



BASH: STANDARD STREAMS + ONELINERS

Shruti Chidambaram

Midterm next week!

Extration this week:

Midterm Review Session!

Sat & Sun 1-2 PM

Gates 5222

Review

- sed
- grep
- echo

Standard Streams: **stdin**, **stdout**, **stderr**

Communication channels for input and output between programs

stdin	stdout	stderr
"standard input"	"standard output"	"standard error"
Listen for text input By default: keyboard	Output "normal" text By default: terminal	Output "error" text By default: terminal

Redirection

- Usually, `stdin` is keyboard and `stdout` & `stderr` are the terminal
- Doesn't have to be!
- Can specify where input comes from or where output goes

Redirection

Syntax	Meaning
<code>[command] < file.txt</code>	stdin from file.txt
<code>[command] > file.txt</code>	stdout to file.txt (overwrite)
<code>[command] >> file.txt</code>	stdout to file.txt (append)
<code>[command] 2> file.txt</code>	stderr to file.txt (overwrite)
<code>[command] 2>> file.txt</code>	stderr to file.txt (append)

Redirection: Examples

```
echo "Hello" > hello.txt
```

```
$ cat hello.txt
```

```
# Overwrite 'hello.txt'
```

```
$ echo "Goodbye" > hello.txt
```

```
$ cat hello.txt
```

```
# Append output
```

```
$ echo "Hello again" >> hello.txt
```

```
$ cat hello.txt
```


Redirection: Examples with stderr

Redirecting stderr is useful for logging output for later reference

```
$ cat yikes
```

```
cat: yikes: No such file or directory
```

```
$ cat yikes 2> errors.log
```

```
$ cat errors.log
```

```
cat: yikes: No such file or directory
```

stdin vs. arguments

- Command line arguments are passed to a program, while `stdin` is an input stream that a program/command gets data from
- `echo` does not read from `stdin`
- `grep`, `cat` take command line arguments or can read from `stdin`

grep with and without redirection

```
grep findme < file.txt
```

```
grep findme file.txt
```



Typing out a 20
character terminal
command

@NPCCompleteleens



Pressing up-arrow
348392345 times
to the place where
you last used said
command

oneliners

|Pipes|

Pipes send `stdout` of one command to `stdin` of another

Pipe character: Shift + Backslash (|)

a fun example: `fortune | cowsay`

Oneliners are chains of pipes

Start with some sort of data, then filter it down

Useful Commands

From last week: sed, grep

New:

find

-name find <directory> -name "<pattern>"

-regex find <directory> -regex "<regex>"

curl curl <url> <options>

xargs xargs <command>

Example

```
find . -name "*pdf" | grep -v "written.pdf" | xargs open
```

“why would I ever use this?”

- A long chain of commands can easily be executed with a oneliner
- Avoid the manual redirection into another file at each step of the way
- True power of bash

Tips for Writing Oneliners

- Construct oneliners iteratively!
 - Try the first command, see what it outputs
 - Try the first two commands, see what they output
 - and so on ...
- Multiple ways/tools do the same thing
 - Choose what you're familiar with
- “Google is your friend! Your friends are your friends!”

Lab Pro Tips

Helpful commands for pipelab:

- Curl - pulls content from an url
- Sed - Edits text (stream editing) (input can be supplied through stdin)
- Xargs <command> - Transformed newline separated text in stdin to arguments for the given command
- Test locally first! Construct iteratively!
- Small secret:
 - ./driver/driver is a bash script
 - Wow! (you can hack it if you want
 - But its probably easier to do the lab...)

