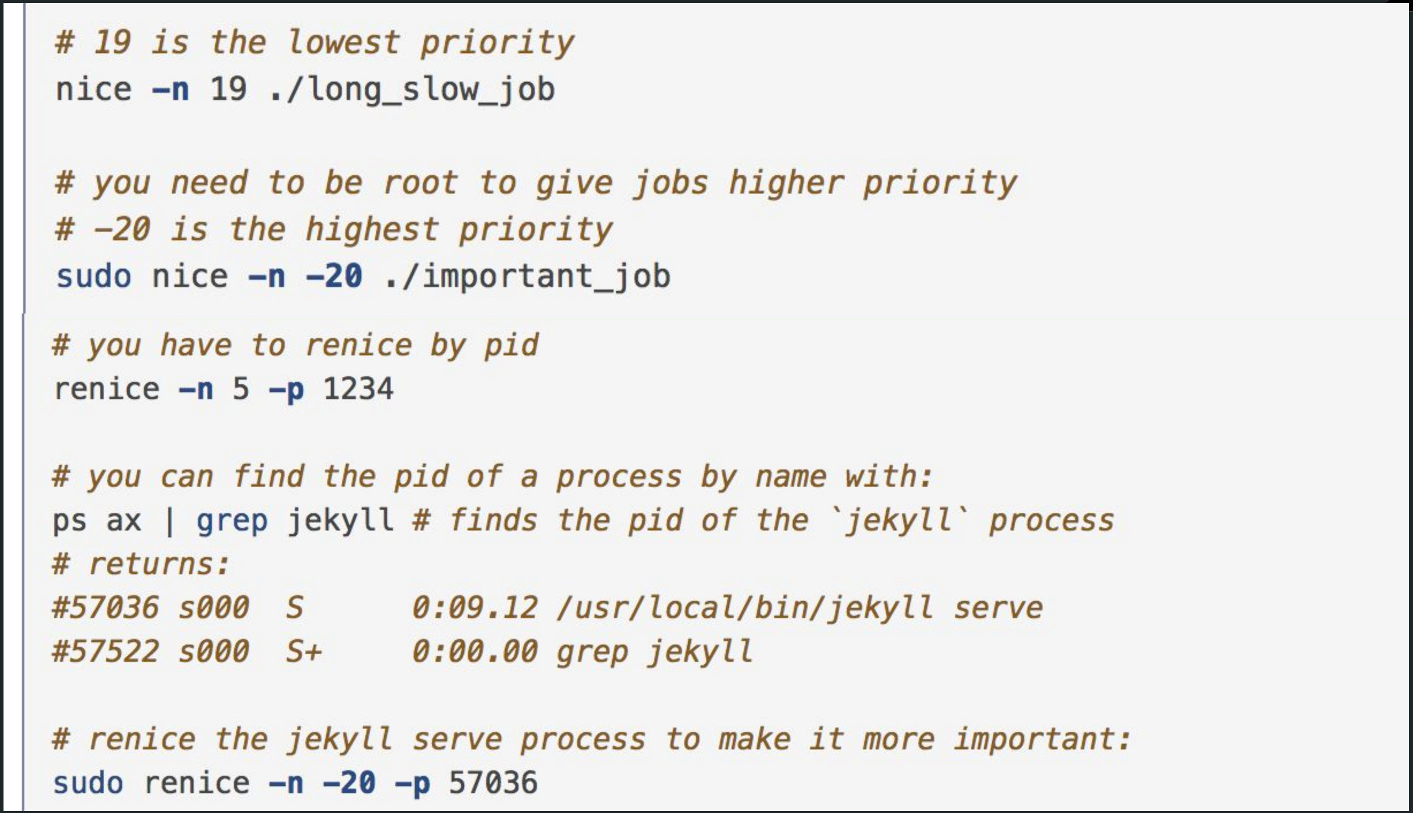
You can set this up on linux using `xclip`, with the following aliases:

```
alias pbcopy='xclip -selection clipboard'
alias pbpaste='xclip -selection clipboard -o'
```

How do you:
be a good citizen?

nice ; renice

Nice & renice



```
# 19 is the lowest priority
nice -n 19 ./long_slow_job

# you need to be root to give jobs higher priority
# -20 is the highest priority
sudo nice -n -20 ./important_job

# you have to renice by pid
renice -n 5 -p 1234

# you can find the pid of a process by name with:
ps ax | grep jekyll # finds the pid of the `jekyll` process
# returns:
#57036 s000  S      0:09.12 /usr/local/bin/jekyll serve
#57522 s000  S+     0:00.00 grep jekyll

# renice the jekyll serve process to make it more important:
sudo renice -n -20 -p 57036
```

How do you:

watch a job while also logging it?

