

# CSE 391

## Shell commands More Redirection

[pollev.com/cse391](https://pollev.com/cse391)

- How was the last homework assignment?

# AGENDA

- Logistics, Roadmap
- Combining Commands
- More input/output redirection
- cut, reading log files

# ROADMAP

- Introduction to the command line
- Input/output redirection, pipes
- More input/output redirection, tee, xargs
- Git: Fundamentals
- Git: Branches and rebasing
- Regular expressions
- More regular expressions, sed
- Users and permissions
- Bash scripting
- Industry applications

# PIPES

`command1 | command2`

- Execute **command1** and send its standard output as standard input to **command2**.
- This is essentially shorthand for the following sequence of commands:

```
command1 > filename
command2 < filename
rm filename
```
- This is one of the most powerful aspects of unix - being able to chain together simple commands to achieve complex behavior!

# COMBINING COMMANDS

**command1 ; command2**

- Execute **command1**, then execute **command2**.

**command1 && command2**

- Execute **command1**, and if it succeeds, then execute **command2**.

**command1 || command2**

- Execute **command1**, and if it fails, then execute **command2**.

**THINK:** [pollev.com/cse391](http://pollev.com/cse391)

- What would happen after running the following command: `ls *.java | javac`



# PAIR: [pollev.com/cse391](https://pollev.com/cse391)

- What would happen after running the following command: `ls *.java | javac`



**2:00**

# XARGS

- **xargs** is a program that converts standard input to command line arguments (i.e. parameters).
- For example, to compile all java files in the current directory we could use the following:
  - `$ ls *.java | xargs javac`

# FIND

- **find** is a program for searching your filesystem for certain files.
- For example, to list all java files in the current directory and all subdirectories, recursively, we would run the following
  - `$ find -name "*.java"`
- This is commonly used with **xargs**. For instance, to compile all Java files in the current directory and all subdirectories recursively
  - `$ find -name "*.java" | xargs javac`
- Note that find has a plethora of options and flags, but we will most commonly use find with the `-name` and `-type` flags

# COMMAND SUBSTITUTION

## `$(command)`

- Another powerful tool is command substitution. It executes the given command and places that string literally into the given context.
- For example, to compile all Java files in the current directory and subdirectories recursively, we can run the following
  - `$ javac $(find -name "*.java")`

**THINK: [pollev.com/cse391](http://pollev.com/cse391)**

- What is the command to remove all of the files listed inside of the file `toRemove.txt`?



# PAIR: [pollev.com/cse391](http://pollev.com/cse391)

- What is the command to remove all of the files listed inside of the file `toRemove.txt`?



**2:00**

# STDERR REDIRECTION

- We've learned that we can redirect standard error using the `2>` operator.
- Sometimes, however, we want standard error and standard out to go to the same location. We can do that with the following syntax:
  - `$ command > out.txt 2>&1`
- To understand this command, this reads as “redirect standard out to `out.txt`, redirect standard error to the same place as standard out”

# TEE

- Sometimes, we want to redirect the output of a command to both a file and to the console. To do this, we can pipe the output of a command to tee
  - `$ command | tee file.txt`
- To redirect both standard output and standard error to a file, and to the console, we use the following
  - `$ command 2>&1 | tee file.txt`

**THINK:** [pollev.com/cse391](http://pollev.com/cse391)

- Suppose we want to run the Java program `Mystery`. What would be the command to output both standard error and standard output to `mystery_out.txt` *and* print both to the console?



# PAIR: [pollev.com/cse391](https://pollev.com/cse391)

- Suppose we want to run the Java program `Mystery`. What would be the command to output both standard error and standard output to `mystery_out.txt` *and* print both to the console?



# CUT

**cut -d<DELIMITER> -f<FIELD>**

- cut is a simple program to split lines based on a given delimiter.
- For example, to split the string “a,b,c,d,e” on commas and get the second entry, we would use the following:
  - `$ echo “a,b,c,d,e” | cut -d, -f2`
  - Note: the echo program simply prints the given string to standard out

# LOGS

- A common exercise in daily software development and operations is looking at log files - basically a status report of what is going on inside the program.
- We can look at the logs for all the CSE course websites by reading the file:  
`/cse/web/courses/logs/common_log`
- For example, to actively watch the log file and only look for access to our own course website, we could use the following

```
$ tail -f /cse/web/courses/logs/common_log | grep "391"
```