

Recitation 4: Unix II

MIT - 6.033

Spring 2022

Henry Corrigan-Gibbs

Plan

- Recitation Qs
- Why this paper
- Processes & fork
- Shell & demo
- Discussion

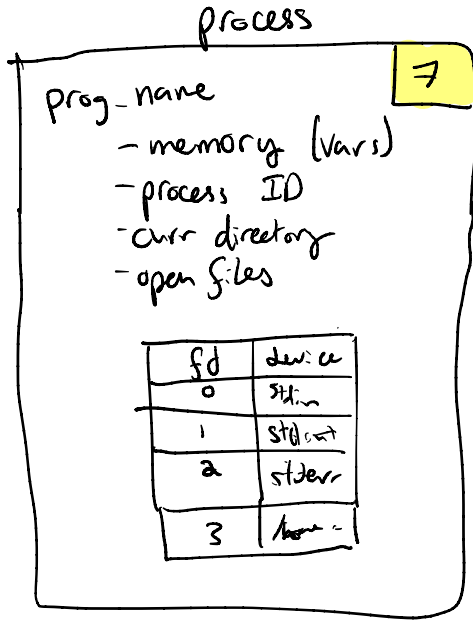
Logistics

- * Hands-on assignment out tomorrow
- * Volunteers for next time

Why this paper?

- Touched on it last week but worth repeating
- Unix is one of the original computer systems
 - ↳ Exemplifier problems need to solve
 - * Naming (hierarchical names)
 - * Modularity / isolation
 - * Abstraction to manage complexity
 - * Virtual memory - each process in own addr space
 - * Processes give rise to concurrency

Processes and Fork



* Explain what is in a process...

* Process of forking

↳ What distinguishes child from parent

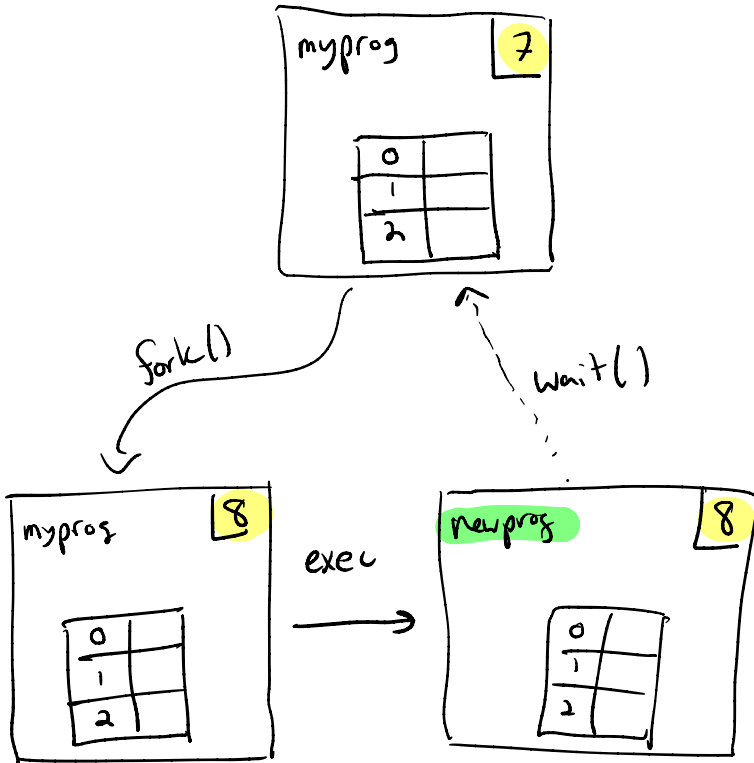
Opening a file

```
int fd = open("file.txt", "r");  
write(fd, "Hello world\n", 12);
```

Special file descriptors

0 = stdin
1 = stdout
2 = stderr

Forking a process



Recitation Qs (Volunteers!)

- 1 What does the shell do?
- 2 How does it work?
- 3 Why is it useful?
- 4 How do you think the Unix developers envisioned their system being used in 2022, if at all?

Shell exercise

What is shell?

- Programmer's / User's interface with the computer
- The way you run programs on a terminal-based machine
- Revolutionary b/c interactive (vs batch)

Unix-like Oses are made to eat and spit out text!

↳ When programming on Unix-like system, it ~~cannot~~ take text as input, spit out text.

Beauty of unix: - Shell is not a "special" program
- Just another program
- Can use your own shell if you prefer

What is Unix without the shell?
e.g. paxemaker, airplane, kiosk, ...

Shell demo

Bash shell

```
cat org.txt.gz | gunzip - | grep stuff | wc
```

```
|cut -w  
-f1
```

```
|sort -r
```

0. Fork & exec

↳ What's the return value of `fork()`?

↳ Problem that child doesn't know its parent's PID?

↳ Isn't it expensive? Copy on write.

↳ What happens if there is a var in parent, then child changes var value?
* Again: copy on write

↳ Problems with implementing `fork()`

↳ random #s ...

↳ shared OS state can get nested up

↳ without `exec`?

Group discussion

To discuss:

* How can two Unix processes communicate with each other?

↳ Example: Word processor process
Printer process

* What are some benefits & shortcomings of the approaches we've covered so far?

- Essentially only via read/write files/sockets.
(later: network, signals, shared memory, ...)

- Pros: Simple! Elegant.

- Cons: Unstructured. Stream oriented
No random

1. Background job

- ↳ Why would you want by job?
- ↳ Very easy to implement!

2. Redirect to file

- ↳ Why would you want this?
- ↳ Write out what this looks like in process diagram
- ↳ dup2

3. Pipe between processes

- ↳ dup2 again
- ↳ Why? Saves storage, blocks writer until reader is ready