Software Systems

Day 18 - Interprocess Communication

Interprocess Communication

- How do processes communicate under the hood?
- In UNIX, everything is a file, so the communication is abstracted as file read/writes (mostly).
- We've seen a bit of this through the standard files (stdin, stdout, stderr) and redirection.

File Pointers and File Descriptors

- File pointers and file descriptors are not the same thing.
- File pointers have type FILE* and offer a higher-level interface (e.g., fprintf/fscanf).
- File descriptors have type int and offer a low-level interface through syscalls (read/write).
- But you can convert between the two with `fileno` and `fdopen`.

  -
    ```
    FILE* passwords = fopen("secrets.txt", "r");
    int fd = fileno(passwords);
    FILE* file_ptr = fdopen(fd, "r");
    ```

Pipes in C

- You can set up two file descriptors as a pipe:

```
#include <unistd.h>

int fd[2];
if (pipe(fd) == -1) {
  error("Can't create the pipe");
}
```

- `fd[0]` is the input/read end, and `fd[1]` is the output/write end.
- Anything written to `fd[1]` can be read from `fd[0]`.

Pipes in C

- Pipes can be used with fork() for parent-child communication:

```
int fd[2];
pipe(fd); // Error checking skipped
pid_t pid = fork();
if (pid == 0) write(fd[1], "Hey Ma!", 8);
else {
  char greeting[8];
  read(fd[0], greeting, 8);
}
```

- Communication only goes one way.