

CS 241: Systems Programming

Lecture 2. Introduction to Unix and the Shell

Spring 2020

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

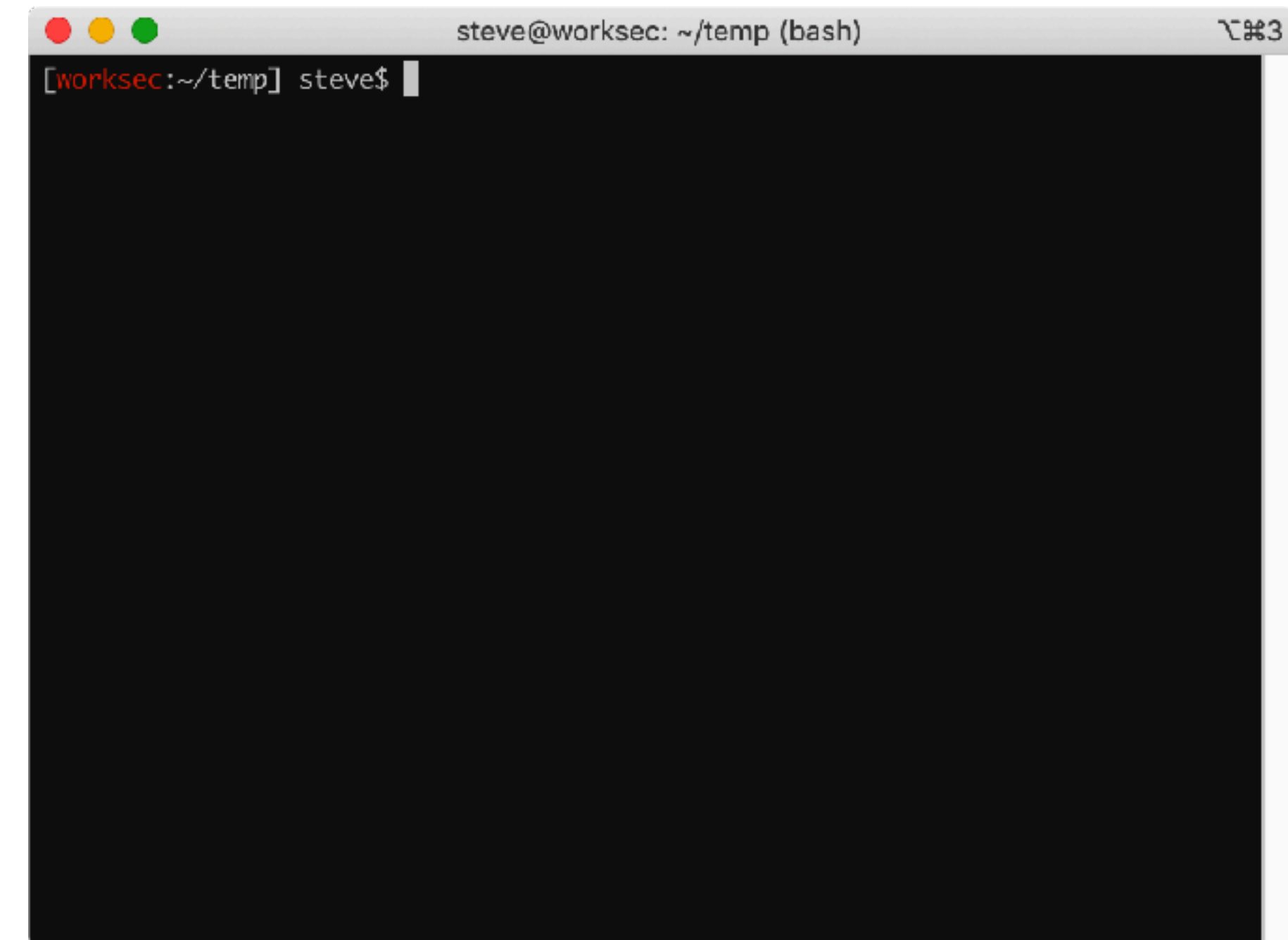