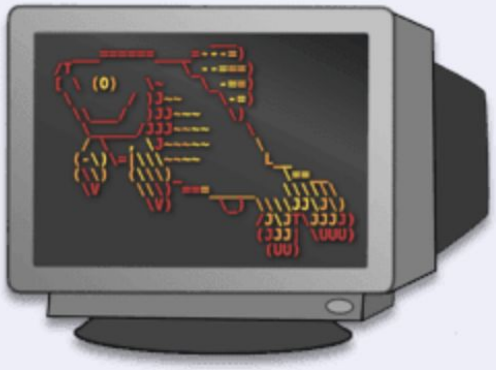
```
./run | tee "log.txt"
```

Using Bash: it's not the only shell

fish ; tcsh ; zsh ; ksh

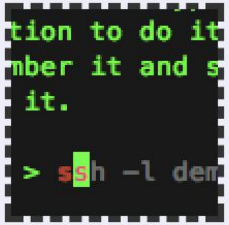
Finally, a command line shell for the 90s

fish is a smart and user-friendly command line shell for macOS, Linux, and the rest of the family.



Autosuggestions

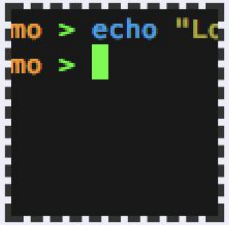
fish suggests commands as you type based on history and completions, just like a web browser. Watch out, Netscape Navigator 4.0!



```
tion to do it
mber it and s
 it.

> ssh -l dem
```

Glorious VGA Color

fish supports 24 bit true color, the state of the art in terminal technology. Behold the monospaced rainbow.



```
mo > echo "Lc
mo >
```

There's also python

Runtime gatekeeping!

Bash History

HISTSIZE= HISTFILESIZE=

not quite right

First, you must comment out or **remove this section of your .bashrc** (default for Ubuntu). If you don't, then certain environments (like running `screen` sessions) will still truncate your history:

```
# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
# HISTSIZE=1000
# HISTFILESIZE=2000
```

Second, **add this** to the bottom of your .bashrc:

```
# Eternal bash history.
# ---------------------
# Undocumented feature which sets the size to "unlimited".
# http://stackoverflow.com/questions/9457233/unlimited-bash-history
export HISTFILESIZE=
export HISTSIZE=
export HISTTIMEFORMAT="[%F %T] "
# Change the file location because certain bash sessions truncate .bash_history file
# http://superuser.com/questions/575479/bash-history-truncated-to-500-lines-on-each-
export HISTFILE=~/.bash_eternal_history
# Force prompt to write history after every command.
# http://superuser.com/questions/20900/bash-history-loss
PROMPT_COMMAND="history -a; $PROMPT_COMMAND"
```

Bash Linter

https://www.shellcheck.net/

Bash … UNIT TESTS?

shunit2

Bash Best Practices

https://jvns.ca/blog/2017/03/26
/bash-quirks/

How do you:

run hundreds of jobs, 2 at a time?

bash loop

There must be a better way

```
MAX_JOBS=2

launch_when_not_busy()
{
  while [ $(jobs | wc -l) -ge $MAX_JOBS ]
  do
      sleep 1
  done

 "$@" &
}

for i in `seq 1 100`; do
  launch_when_not_busy $command $i
done
```

Part 3: Remote Servers & Dotfiles

How do you:
get in?

ssh ; keyless ssh ; ssh aliases

How do you:

get out?

<enter> ~ .

CNTRL + D --> EOF character

How do you:
get stuff in / out?

