CS61 Lecture 21 — Nov 12, 2019

Process control

- last-time creation (fork())
- this time: changing process program image
  waiting for completion
  communication streams

Fork+EXEC syscall

- take current process image (memory and registers) w/ new process image
  (entrance to main (not fd or other parts of prog.)
- exec runs another process within its defined scoped program

Spawn syscall

- fork and exec in one step
- doesn't copy process image, starts one from scratch

+ processes can't steal resources from another

- Unix prevents process from gaining access to a pipe after creation

Waitpid syscall (pid_t pid, int *wstatus, int 0)

What does process do while waiting for child?

- yield, of course

- blocking system call: calling process yields CPU and does not run till syscall completes

Polling (nonblocking syscall)

- calling process resumes even if system call cannot complete successfully

Waitpid in parent returns status of exited child
→ child can't be reused.

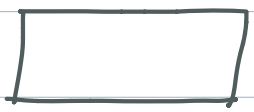
ZOMBIE process: has exited but parent has not called waitpid for it.

Pipe syscall

int
0 success
error

pipe(int pfd[2]);
pfd[0]: read end of pipe
pfd[1]: write end of file

characters written to pfd[1] can be read from pfd[0]



pipe(pfd)
pfd[0]=3, pfd[1]=4
fork()
fork()
close(fpd[0])
close(fpd[1])


close(fpd[0])


close(fpd[0])

u

k


keyboard
monitor