Terminals/terminal emulators

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

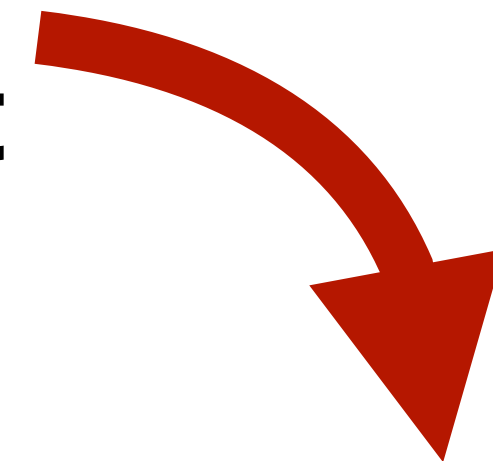
```
[worksec:~/temp] steve$ █
```

**Display
prompt**

Interpreter loop

```
[worksec:~/temp] steve$ ls
```

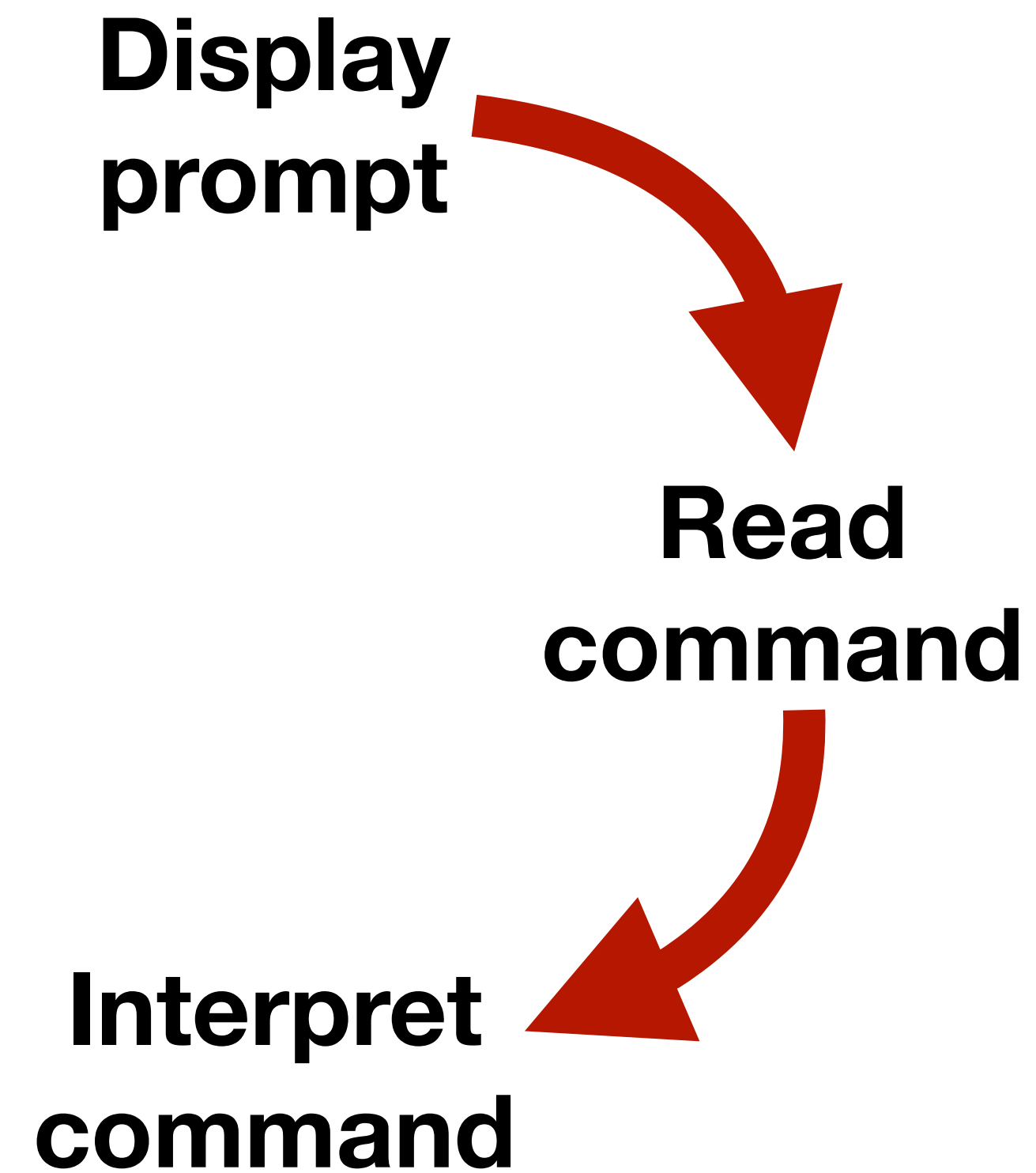
**Display
prompt**



**Read
command**

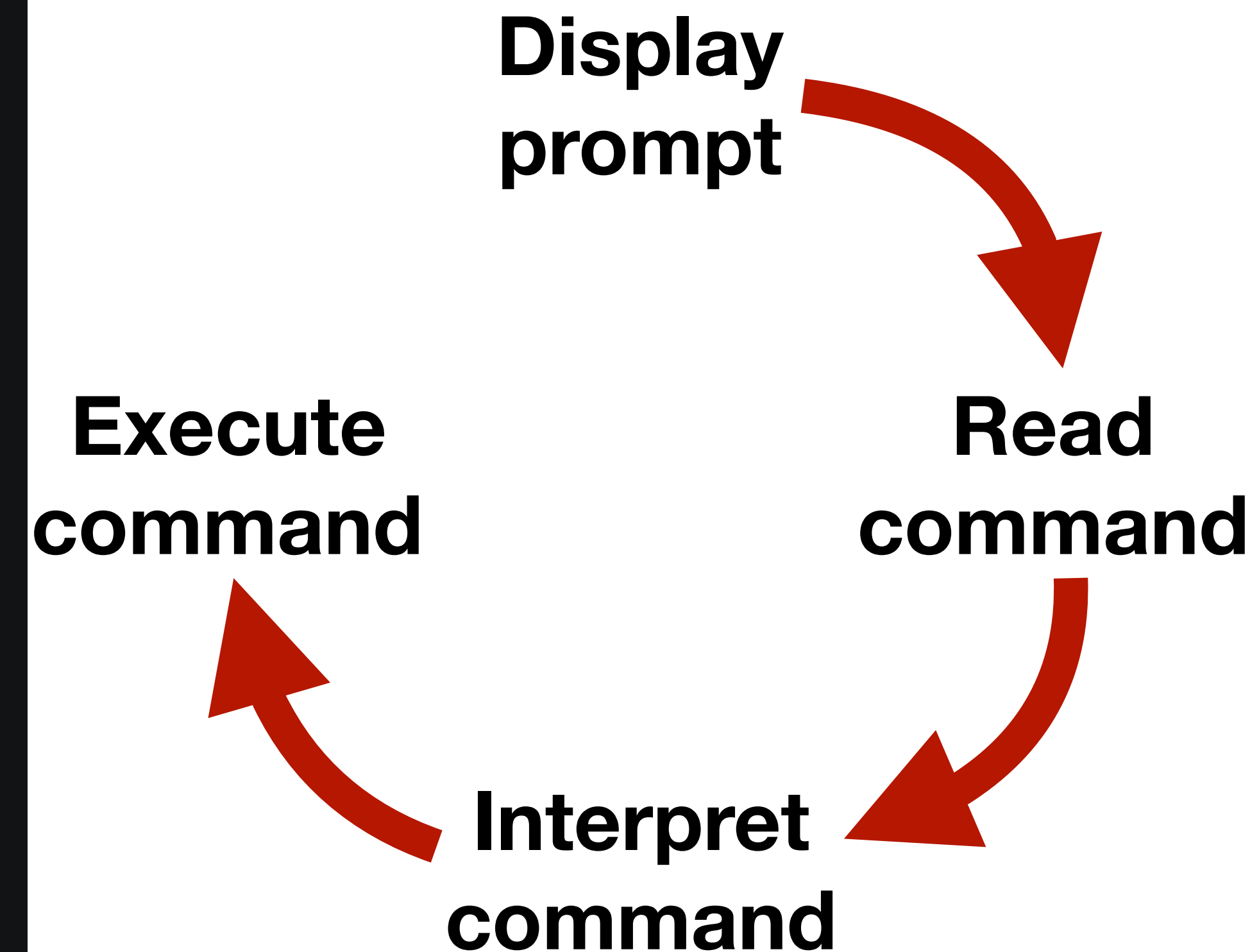
Interpreter loop

```
[worksec:~/temp] steve$ ls
```



