

CS 241: Systems Programming

Lecture 2. Introduction to Unix and the Shell

Fall 2019

Prof. Stephen Checkoway

What is the shell?

Text-based interface to the operating system and to the file system

User enters commands

The shell runs the commands

Output appears on a terminal (terminal emulator)

Commands can change files/directories on the file system

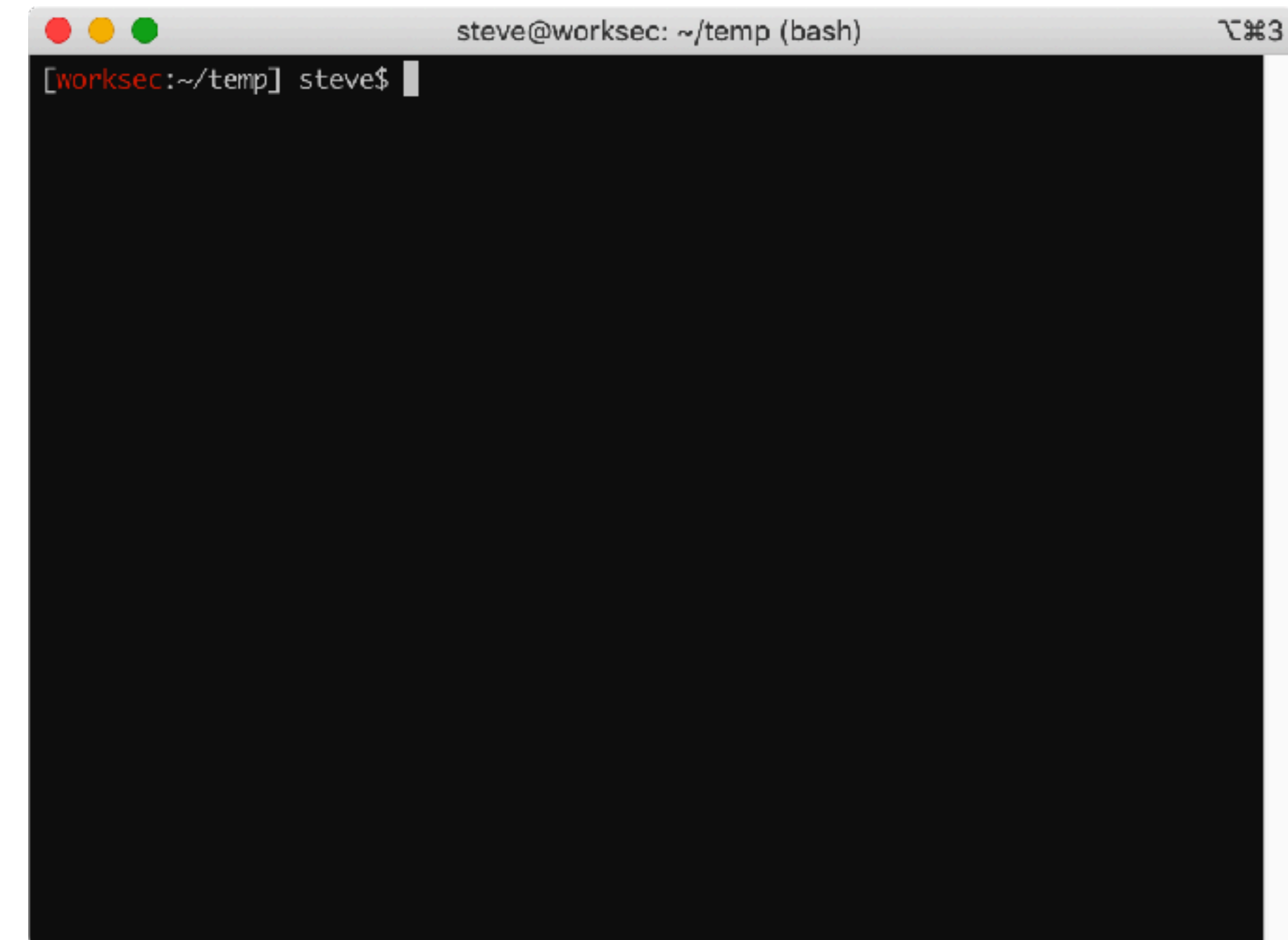
Terminals/terminal emulators

DEC VT100 terminal



<https://upload.wikimedia.org/wikipedia/commons/6/6f/Terminal-dec-vt100.jpg>

iTerm2 terminal emulator



There are many shells

sh Bourne shell

bash Bourne again shell (the one we'll be using)

