Unix

Operating Systems

OS Interaction

- System Calls
  - I/O: open, read, write, lseek, close
  - Processes: fork, pipe, exec, wait, exit
- Shell
  - Allows for the execution of programs
  - redirection, filters, sequential, concurrent

I/O System Calls

```c
int main(int argc, char *argv[]) {
  char buf[128];
  int fd = open("file.txt", O_CREAT | O_RDWR);

  // Reading 128 bytes into buffer
  read(fd, buf, 128);

  printf("I read, and am about to write, the following: %s\n", buf);

  // Writing 128 bytes into file
  write(fd, buf, 128);

  // Seeking to the 64th byte in the file
  lseek(fd, 64, SEEK_SET);

  // Writing 128 bytes into file
  write(fd, buf, 128);

  close(fd);
}
```

Process System Calls

```c
int main(int argc, char *argv[]) {
  int pipefd = pipe();
  int pid = fork();
  if (pid != 0) {
    // this is the parent process
    char *data = "this is my data";

    // write the data into the buffer and wait for pid to exit
    write(pipefd, data, 16);
    wait(pid);
  } else {
    // this is the child process
    char buf[16];

    // when read returns, it contains "this is my data"
    read(pipefd, buf, 16);

    exit(0);
  }
}
```

The Shell

```
Simple commands:
  $ someCommand some arguments here
  $ ls
  $ cat someFile.txt

int main(int argc, char *argv[]) {
  char *programName = parse_name_from_user_input();
  char *programArgs[] = parse_args_from_user_input();

  int pid = fork();
  if (pid != 0) {
    wait(pid);
  } else {
    exec(programName, programArgs);
  }
}
```

The Shell

```
Redirection:
  $ someCommand some arguments here > someFile
  $ ls > output.txt
  $ cat < someFile.txt

int main(int argc, char *argv[]) {
  char *programName = parse_name_from_user_input();
  char *programArgs[] = parse_args_from_user_input();

  int pid = fork();
  if (pid != 0) {
    wait(pid);
  } else {
    exec(programName, programArgs);
  }
}
```

The Shell

```
Redirection:
  // 0 = standard input, 1 = standard output, 2 = standard error
  $ someCommand some arguments here > someFile
  $ ls > output.txt
  $ cat < someFile.txt

int main(int argc, char *argv[]) {
  char *programName = parse_name_from_user_input();
  char *programArgs[] = parse_args_from_user_input();
  char *redirFile = parse_fileName_from_user_input();

  int pid = fork();
  if (pid != 0) {
    wait(pid);
  } else {
    int fd = open(redirFile, O_CREAT | O_RDONLY); // flag depends on <, >
    // set file descriptor 0 or 1 to point to fd’s file table index
    exec(programName, programArgs);
  }
}
```