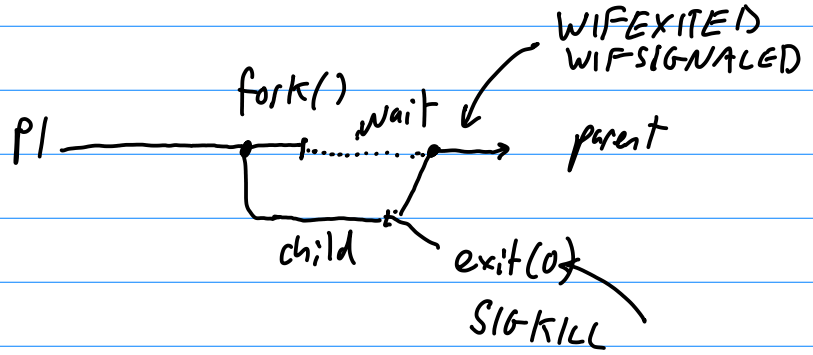
CS 3214 lecture #6 "fork/exec fds & pipes"

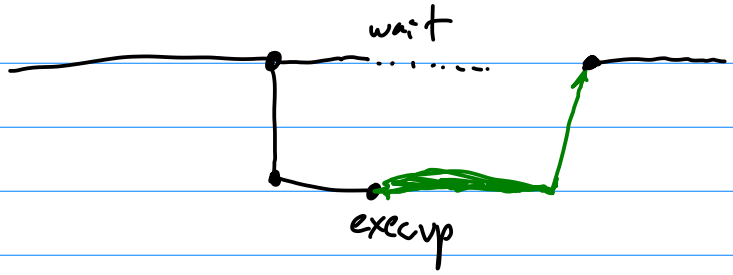
Ex 1, project 1

Recap: fork



1. speed up usage of fork
   parallelization usage

2. "exec" usage of fork



Fork
new process, old program
clone of parent
call once return 2x

exec
same process, new program
everything* is re-initialize
heap, stack regs
→ fds are kept
call once never return

"File descriptors"
how to interact/use kernel object
integers ← low number
handle
file descriptor
↳ sequence of bytes

["everything is a file"]

0 - stdin
1 - stdout
2 - stderr
3

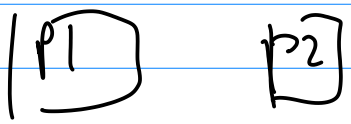
cat foo > file

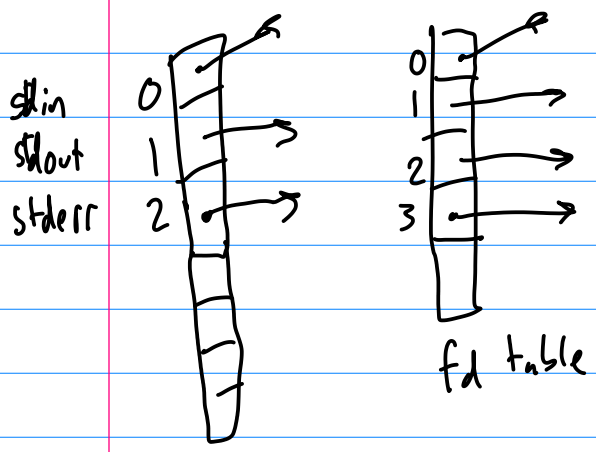
unified* interface

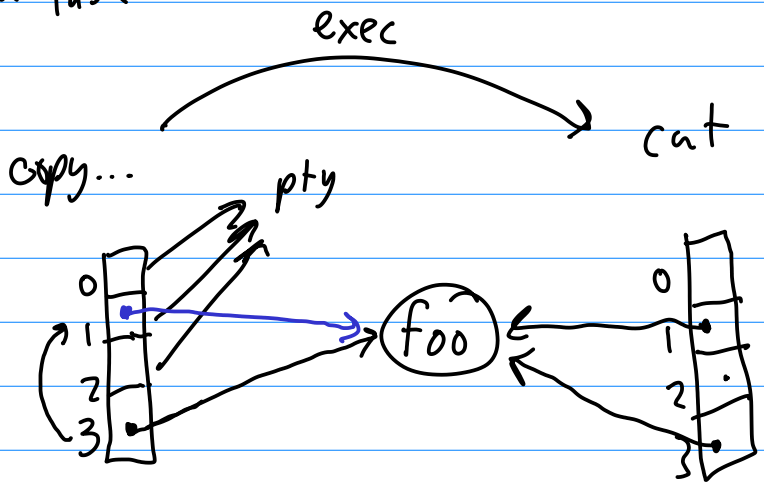
```
int fd = open ("/bin/ls
         read (fd, ...
         write (fd, ...
         close (fd, ...
         lseek (fd
         dup2
```

← "rename / rewire"

types of things the kernel controls
- file on disk
- devices (abstracted)
- socket
- terminals (pty)
- IPC inter process communication
  pipes



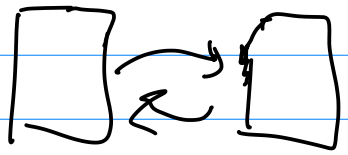




redirection of fd survived exec!

cat foo | grep | wc

Pipes - interprocess communication



kernel managed way for processes to communicate safely

block

What if reader is too fast? block

what if writer is too fast? block