Intro to BASH

Group 1

First Year Bootcamp, 2016

Table of Contents

What are we bashing and why?

- No violence involved, bash is a program!

What are we bashing and why?

- No violence involved, bash is a program!
- bash (the name is an acronym for Bourne-Again SHell, don't ask) is a powerful command line interpreter that is the default on most Linux distros and OS X.

What are we bashing and why?

- No violence involved, bash is a program!
- bash (the name is an acronym for Bourne-Again SHell, don't ask) is a powerful command line interpreter that is the default on most Linux distros and OS X.
- In other words, we're going back to the way people used to use computers in the old days, or the way that "hackers" use them on TV.

What are we bashing and why?

- No violence involved, bash is a program!
- bash (the name is an acronym for Bourne-Again SHell, don't ask) is a powerful command line interpreter that is the default on most Linux distros and OS X.
- In other words, we're going back to the way people used to use computers in the old days, or the way that "hackers" use them on TV.
- Why? Bash allows you to:
  - Simplify tasks that you could possibly do in other ways. Want to rename 1000 data files? Bash makes it (relatively) easy!

What are we bashing and why?

- No violence involved, bash is a program!
- bash (the name is an acronym for Bourne-Again SHell, don't ask) is a powerful command line interpreter that is the default on most Linux distros and OS X.
- In other words, we're going back to the way people used to use computers in the old days, or the way that "hackers" use them on TV.
- Why? Bash allows you to:
  - Simplify tasks that you could possibly do in other ways. Want to rename 1000 data files? Bash makes it (relatively) easy!
  - Access powerful tools like ssh and git that you otherwise couldn't.

Opening bash

- **OS X/Linux:** Open terminal, bash is default shell
- **Windows:** Open git shell, which includes bash

Opening bash

- **OS X/Linux:** Open terminal, bash is default shell
- **Windows:** Open git shell, which includes bash
- You can get a list of commands by typing *help* in the prompt and hitting enter.

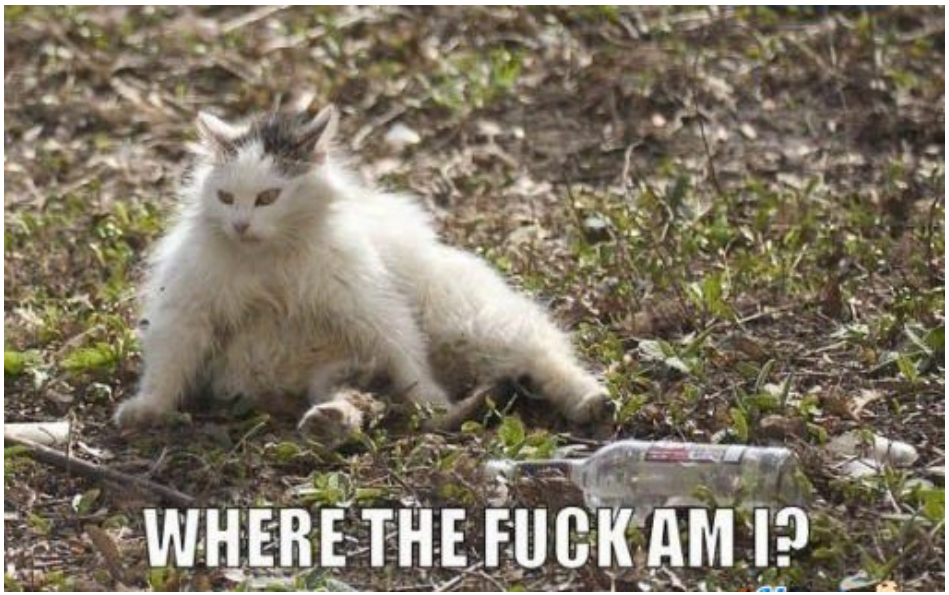
Opening bash

- **OS X/Linux:** Open terminal, bash is default shell
- **Windows:** Open git shell, which includes bash
- You can get a list of commands by typing *help* in the prompt and hitting enter.
- You can learn more about a command or program by typing *help command* or *man command*

Table of Contents

Where am I? (Directories)

- Just like when you use explorer or finder to navigate on your computer, bash sees your files as organized into directories (folders). Whenever you use bash, you're always in a directory (your working directory). To find out what directory you're in right now, try typing *pwd* (print working directory) in the prompt, and then hitting enter.