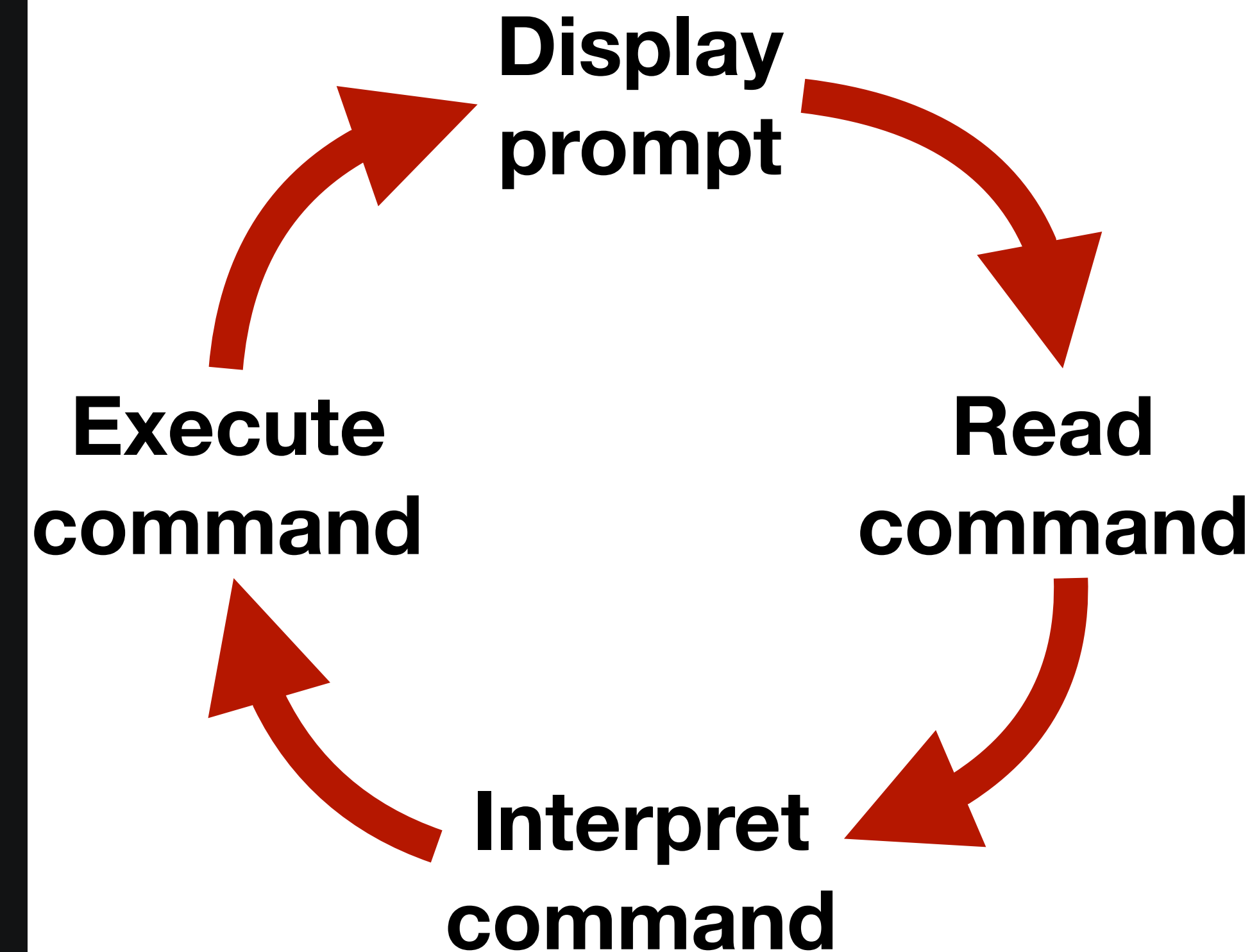
Interpreter loop

```
[worksec:~/temp] steve$ ls
Messages      context-state families.log    olt.s          texput.log
a.c           count.c       families.tex    pickle         twitter
a.cc          count.s       findjumps.c     serial         twitter.dSYM
a.out         crc           foo             serial2        twitter.tar.gz
a.out.ld_AQpFuk crc.c        implicitcatcode.tex silly.hs       unix09.cls
b.s           crypto.py     linenumbr.aux  simd.c         whee.c
b.s-jmp       example.aux   linenumbr.log  ssl.html       whee.o
b.s-ret-cache example.ent   linenumbr.tex  test.aux       whee.s
bug           example.log   luhn.py        test.log       xmodem
cfi           example.pdf   mmap_crash.c   test.pdf       xmodem.c
comparison.h  example.tex  notebook       test.tex       xmodem.dSYM
concat       families.aux olt.cc         texput.aux    xmodem.out
```