dash Light-weight Bourne shell (often named sh on Linux)

csh C shell

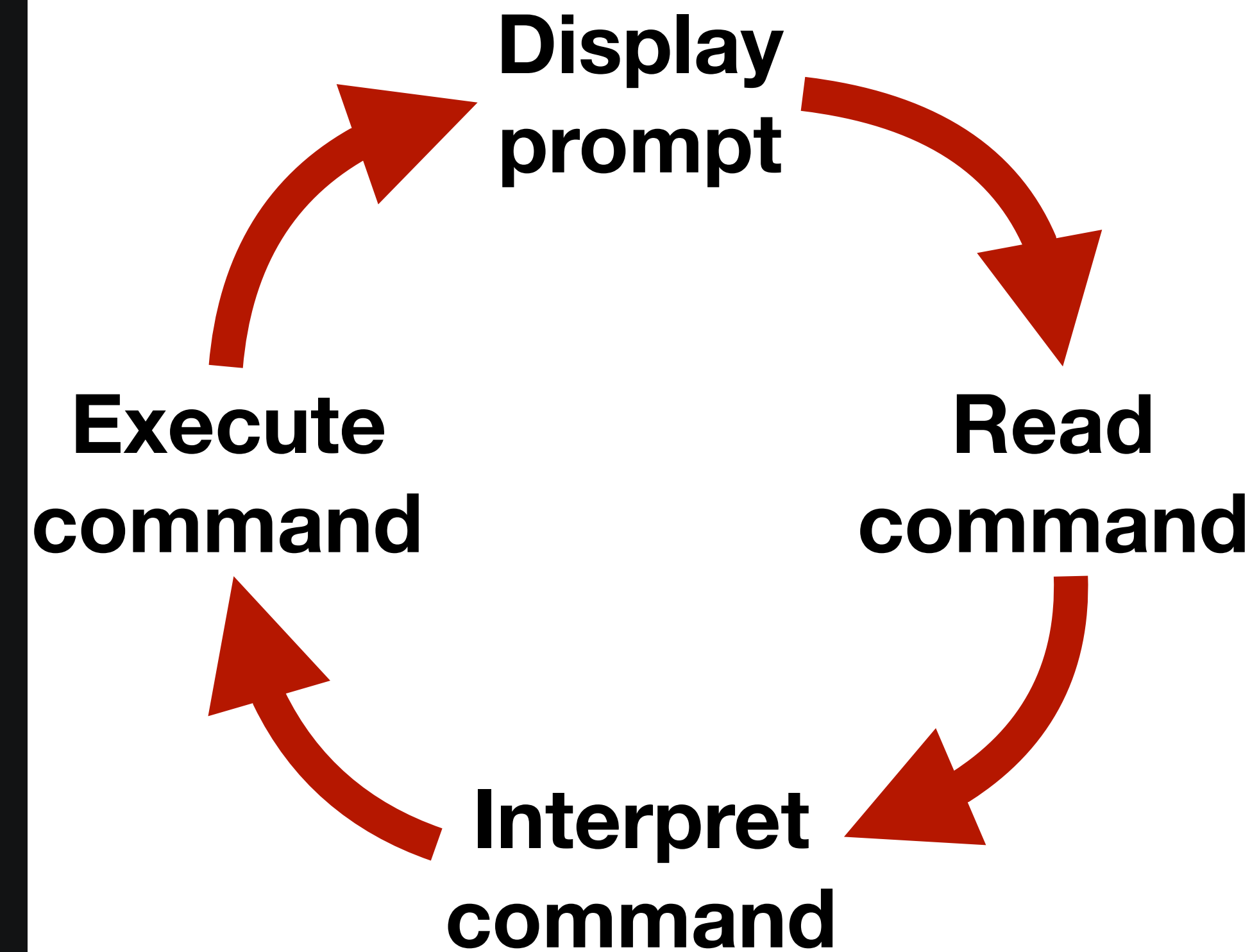
tcsh An improved csh

ksh Korn shell (sh-compatible, some csh features)

zsh Z shell (incorporates aspects of tcsh, ksh, and bash)