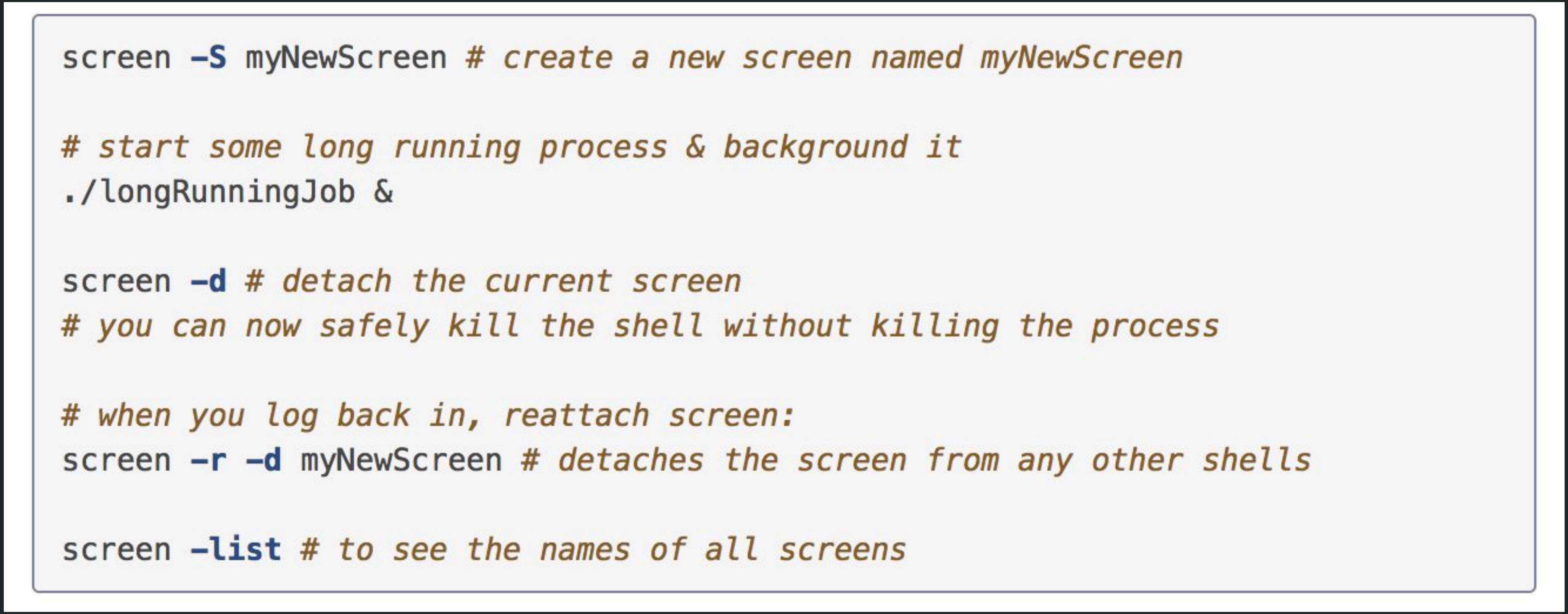
scp
wget ; curl

How do you:

keep remote jobs live?

tmux ; screen

Screen



```
screen -S myNewScreen # create a new screen named myNewScreen

# start some long running process & background it
./longRunningJob &

screen -d # detach the current screen
# you can now safely kill the shell without killing the process

# when you log back in, reattach screen:
screen -r -d myNewScreen # detaches the screen from any other shells

screen -list # to see the names of all screens
```

How do you:

find who's eating all the memory?

htop

who ; whoami

How do you:

how much are they eating?

free -g

How do you:
find what's eating all my disk?

```
ncdu
du -h  [-s]  [-d 1]
```

How do you:

go down a rabbit hole for a sec?