Where am I? (Directories)

- Just like when you use explorer or finder to navigate on your computer, bash sees your files as organized into directories (folders). Whenever you use bash, you're always in a directory (your working directory). To find out what directory you're in right now, try typing *pwd* (print working directory) in the prompt, and then hitting enter.
- Did it print a directory name? Good! Now try typing *ls* (list) into the prompt. This should list the directories and files that are within this directory.

Where am I? (Directories)

- Just like when you use explorer or finder to navigate on your computer, bash sees your files as organized into directories (folders). Whenever you use bash, you're always in a directory (your working directory). To find out what directory you're in right now, try typing *pwd* (print working directory) in the prompt, and then hitting enter.
- Did it print a directory name? Good! Now try typing *ls* (list) into the prompt. This should list the directories and files that are within this directory.
- You can change to a new directory by typing *cd* (change directory) followed by the directory name. For example, try *cd* $\sim$. ($\sim$ is a special character that refers to your home directory, usually */home/your-user-name/*.)

Where am I? (Directories)

- Just like when you use explorer or finder to navigate on your computer, bash sees your files as organized into directories (folders). Whenever you use bash, you're always in a directory (your working directory). To find out what directory you're in right now, try typing *pwd* (print working directory) in the prompt, and then hitting enter.
- Did it print a directory name? Good! Now try typing *ls* (list) into the prompt. This should list the directories and files that are within this directory.
- You can change to a new directory by typing *cd* (change directory) followed by the directory name. For example, try *cd* $\sim$. ($\sim$ is a special character that refers to your home directory, usually */home/your-user-name/*.)
- Now try using *ls* again, and then using *cd* to enter one of the directories you see (perhaps Documents, Desktop, Downloads or similar, depending how your OS is set up).

Arguments and flags for *ls*

- What if you want to list the contents of a directory other than your current one? In that case you can just give that directory's path as an argument to *ls*. For example: *ls /home* will tell you what's in the directory */home*, no matter where you are.

Arguments and flags for *ls*

- What if you want to list the contents of a directory other than your current one? In that case you can just give that directory's path as an argument to *ls*. For example: *ls /home* will tell you what's in the directory */home*, no matter where you are.
- There are various flags (usually a - followed by a single character, or a – followed by a word) you can pass to *ls*. For example, try *ls -l* (more information every item on its own line), *ls -a* (lists hidden files as well), or *ls –group-directories-first* (self explanatory).

