%title: OLUG.org August 2nd, 2022 %author: Aaron Grothe & Matt Payne %date: 2022-08-02

-> CLI is Hard <-

In the terminal, the command line interface, things are hard.

1. Don't know your location (`pwd` == print working directory)
2. `cd new_location` == goes to new_location
    1. Absolute paths start with `/`
    2. Relative paths do not start with `/`
    3. `.` is the current directory and `..` is the parent directory
3. `cd -` == goes to the previous location

Things are hard, until you are used to it.

1. you can not use your mouse
2. Arrow keys help
3. CTRL-R reverse searches the history. Also the `history` command.
4. Result code: `echo $?` (0 is good, not zero is bad)

-> CLI power tools <-

1. redirection
2. pipes
3. find
4. grep, egrep, git grep
5. xargs
6. for loops
7. job controls
8. environment variables
    1. source script.sh # notes about fork & exec
9. sed & awk

-> CLI power tools: Redirection <-

redirection - combine commands and files

1. `>` Send standard out (stdout) to a file
    1. `w > w.ouput`
2. `>>` Append standard out (stdout) to a file
    1. `uptime >> uptime.ouput`
3. `<` Take standard in (stdin) to a file
    1. `tidy -i -xml < some-badly-formatted.xml`

-> CLI power tools: Pipes <-

pipes - combine commands 1. `cat filename | nl` # Number the lines of a file 2. `cat filename | grep -i aaron | nl` # Number lines that contain Aaron (`-i` is case insensitive)

-> CLI power tools: find <-

find - Walk directory trees and output files & directories 1. `find .` # print 'em all! 2. `find . -type f` # Just print regular files (no directories)

-> CLI power tools: environment grep etc <-

egrep, git grep 1. extended grep handles regular expressions 1. `find . -type f | egrep -i 'java|kt|ts' | nl` # print files with java or kt or ts in their names 2. git grep - like grep but only search files being tracked by git 1. `git grep -i repository \*.java` # search java files for repository

---

-> CLI power tools: xargs <-

xargs - Flip stdin to parameters

Also, sometimes, useful when you hit the limit of the # of command line parameters...

`ps -ef | grep runaway | awk ' { print $2 }' | xargs kill` # kill runaway process

-> CLI poser tools: for loop <-

When you want to build up what you're doing 1. Confirm what you're getting 2. Then put the action into the loop

-> # The most BASH Matt ever writes <-

```
for f in `find . -type f | egrep -i 'something|otherthing'` do echo $f cp $f ${HOME}/some/place/else done
```

-> CLI power tools: job control <-

job control do things in the background and then bring it into the foreground

1. CTRL-Z # put the current jobs into the background. Suspends it too
2. `jobs` # List the jobs
3. `bg %2` # Run job two in the background
4. `fg` *OR* `fg %1` # Bring job one into the foreground

-> CLI power tools: environment variables <-

environment variables - Global variables that can't be changed by a child process

---

-> CLI power tools: environment sed & awk <-

1. sed - Stream editor, good for editting very large files. And quick CLI jobs.
    1. `cat bigfile | sed -e 's/Matt/Matt Payne/g'` # Note the vi style substitution
    2. Really, vi uses ed style substitution
2. awk - VERY handy and quick to learn. Father of PERL etc.

AWK command I used at work the other day

```
grep '<testsuite' * | awk ' { printf("%s %s %s %s\n", $(NF-3), $(NF-2), $(NF-1), $NF); }' tests="2" errors="0" skipped="0"
failures="0"> tests="3" errors="0" skipped="0" failures="0"> tests="1" errors="0" skipped="0" failures="0"> tests="4"
errors="0" skipped="0" failures="0"> tests="6" errors="1" skipped="0" failures="0"> tests="9" errors="0" skipped="0"
failures="0">
```

-> Classic AWK hello world -- word frequencies

https://jeffsum.com/ to get some random chatter - copy & paste with `cat > jeffsum.txt`

```
cat jeffsum.txt | awk ' { for (i=1; i < NF; i++) { w[$i]++; } } END { for (i in w) { printf("%s happens %d times\n", i,
w[i]); } }'
```

Not the parts that 1. Only run at the BEGIN of the input (we don't have one) 2. Run for each line of the input 1. Notice that $1 is the first field (word - splits on blanks) 3. Only run at the END of the input (we use it to dump the associative array)

-> Slide: References <-

https://swcarpentry.github.io/shell-novice/

-> CLI misc tricks <-

1. export A=`pwd` # then use $A as part of a destination
2. export A=$(pwd) # then use $A as part of a destination, but passes bash lint.
3. mkdir -p some/big/deep/{part1,part2}/paths/you/{must,want}/to/make
4. cd - # Like the previous channel on a remote control
5. the deal with source script vs. ./script
6. tricks with redirecting and pipes
7. environment variables -- what's to know other than PATH?