Pipes

I/O Redirection to Processes...

- What if we wanted to
  - Use output from one program
  - As input to a second program
  - *cmd1* | *cmd2*
- Could just do I/O redirection
  - *cmd1 > tmp_file*
  - *cmd2 < tmp_file*
  - *rm tmp_file*
- However this is
  - Clunky
  - Has high file overhead
  - Need to worry about file naming collisions
- We need pipes

pipe

- #include <unistd.h>
  - int pipe(int pipefd[2])
    - pipefd[0] = read end
    - pipefd[1] = write end
    - returns 0 on success, -1 on error
- Sets up a communication channel between two file descriptors

Pipe Example

```
int fd[2];


if (fork() == 0) {
  //Child (cmd1 | cmd2)
}
else {
  //Parent (Shell)
}
```

- Why should you fork before piping?

Pipe Example

```
int fd[2];

if (fork() == 0) {
  //Child (cmd1 | cmd2)
  pipe(fd);
  if (fork() == 0) {
    //cmd 1 (Writer)
    //Handle fds
    //Execute command
  }
  Else {
    //cmd 2 (Reader)
    //Handle fds
    //Execute command
  }
}
else {
  //Parent (Shell)
}
```

- Why should
  - The writer be the child?
  - The reader be the parent?

Pipe Example

```
int fd[2];

if (fork() == 0) {
  //Child (cmd1 | cmd2)
  pipe(fd);
  if (fork() == 0) {
    //cmd1 (Writer)
    close(STDOUT_FILENO);
    dup(fd[1];
    close(fd[0]);
    close(fd[1]);
    //Execute Command
  }
  else {
    //cmd2 (Reader)
    close(STDIN_FILENO);
    dup(fd[0];
    close(fd[0]);
    close(fd[1]);
    //Execute Command
  }
}
else {
  //Parent (Shell)
  close(fd);
}
```

- How to implement more than one pipe?