Arguments and flags for *ls*

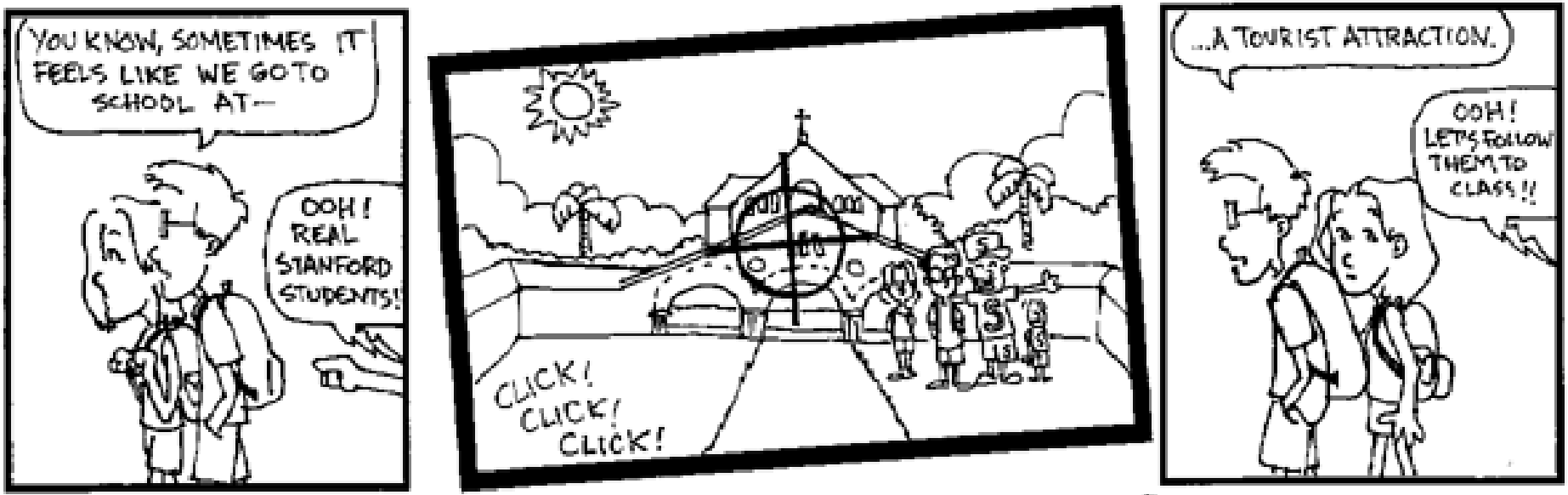
- What if you want to list the contents of a directory other than your current one? In that case you can just give that directory's path as an argument to *ls*. For example: *ls /home* will tell you what's in the directory */home*, no matter where you are.
- There are various flags (usually a - followed by a single character, or a – followed by a word) you can pass to *ls*. For example, try *ls -l* (more information every item on its own line), *ls -a* (lists hidden files as well), or *ls –group-directories-first* (self explanatory).

Arguments & flags

Most commands we will talk about can take many different kinds of arguments and flags to alter their function, don't forget to use *man* and *help* to find out more about them! Try it now with *man ls*.

Creating, moving, and destroying directories

- Maybe you want to create a new directory to house all your exciting new grad stuff, like the pictures that random tourists take of you. To do this, type *mkdir* (make directory) followed by the name for the directory, e.g. *mkdir exciting-grad-school-stuff*



Creating, moving, and destroying directories

- Maybe you want to create a new directory to house all your exciting new grad stuff, like the pictures that random tourists take of you. To do this, type *mkdir* (make directory) followed by the name for the directory, e.g. *mkdir exciting-grad-school-stuff*
- Maybe you changed your mind about what to call it though, so you want to change it to *boring-grad-school-stuff*, and also move it somewhere else. Not to worry! You can use *mv* (move) to rename the directory, or to move it to a new place. For example, you could use *mv exciting-grad-school-stuff* $\sim$*/boring-grad-school-stuff* to move it to your home directory and rename it. Give it a try!

Creating, moving, and destroying directories

- Maybe you want to create a new directory to house all your exciting new grad stuff, like the pictures that random tourists take of you. To do this, type *mkdir* (make directory) followed by the name for the directory, e.g. *mkdir exciting-grad-school-stuff*
- Maybe you changed your mind about what to call it though, so you want to change it to *boring-grad-school-stuff*, and also move it somewhere else. Not to worry! You can use *mv* (move) to rename the directory, or to move it to a new place. For example, you could use *mv exciting-grad-school-stuff* $\sim$*/boring-grad-school-stuff* to move it to your home directory and rename it. Give it a try!
- There is also a command called *cp* (copy) that works like *mv* except it copies the file.

