Stderr and Stdout

Data Streams

Streams are simply transfer data from one point to another.

The 3 standard streams in Linux are `stdin`, `stdout`, and `stderr`.

- `stdin` takes data from the shell to the program.
- `stdout` and `stderr` take data from the program to the shell.
  - They are functionally equivalent

Streams can be treated the same as files in many cases.

Usage in C

```
fprintf(file, format, ...)
```

You can use `stdout` and `stderr` in place of the file.

Outputting to `stderr` won't stop the program, you'll have to stop the program yourself if that is your intended functionality.

C Code:

```
fprintf(stdout, "%s", "stdout gets piped!\n");
fprintf(stderr, "%s", "stderr doesn't get piped\n");
fprintf(stdout, "%s", "stdout gets piped again!\n");
```

Output:

```
stdout gets piped!
stderr doesn't get piped
stdout gets piped again!
```

Redirects

```
$ ./prog [redirect] output.txt
```

- `>`: Stdout of program is put into file
- `2>`: Stderr of program is put into file
- `&>`: Stdout and stderr is put into file

Can be chained:

```
$ ./ prog > output.txt 2> errors.txt
```

Pipes

```
$ ./prog | grep -n "pipe"
```

Output:

```
stderr doesn't get piped
1:stdout gets piped!
2:stdout gets piped again!
```

Notice: The final output gets misordered since the first and third output lines get piped into grep and resolved with the grep command while the second output line gets resolved with the program.