1. Unix big picture

- Unix philosophy
- Filters + pipes
- Parts of an OS
- Shell:
  - `sh`, `csh`, `bash`, `zsh`, `ksh`, `tcsh`, ...
  - Interpreter
  - Programming language for shell scripts
- Unix allows multiple users to share a computer and use it simultaneously

Invest in Unix

Unix's *filters + pipes* paradigm provides a productive platform for doing data science:

- Each command does one thing (only) well
- Commands read from STD_IN and write to STD_OUT
- Can combine commands via I/O redirection with `|`, `<`, and `>`.
- Easy to write scripts to build bespoke commands from Unix's building blocks

References

Some references:

- Practical Guide to LINUX Commands, Editors, & Shell Programming
- Design and Implementation of the FreeBSD Operating System

Processes

Unix runs commands in a process:

- Contains trampoline code (crt0) to start & terminate a thread of execution
- Memory for stack & heap
- File descriptor table
- Environment variables which contain state information
- Error status (`errno`)
- Signal handlers to respond to OS events, like SIGFPE
- Unix creates a new process by calling `fork` and then `exec`s your command

Privilege

Unix processes run at different privilege levels:

- User vs. superuser
- Try to run processes at the lowest possible level to increase security
- For admin tasks, may need to run at superuser level using `sudo`
- In the old days, there was a special superuser account named `root`

The Shell

You interact with the shell:

- Many shells exist: `sh`, `csh`, `bash`, `zsh`, `ksh`, `tcsh`, ...
- `bash` is most popular today
- Runs whatever commands you type in
- Has control flow and can also run scripts just like a programming language

What is going on with my *nix?

Some useful commands to check the state of your system:

- `uname`
- `whoami`
- `who`, `users`
- `ps`, `top`, `vmstat`

Unix commands

Unix command philosophy:

- Most commands do one thing and do it well
- May have multiple options to configure performance
- By default, commands read from STD_IN and write to STD_OUT

2. Help

There are many help options:

- References provided above
- `man`
- `info`
- run command with `-help` or `--help`
- Google
- StackOverflow

3. Navigating in Unix

The key objects you will need to manipulate:

- Files: contain data and can be text or binary
- Directories: contain files, directories, links, and special files
- Links: a short-cut which points to another file or directory (hard, soft)

If you live in the GUI world and not the CLI (which is sacred), you will call 'directories' 'folders'

Navigation essentials (1/2):

Essentials to navigating:

- You have a home directory:
  - contains your personal directories & files
  - Stored in `$HOME` environment variable
- Directory shortcuts
  - `~` refers to home directory
  - `.` refers to current directory
  - `..` refers to parent directory
- Use `/` to separate directory and/or file names
- `pwd` displays current directory
- `ls` displays files in current directory

Navigation essentials (2/2)

Essentials to navigating:

- Use `cd` to change directories:
  - `cd from_dir to_dir`
  - `cd` with no arguments takes you to `~`
- Create/delete directories with `mkdir` and `rmdir`
- Rename files or directories with `mv old_name new_name`
- Use `file some_file_or_dir` to determine type of file
- `rm` will delete files
- `rm -rf dir` will recursively delete `dir` and everything it contains

Permissions

Unix permissions are set at three levels of access:

- Grouped by *user*, *group*, and *world*
- Permission for each group can be read and/or write and/or execute
- Set via `chmod`

Example: permissions

```
$ ls -l ~/.ssh
total 64
-r--------@ 1 bss  staff  1696 Aug 25  2015 bss-aws-master.pem
-rw-r--r--  1 bss  staff   381 Aug 25  2015 bss-aws-master.pub
-rw-r--r--@ 1 bss  staff   295 Aug 25  2015 config
-rw-------  1 bss  staff  3247 Jun  3  2015 git-hub-id_rsa
-rw-r--r--  1 bss  staff   748 Jun  3  2015 git-hub-id_rsa.pub
-rw-------  1 bss  staff  3247 Jun  3  2015 id_rsa
-rw-r--r--  1 bss  staff   748 Jun  3  2015 id_rsa.pub
-rw-r--r--  1 bss  staff  4018 May  3 09:02 known_hosts
$ chmod 600 id_rsa
```