Creating, moving, and destroying directories

- Maybe you want to create a new directory to house all your exciting new grad stuff, like the pictures that random tourists take of you. To do this, type *mkdir* (make directory) followed by the name for the directory, e.g. *mkdir exciting-grad-school-stuff*
- Maybe you changed your mind about what to call it though, so you want to change it to *boring-grad-school-stuff*, and also move it somewhere else. Not to worry! You can use *mv* (move) to rename the directory, or to move it to a new place. For example, you could use *mv exciting-grad-school-stuff ~/boring-grad-school-stuff* to move it to your home directory and rename it. Give it a try!
- There is also a command called *cp* (copy) that works like *mv* except it copies the file.
- Finally, maybe you think this is a silly directory and want to get rid of it. You can do that by using the command *rmdir* (remove directory), e.g. *rmdir ~/boring-grad-school-stuff*.

Special directories

- There are two special directories you'll see listed if you type *ls -a*, . and .., which are used to refer to the current directory and its parent, respectively.

Special directories

- There are two special directories you'll see listed if you type *ls -a*, *.* and *..*, which are used to refer to the current directory and its parent, respectively.
- For example, if you are in *~/Documents/grad/* and type *cd ..* your working directory will change to the parent of your current directory, i.e. *~/Documents*.

Table of Contents

What's all this stuff? (Files)

- Now let's look at the files within a directory. Many of the commands you've already learned still apply, *ls* will list them, and *mv* will move them.

What's all this stuff? (Files)

- Now let's look at the files within a directory. Many of the commands you've already learned still apply, *ls* will list them, and *mv* will move them.
- Let's create an empty file to play around with. You can do this by typing *touch emptyfile.txt* (in real life you'll usually be working with files you create in other programs, this way of creating them is just an example).

What's all this stuff? (Files)

- Now let's look at the files within a directory. Many of the commands you've already learned still apply, *ls* will list them, and *mv* will move them.
- Let's create an empty file to play around with. You can do this by typing *touch emptyfile.txt* (in real life you'll usually be working with files you create in other programs, this way of creating them is just an example).
- Now let's try to put some text in the file and save it, just for fun. You can do this from the command line, but how exactly you do it will depend on your OS.
  - **OS X:** try *open emptyfile.txt*.
  - **GNOME-based linux distros:** try *gedit emptyfile.txt*.
  - **Windows:** try *notepad emptyfile.txt*

Manipulating files, listing selectively

- Now that we put some text in the file, maybe we ought to rename it to *nonemptyfile.txt*. How do you think we do that?

Manipulating files, listing selectively

- Now that we put some text in the file, maybe we ought to rename it to *nonemptyfile.txt*. How do you think we do that?
- Yup, *mv emptyfile.txt nonemptyfile.txt*.

Manipulating files, listing selectively

- Now that we put some text in the file, maybe we ought to rename it to *nonemptyfile.txt*. How do you think we do that?
- Yup, *mv emptyfile.txt nonemptyfile.txt*.

Warning!

mv overwrites any files with the same name(s) in its destination, so be careful when using it! If there was another file in this directory called nonemptyfile.txt, we would have overwritten it.

Selectivity and globs

- Let's suppose you come back tomorrow and can't remember what you called this file. You can type *ls* to list everything in the directory and look through for it, but there might be a lot of other stuff. You can make commands be more selective by giving them some hints. For example, type *ls *.txt* to list all files in the current directory ending with a *.txt* extension. Type *ls *empty** to list files with *empty* in their name.

Selectivity and globs

- Let's suppose you come back tomorrow and can't remember what you called this file. You can type *ls* to list everything in the directory and look through for it, but there might be a lot of other stuff. You can make commands be more selective by giving them some hints. For example, type *ls *.txt* to list all files in the current directory ending with a *.txt* extension. Type *ls *empty** to list files with *empty* in their name.

Globs

The character * is called a glob, because it sticks together all the files that complete the rest of the pattern I guess, I don't know. Globs are often useful, e.g. you could move all the csv files in the current directory to new directory by typing *mv *.csv /Documents/my-new-data-directory*

Removing files

- Just like you can remove directories, you can remove files by using the command *rm* (remove). Try it now by typing *rm nonemptyfile.txt*.

