Lecture 05 - Unix shell scripts

why write a shell script?

why write a shell script?

- repeat tasks easily
- reproducible and documented (comments!!!!)
- modularize your analysis processes

arguments to a shell script

- `$1`, `$2`, `$3`, etc.
- `$@` - useful with loops in a shell script!

shell scripts in an ‘orthoganal’ manner

- remember the power of Unix is it’s modularity
- BUILD SCRIPTS TO READ FROM `stdin` AND WRITE TO `stdout`

convert a SWC shell script to use `stdin` and `stdout`

```
# Select lines from the middle of a file.
# Usage: bash middle.sh filename end_line num_lines
head -n "$2" "$1" | tail -n "$3"
```

```
bash middle.sh pentane.pdb 15 5
```

convert a SWC shell script to use `stdin` and `stdout`

```
# Select lines from the middle of a file.
# Usage: middle2.sh end_line num_lines
head -n "$1" | tail -n "$2"
```

```
cat pentane.pdb | bash middle2.sh 15 5
```

making scripts behave even more like a function

To behave like other basic functions a shell script needs:

- a 'shebang' as the first line
  - `#!/path/to/interpretor`
  - this tells the shell what to use to run the text in the script

- the script file must be executable by the user

file permissions

if we use `ls -l *.sh` to look at our shell scripts, we can see read, write, and execute permissions

- usually the first character indicates files vs. directories

- the next nine characters indicate the presence or absence of read (r), write (w), and execute (x) permissions for the "owner", "group", and "other" users

changing file permissions

- the function `chmod` can be used to change permissions

- to add executable permission for the owner try: `chmod u+x filename`
  - *who* comes first: user (u), group (g), other (o), or all (a)
  - "op" comes next: this indicates if we are adding (+) or removing (-) permissions
  - "permission" is last: read (r), write (w), and execute (x) are the most common

convert a SWC shell script to use `stdin` and `stdout`

```
#!/usr/bin/env bash

# Select lines from the middle of a file.
# Usage: middle3.sh end_line num_lines
head -n "$1" | tail -n "$2"
```

```
chmod u+x middle3.sh
```

```
cat pentane.pdb | ./middle3.sh 15 5
```

convert a SWC shell script to use `stdin` and `stdout`

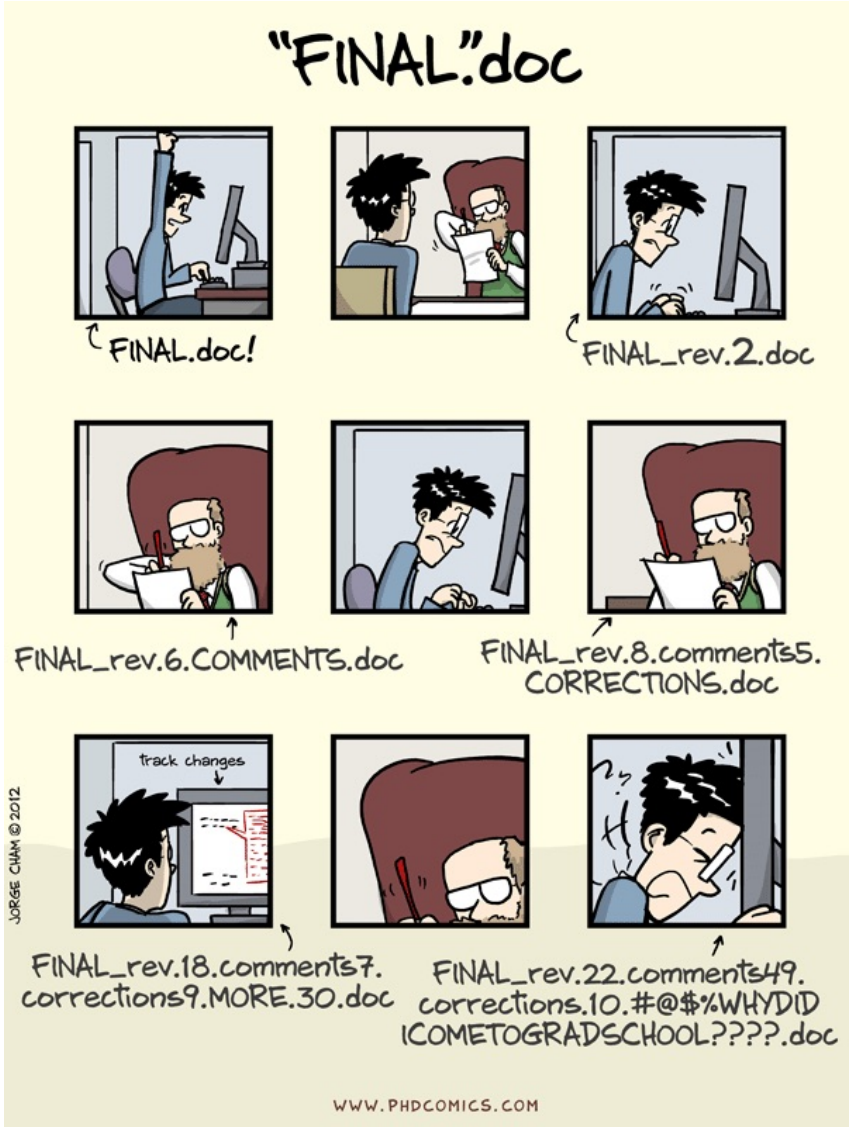
```
bash middle.sh pentane.pdb 15 5
```

```
cat pentane.pdb | bash middle2.sh 15 5
```

```
cat pentane.pdb | ./middle3.sh 15 5
```

version control and Git

A computational best practice that solves this problem, among others. . .



version control and Git

- A set of functions that allow tracking of progress in a project and facilitate collaboration.
- Becoming the gold standard in biocomputation-focused research.
- Sets up a hidden directory that stores information about changes in a working directory.

```
git init

git add file_names
git commit -m "custom commit message"
```

- Consult documentation ('Git_Setup.pdf') available on Sakai for installation instructions.

function review

```
chmod
```

```
#!/path/to/interpretor
```