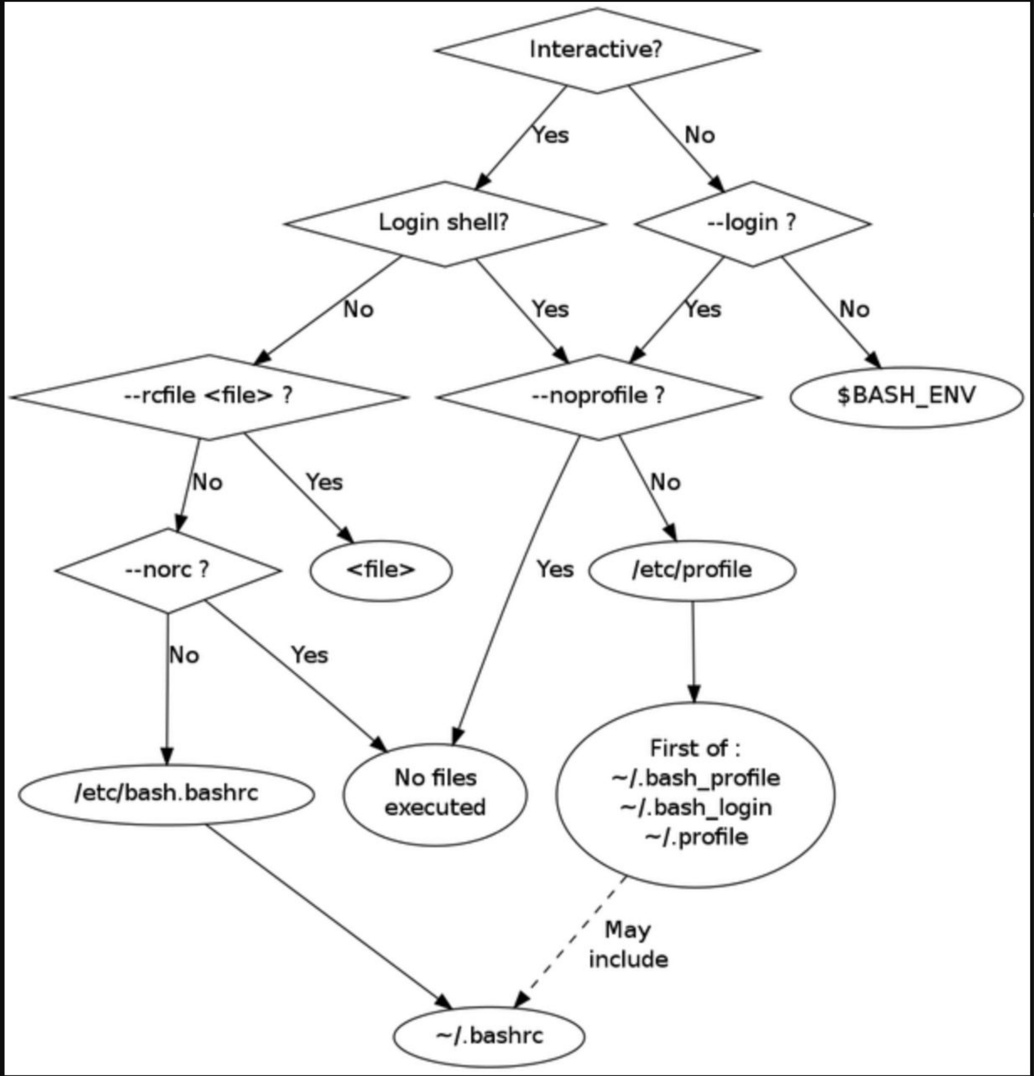
pushd / popd

How do you:

figure out if the internet is up?

```
ping www.google.com
```

.profile vs .rc?