Warning!

rm is VERY DANGEROUS, especially when used with globs and/or with certain flags (use *man rm* to find out more). It does not simply move a file to the trash, it deletes it completely. Double check what you type before you run it, and don't use *rm* if you're not sure what you're doing.

Table of Contents

There are computers other than mine? (Servers/SSH/Websites)

Yes, yes there are. Now we're going to show you how working with files on other computers from bash isn't too much harder than working with files on your own computer, using the tools *ssh* and *scp*. For this example, we'll use Stanford's FarmShare servers.

Basic SSH

- Let's try to connect to one of Stanford's **corn** servers, which are available for general use. Type *ssh your-sunet-id@corn.stanford.edu*
- When prompted for your password, type the password that corresponds to your sunet id. It won't show any characters being typed, just type the password and hit enter. Note: you will probably need to use two-factor authentication, and the timeout is relatively short, have your phone ready

Basic SSH

- Let's try to connect to one of Stanford's **corn** servers, which are available for general use. Type *ssh your-sunet-id@corn.stanford.edu*
- When prompted for your password, type the password that corresponds to your sunet id. It won't show any characters being typed, just type the password and hit enter. Note: you will probably need to use two-factor authentication, and the timeout is relatively short, have your phone ready
- You should see a welcome screen. If so, congrats, you're connected to a server!

Basic SSH

- Luckily these servers are running bash as well, so all the commands you've just learned should work here! Try *ls* and see what's around. This is your space on Stanford's servers, you can store documents here, and if you make a Stanford website it will be hosted in the WWW folder.
- Remember, you're no longer on your own computer, so the directory structure here will be different.
- When you're finished, you can type *exit* to close the connection to the server and return to your computer. (But don't do so yet, we're going to keep using it.)

A toy website

- Let's try creating a (very) simple website for you!

A toy website

- Let's try creating a (very) simple website for you!
- Create a text file called simplewebsite.txt somewhere on your computer, and put some text in it (like "Hello World!").

A toy website

- Let's try creating a (very) simple website for you!
- Create a text file called simplewebsite.txt somewhere on your computer, and put some text in it (like "Hello World!").
- Now, open a new terminal on your own computer (leave the other terminal with the ssh connection open, we'll go back to it in a bit). *cd* to the directory where you saved simplewebsite.txt.

A toy website (cont.)

- We need to move *simplewebsite.txt* to the Stanford servers. To do that, we'll use the command *scp* (secure copy), which allows you to copy files from/to your local computer to/from a remote one much like you would copy a file on your local computer using *scp*

A toy website (cont.)

- We need to move *simplewebsite.txt* to the Stanford servers. To do that, we'll use the command *scp* (secure copy), which allows you to copy files from/to your local computer to/from a remote one much like you would copy a file on your local computer using *scp*
- To copy *simplewebsite.txt* to the *~/WWW* folder on the server, try *scp simplewebsite.txt your-SUNetID@corn.stanford.edu:~/WWW/*

A toy website (cont.)

- We need to move *simplewebsite.txt* to the Stanford servers. To do that, we'll use the command *scp* (secure copy), which allows you to copy files from/to your local computer to/from a remote one much like you would copy a file on your local computer using *scp*
- To copy *simplewebsite.txt* to the *~/WWW* folder on the server, try *scp simplewebsite.txt your-SUNetID@corn.stanford.edu:~/WWW/*
- If all went well, you should see the name of the file followed by 100% (since the file is so small, the transfer will complete very rapidly).
- If so, try opening your web browser and going to *web.stanford.edu/~your-SUNetID/simplewebsite.txt*

A toy website (cont.)