Interpreter loop

```
[worksec:~/temp] steve$ █
```



The file system

Structured as a single tree with **root** node: /

Directories hold files and directories

We name files (or directories) by giving a path through the tree

- ▶ **Absolute path**: /usr/bin/ssh
- ▶ **Relative path** (we'll come back to this)



Some important directories

<code>/</code>	The root directory
<code>/bin</code>	Holds programs used for essential tasks (e.g., <code>cp</code> , <code>mv</code> , <code>ls</code>)
<code>/sbin</code>	Superuser (administrator) binaries
<code>/etc</code>	System-wide configuration files
<code>/usr</code>	Holds programs and support files for user programs
<code>/usr/bin</code>	User binaries
<code>/home</code>	Holds users' home directories (this is configurable)

The current working directory

Every program on the system has its own **current working directory**

Not related to where the program lives in the file system

Programs can change their current working directory

The initial working directory of a running program is the current working directory of the parent

Bash's current working directory

The shell has a current directory (like every running program)

`cd` changes the current working directory

`pwd` prints the current working directory

Recall that we can name files using an absolute path or a relative path

- Absolute (starts with a `/`): `/usr/bin/ssh`
- Relative to the current working directory (doesn't start with a `/`)

Programs run by `bash` start with their initial working directory set to `bash`'s current working directory

Example of a relative path

```
steve@clyde:~$ █
```

If we have three (poorly named) files with paths

`/dir/file`

`/dir/dir/file`

`/dir/dir/dir/file`

and we run the two commands

```
$ cd /dir
```

```
$ rm dir/file
```

which file is deleted?

A. `/dir/file`

B. `/dir/dir/file`

C. `/dir/dir/dir/file`

D. All three files

E. None of them (e.g., because it's an error)



Two special directory entries

Each directory contains two special entries

- ▶ `.` the directory itself (pronounced "dot")
- ▶ `..` the directory's parent (pronounced "dot dot")

We can use these in paths

- ▶ These all refer to the same directory
 - `/usr/bin`
 - `/usr/./bin/.`
 - `/etc/../usr/bin`
- ▶ `.` is usually only used at the start of a relative path as `./`
 - `./foo`
- ▶ `cd ..` takes us to the parent directory of the current directory
- ▶ `cd ../..` takes us to the current directory's parent's parent

Which directory is listed if we run the following two commands in the shell?

```
$ cd /usr
```

```
$ ls bin/../../bin
```

A. /

B. /bin

C. /usr/bin

D. /usr/bin/bin

E. Some other directory

Anatomy of a single command

⟨command⟩ ⟨options⟩ ⟨arguments⟩

- ▶ ⟨command⟩ is the name of a command or a path to a program
- ▶ ⟨options⟩ are directives to the command to control its behavior
 - Short options are a hyphen and a letter: `-h`
 - Long options are (usually) two hyphens and multiple letters: `--help`
 - Multiple short options can be combined `-a -b -c` is the same as `-abc`
 - Options can take arguments: `-o file.txt` or `--output=file.txt`
- ▶ ⟨arguments⟩ are the things the command acts on
 - Often file paths or server names or URLs
 - When no arguments are given (or a single `-`), many commands read stdin

Example: `tar -zcf archive.tar.gz --verbose dir/file1 file2`

Example meaning

▼ `tar(1)` `-zcf` `archive.tar.gz` `--verbose` `dir/file1` `file2`

• The GNU version of the tar archiving utility

• `-z, --gzip, --gunzip --ungzip`

• `-c, --create`
create a new archive

• `-f, --file ARCHIVE`
use archive file or device ARCHIVE

`-v, --verbose`
verbosely list files processed

`tar [-] A --catenate --concatenate | c --create | d --diff --compare | --delete | r --append | t --list |`
`--test-label | u --update | x --extract --get [options] [pathname ...]`

[Click to go to explainshell.com](https://explainshell.com)

Shell commands

Shell builtins

- Functionality built into bash (all listed in the manual)
- E.g., `cd`, `alias`, `echo`, `pwd`

Shell functions

- User-defined functions (we'll get to these later)