Interpreter loop

```
[worksec:~/temp] steve$ ls
Messages      context-state families.log    olt.s          texput.log
a.c           count.c       families.tex    pickle         twitter
a.cc          count.s       findjumps.c     serial         twitter.dSYM
a.out         crc           foo             serial2        twitter.tar.gz
a.out.ld_AQpFuk crc.c        implicitcatcode.tex silly.hs       unix09.cls
b.s           crypto.py     linenumbr.aux  simd.c         whee.c
b.s-jmp       example.aux  linenumbr.log  ssl.html       whee.o
b.s-ret-cache example.ent  linenumbr.tex  test.aux       whee.s
bug           example.log  luhn.py        test.log       xmodem
cfi           example.pdf  mmap_crash.c   test.pdf       xmodem.c
comparison.h  example.tex  notebook       test.tex       xmodem.dSYM
concat       families.aux olt.cc         texput.aux     xmodem.out
[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—the program that launched the program

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

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:~$ pwd
/usr/users/noquota/faculty/steve
steve@clyde:~$ ls /usr/bin/ssh
/usr/bin/ssh
steve@clyde:~$ cd /usr
steve@clyde:/usr$ pwd
/usr
steve@clyde:/usr$ ls bin/ssh
bin/ssh
```

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?

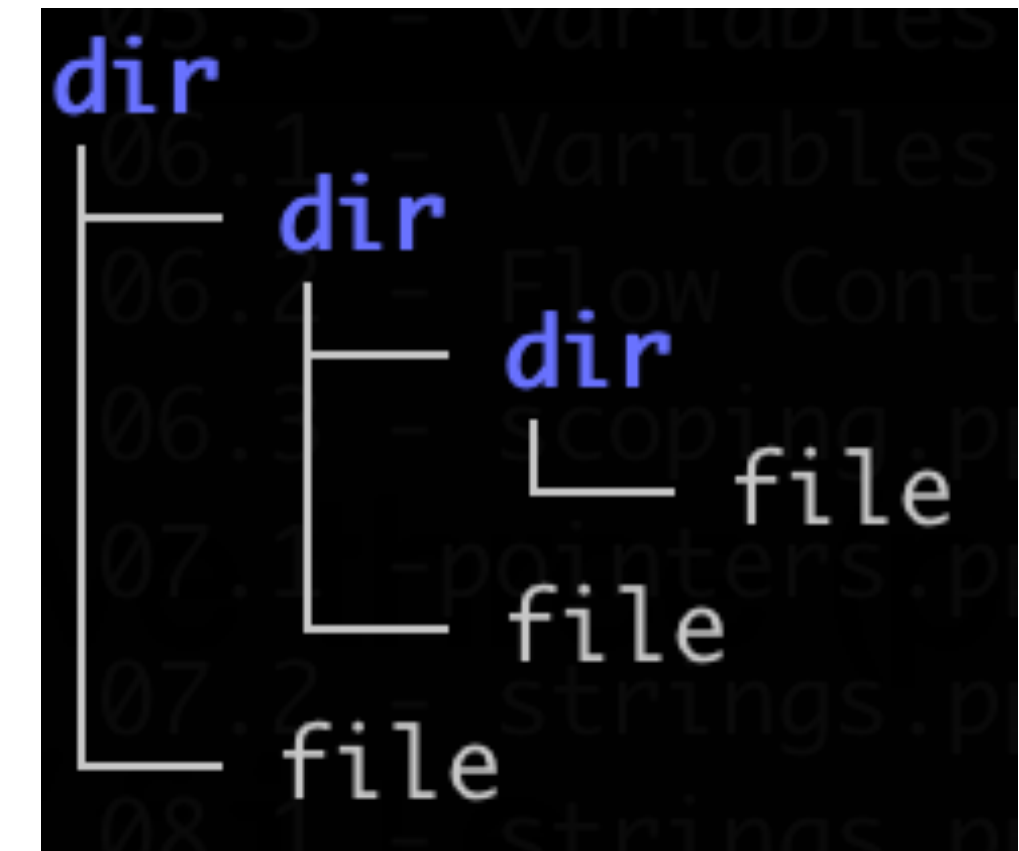
A. `/dir/file`

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

Two special directory entries

Each directory contains two special entries

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

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

Commands

⟨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
 - Usually start with one or two hyphens
- ▶ ⟨arguments⟩ are the things the command acts on
 - Often file paths or server names or URLs

Example: `rm -r foo bar`

Example meaning

▼ `rm(1)` `-r` `foo` `bar`

remove files or directories

`-r, -R, --recursive`
remove directories and their contents recursively

Remove (unlink) the FILE(s).

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

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

- 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

In-class exercise

<https://checkoway.net/teaching/cs241/2020-spring/exercises/Lecture-02.html>

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