- We need to move *simplewebsite.txt* to the Stanford servers. To do that, we'll use the command *scp* (secure copy), which allows you to copy files from/to your local computer to/from a remote one much like you would copy a file on your local computer using *scp*
- To copy *simplewebsite.txt* to the *~/WWW* folder on the server, try *scp simplewebsite.txt your-SUNetID@corn.stanford.edu:~/WWW/*
- If all went well, you should see the name of the file followed by 100% (since the file is so small, the transfer will complete very rapidly).
- If so, try opening your web browser and going to *web.stanford.edu/~your-SUNetID/simplewebsite.txt*
- Do you see your file? Congratulations! You've got a very basic website now. You can use the farmshare system to host experiments that you run online, to create a website for yourself so that people can look you up, etc. The process will be similar to this, except that you'll probably be creating html files instead of text files.

Cleaning up the website

- Why did we leave the ssh connection open in the other terminal? Because (hopefully) it allowed you to run *scp* without having to log in to the server again. Also, we might want to do something with the file on the server.

Cleaning up the website

- Why did we leave the ssh connection open in the other terminal? Because (hopefully) it allowed you to run *scp* without having to log in to the server again. Also, we might want to do something with the file on the server.
- Go ahead and run *ls ~/WWW* on the server so you can see *simplewebsite.txt* is there.

Cleaning up the website

- Why did we leave the ssh connection open in the other terminal? Because (hopefully) it allowed you to run *scp* without having to log in to the server again. Also, we might want to do something with the file on the server.
- Go ahead and run *ls ~/WWW* on the server so you can see *simplewebsite.txt* is there.
- You might not want the world to be able to see this file forever, so change to the *WWW* directory and remove the file.

Cleaning up the website

- Why did we leave the ssh connection open in the other terminal? Because (hopefully) it allowed you to run *scp* without having to log in to the server again. Also, we might want to do something with the file on the server.
- Go ahead and run *ls ~/WWW* on the server so you can see *simplewebsite.txt* is there.
- You might not want the world to be able to see this file forever, so change to the *WWW* directory and remove the file.
- Finally, close your connection to the server by typing *exit*.

Table of Contents

Wrapping up

Bash contains or allows access to many powerful tools, and has many arcane (but useful!) features. Here are a few examples of the things you can do with it that we won't have time to explain to you. If you're interested in more info, talk to us and we'll be happy to provide it!

Being lazy (scripts)

One of the main reasons bash is useful is that bash commands can be saved into scripts that can be reused. For example, here's a simple shell script I wrote to create anonymized data files by replacing participant identifiers in filenames with numbers starting from 0:

```
#!/bin/bash
i=0
for f in data_subject_*.json
do
  cp $f ../anonymized_data/data_subject_${i}.json
  i=$((i+1))
done
```

Programming

The terminal gives you access to programming languages like python, both by running programs directly, and by using **interpreters**, programs that run in the terminal and allow you to run commands interactively, much as you would in Matlab or R.

```
andrew@Galadriel:~/Documents/grad/teaching/bootcamp$ python
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>> x = numpy.ones((3,3))
>>> x
array([[ 1.,  1.,  1.],
       [ 1.,  1.,  1.],
       [ 1.,  1.,  1.]])
>>> numpy.dot(x,x)
array([[ 3.,  3.,  3.],
       [ 3.,  3.,  3.],
       [ 3.,  3.,  3.]])
>>>
```

Etc.

Also:

- Powerful commands for text manipulation (sed, awk, grep). Want to find all occurrences of a variable in many different code files and rename it? Want to extract lines from your datafiles that match a certain condition without having to read the files into R? Have a directory full of awful tab-separated datafiles and want to make them comma-separated? These tools can do it.
- Streams and piping: make commands work together, do file I/O, etc.
- Automation: more system specific, but cron, rc files, etc. allow for automation of things like backups, mounting filesystems when you start your computer, or sending reminders to subjects about a task on a schedule.
- Text-editors like *vim*.
- Git – You'll learn about this at the later tutorial!


"That's all Folks!"