Aliases

- E.g., `alias ls='ls --color=auto'`

Programs stored on the file system

- `/bin`, `/usr/bin`, `/usr/local/bin`, `/sbin`, `/usr/sbin`
- E.g., `ssh`, `cat`, `ls`, `rm`

Useful commands

- ▶ `ls` – list files
- ▶ `cd` – change directory
- ▶ `pwd` – print the working directory
- ▶ `pushd`, `popd`, `dirs` – use a stack to change directories
- ▶ `cp` – copy a file
- ▶ `man` – show the manual page
- ▶ `mv` – rename (move) a file
- ▶ `mkdir`, `rmdir` – make or delete a directory
- ▶ `rm` – delete a file
- ▶ `chmod` – change file permissions
- ▶ `cat` – concatenate files
- ▶ `more`, `less` – pagers
- ▶ `head`, `tail` – show first/last lines
- ▶ `grep` – match lines
- ▶ `wc` – count words
- ▶ `tr` – transform characters
- ▶ `split`, `join`, `cut`, `paste`
- ▶ `sort`, `uniq`

Manual (man) pages

`man` is the system manual pager

- Use this to find out more about Unix programs
- `$ man cp`

`whatis` show just single line information

- also via `$ man -f cp`

`apropos` search for keyword, return single lines

- also via `$ man -k cp`

`whereis` locate binary, source, man page

- `$ whereis cp`
`cp: /bin/cp /usr/share/man/man1/cp.1.gz`

Sections of the manual

Divided into sections

1. user commands (e.g., `cp(1)`, `ls(1)`, `cat(1)`, `printf(1)`)
2. system calls (e.g., `open(2)`, `close(2)`, `rename(2)`)
3. library functions (e.g., `printf(3)`, `fopen(3)`, `strcpy(3)`)
4. special files
5. file formats (e.g., `ssh_config(5)`)
6. games
7. overview, conventions, and miscellany section
8. administration and privileged commands (e.g., `reboot(8)`)

Use `man 3 printf` to get info from section 3

- You can use `man -a printf` to get all sections

Pathname expansion/globbing

Bash performs pathname expansion via **pattern matching** (a.k.a. **globbing**) on each unquoted word containing a wild card

Wild cards: *****, **?**, **[**

- ▶ ***** matches zero or more characters
- ▶ **?** matches any one character
- ▶ **[...]** matches any single character between the brackets, e.g., **[abc]**
- ▶ **[!...]** or **[^...]** matches any character not between the brackets
- ▶ **[x-y]** matches any character in the range, e.g., **[a-f]**

Example

```
$ ls ex/*.txt
```

```
ex/a-1.txt  ex/a-2.txt  ex/a-3.txt  ex/b-1.txt  
ex/b-2.txt  ex/b-3.txt
```

```
$ ls ex/?-3.*
```

```
ex/a-3.bin  ex/a-3.txt  ex/b-3.bin  ex/b-3.txt
```

```
$ ls ex/[^acd]-[0-9].b*in
```

```
ex/b-1.bin  ex/b-2.bin  ex/b-3.bin
```

```
$ ls "ex/*"
```

```
ls: cannot access 'ex/*': No such file or  
directory
```

```
ex  
├── a-1.bin  
├── a-1.txt  
├── a-2.bin  
├── a-2.txt  
├── a-3.bin  
├── a-3.txt  
├── b-1.bin  
├── b-1.txt  
├── b-2.bin  
├── b-2.txt  
├── b-3.bin  
├── b-3.txt  
└── README
```

```
CP(1)                                User Commands                                CP(1)
NAME
  cp - copy files and directories

SYNOPSIS
  cp [OPTION]... [-T] SOURCE DEST
  cp [OPTION]... SOURCE... DIRECTORY
  cp [OPTION]... -t DIRECTORY SOURCE...

DESCRIPTION
  Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.
```

Which command copies all Java source files (those whose names end in `.java`) from the directory `a/b` to the directory `/tmp`?

- A. `$ cp a/b/[a-z].java /tmp`
- B. `$ cp a/*/*.java /tmp`
- C. `$ cp a/b/*.java /tmp`
- D. `$ cp a/b/?*.java /tmp`
- E. `$ cp a/b /tmp *.java`

In-class exercise

<https://checkoway.net/teaching/cs241/2019-fall/exercises/Lecture-02.html>

Grab a laptop and a partner and try to get as much of that done as you can!