Examining files

There are many options to examine the contents of a file:

- `cat`
- `less`
- `head` & `tail`

Manipulating files

A couple useful tools to manipulate files:

- `wc`
- `sort`, `uniq`, `cut`, `paste`
- `touch`

Tar files

The old school way to create archives is with `tar`:

- Stands for "Tape ARchive"
- `tar cvf mytarfile.tar *` to create a tar file
- `tar xvf mytarfile.tar` to extract a tar file
- `tar tvf mytarfile.tar` to list the contents of a tar file
- Add option `z` for compression
- Can also use `zip`, `bzip2`, etc.

4. Environment

Environment variables store information about the state of the system:

- Create with `export MY_VAR=my_value`
- Usually stored in `~/.profile` or `~/.bashrc`
- Must set `PATH` and `LD_LIBRARY_PATH`
- Will also want to set `PROMPT` or `PS1`
- May need to set other configuration information like `MANPATH` `PYTHONPATH`, `AWS_ACCESS_KEY_ID`, and `AWS_SECRET_ACCESS_KEY`

Using environment variables

To examine your environment:

- `env`
- `echo $PATH`

To access an environment variable:

- Use `$MY_VAR` or `${MY_VAR}`
- Will expand like a macro
- More complex manipulations are possible – RTFM

5. Configuration

To configure your system:

- Must setup dotfiles to explain where your resources are located
- Setup aliases for convenience and productivity

Which dotfile should I use to configure `bash`?

Configure `bash` using special dotfiles:

- `~/.bash_profile` should just load `~/.profile` and then `~/.bashrc`
- `~/.profile`:
  - Anything which is not bash-specific
  - Any setup needed by login shells
  - E.g., setting PATH and other environment variables
- `~/.bashrc`:
  - Anything for interactive bash session
  - E.g., prompt, EDITOR, aliases, etc.
  - Do not write to STD_OUT

From http://superuser.com/questions/789448/choosing-between-bashrc-profile-bash-profile-etc

Example: `~/.bash_profile`

```
# Setup basic environment
if [-f ~/.profile ]; then
    . ~/.profile
fi

# Setup interactive shell
if [-f ~/.bashrc ]; then
    . ~/.bashrc
fi
```

Checking your environment

Several tools to check your setup:

- `env`
- `echo $PATH`
- `which *cmd*`

6. Finding stuff

There are many powerful tools to find files:

- `grep`, `egrep`, and `fgrep`
- `find`
- `locate`
- `ack`
- Most use regular expressions to describe search pattern

Regular Expressions (RE)

Many Unix tools support RE:

- General language to describe complex patterns
- Enables sophisticated searching, find/replace, and editing
- Used by `sed`, `awk`, `grep`, `vi`, ...
- See reading

7. Other topics

Standard in, out, and error

Unix refers to the default devices for I/O as:

- Standard input (0)
- Standard output (1)
- Standard error (2)

Often referred to via file descriptors 0, 1, and 2 respectively

I/O redirection

You can combine commands via I/O redirection

- `|`
- `>`
- `<`
- `<< EOF`

Stream editors

Unix pioneered several 'programmable' editors which can help you manipulate files & data:

- `sed` is a stream editor which performs editing operations on a filestream
- `awk` is a programmable C or bash-like language which performs operations whenever a pattern matches the filestream
- `perl` and `python` evolved from these languages

Job control

You can manage jobs & processes:

- `ps`
- `kill`
- `&`, `CTRL-Z`, `bg`, `fg`, `jobs`
- Parent & child processes; reaping

Remote access

Unix is fabulous for accessing remote machines:

- `ssh`

- `sftp`
- `scp`
- Do not use `telnet` or `rlogin`

Installing packages

- `sudo apt-get install ....`
- `sudo yum ...`
- `sudo rpm ...`

Shell scripts

The shell is also a programming language:

- Can quickly write scripts to automate tasks
- Supports control flow and functions
- First line of script should be `#!/bin/bash`
- Set script's mode to executable with `chmod`