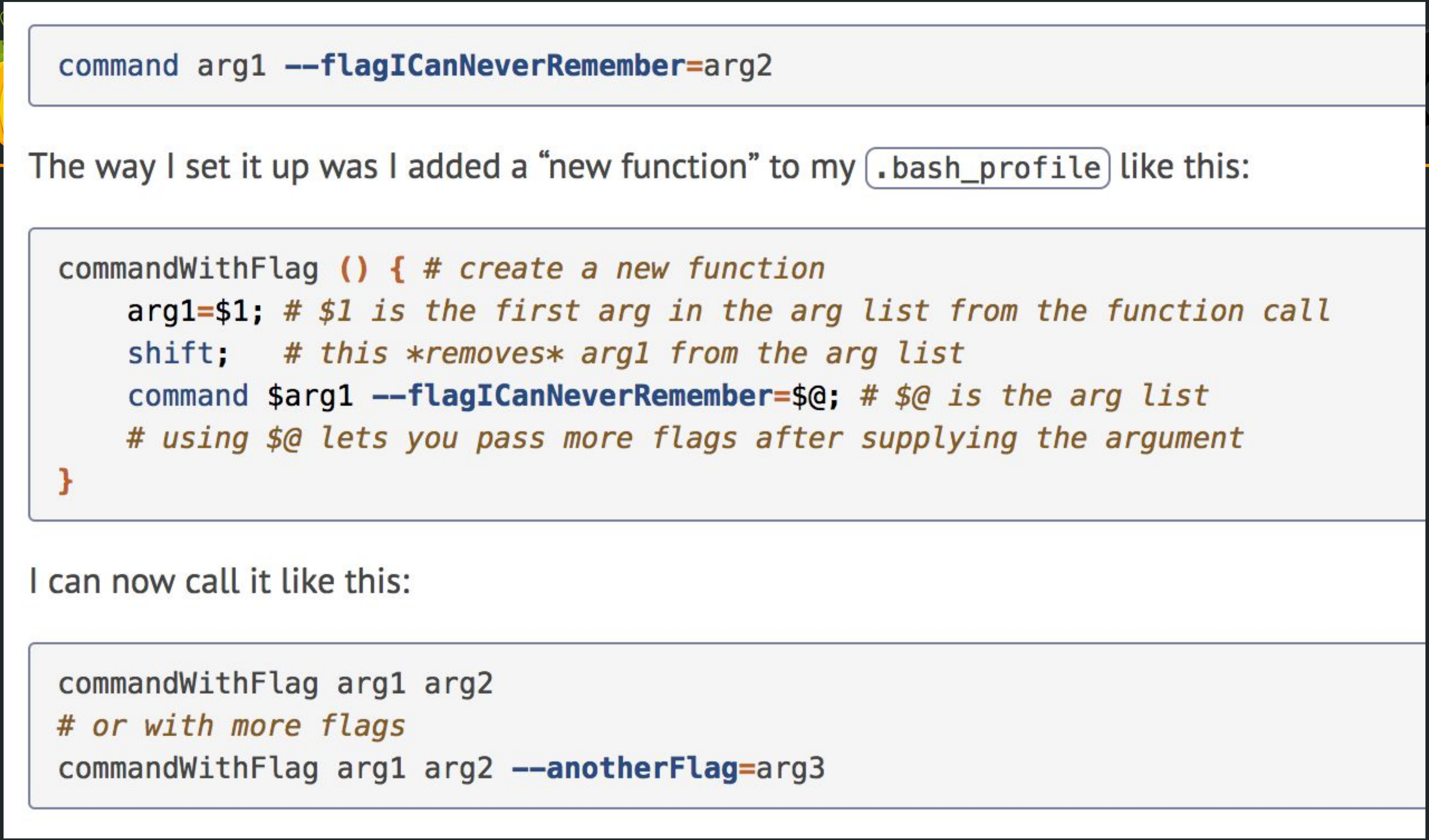

Interactive?
Yes
No
Login shell?
--login ?
No
Yes
Yes
No
--rcfile <file> ?
--noprofile ?
$BASH_ENV
No
Yes
No
<file>
--norc ?
Yes
/etc/profile
No
Yes
/etc/bash.bashrc
No files executed
First of :
~/.bash_profile
~/.bash_login
~/.profile
May include
~/.bashrc

How I live my life

- make a ~/.alias file for aliases
- make a ~/.paths file for additions to your PATH
- `source` those two files in ~/.bashrc (and the equivalent file for other shells)
- anything else can go in ~/.bashrc (and the equivalent file for other shells)
- `source` the ~/.bashrc file in ~/.bash_profile

How do you:
be lazy and/or funny?

alias
unalias



```
command arg1 --flagICanNeverRemember=arg2
```

The way I set it up was I added a "new function" to my `.bash_profile` like this:

```
commandWithFlag () { # create a new function
    arg1=$1; # $1 is the first arg in the arg list from the function call
    shift;   # this *removes* arg1 from the arg list
    command $arg1 --flagICanNeverRemember=$@; # $@ is the arg list
    # using $@ lets you pass more flags after supplying the argument
}
```

I can now call it like this:

```
commandWithFlag arg1 arg2
# or with more flags
commandWithFlag arg1 arg2 --anotherFlag=arg3
```

Dotfiles??

vcsh + myrepos
dotbot
stow

Part 4: Challenges

https://cmdchallenge.com/

http://overthewire.org/wargames/bandit/

Part 5: Resources

Resource List: Bash Best Practice

https://www.shellcheck.net/

http://www.kfirlavi.com/blog/2012/11/14/defensive-bash-programming/

https://jvns.ca/blog/2017/03/26/bash-quirks/

http://www.tldp.org/LDP/abs/abs-guide.pdf

Resource List: "Applied NLP"

http://www.cis.lmu.de/~davidk/ap/

https://github.com/nschneid/unix-text-commands

https://wordnet.princeton.edu/wordnet/man/wn.1WN.html

https://www.stanford.edu/class/cs124/kwc-unix-for-poets.pdf

Correct bash history

https://stackoverflow.com/questions/9457233/unlimited-bash-history

My command line cribsheet

```
https://anniecherkaev.com/commandline-cribsheet
```