Lab 3 qs - last one

Can’t answer questions about the final, but I can answer stuff about the content

Fd/Pipes

- File Descriptors (special, is_reader, pipe) - max 64 works fine
  - **Fds give information about what kind of device are you interacting with for this lab (console, pipe)**
  - **Keeping track of reference counts is optional here since we don’t have files**
- Pipes are just a circular buffer same as console_buf - max 8k works
  - **But with reference counts to their readers/writers**

Expected Behavior - write

- **Always** want to make progress
  - **don’t block if there are no readers**
- Don’t want to interleave writes, only one writer can write to a pipe at a time, use syncing
- If there is no reader at the beginning you can just return
  - **If its mid-write you can return the number of characters written**

- Same log as console_buf with a circular/bounded buffer

Expected Behavior - read

- **Always** want to make progress
  - **don’t block if there are no writers**
- Don’t want to interleave reads, only one reader can read from a pipe at a time, use syncing
- If there is no writer at the beginning you can just return
  - **If there is and you haven’t read anything - block**
  - **If its mid-read you can return the number of characters read**
- Block only when there are readers and you haven’t read anything yet
  - **Else don’t block**

Managing blocked read/writ(ers)

- Since they are blocked, you need to wake them up when a pipe end has a reference count of 0
  - **So in your decrement() and the corresponding end results in 0 writers or readers, make sure you wake up a writer/reader with a V() on nslot/nelem respectively**
  - **Also free Fd and pipes when their reference counts become 0**

Main functions to change

- Read/Write
- Fork
- Close
- Exit
- InitUserProcess

- And of course the new syscalls
- dup/dup2/pipe

Dup(oldfd)/Dup2(oldfd,newfd)

- Same thing pretty much

- Dup takes in fd to (dup)licate
- Finds the next free new fd index not used
- Copies fd over, handle reference counts for pipes
- Returns newfd

- Dup2 just takes an extra parameter newfd, to make the same as oldfd
- Close newfd if its used
- Rest is same as dup

Pipe

- Creates (malloc) a new pipe struct
- Initializes pipe values
- Find two new fd’s
- Handle/initialize the two new file descriptors and reference counts for pipe
- Put fd’s into user address passed in [fd1,fd2]
- Return 0

Rest of em

- InitUserProcess
  - **Precreate special Fds for 0,1,2 for first PCB**
- Exit
  - **Close all remaining fd's that haven't been closed**
- Close
  - **Resets fd's, decrement counts**
- Fork
  - **Initialize new PCB fields, copy over fd values, handle reference counts**
- Read/write -explained

Error checks

- Return -EPIPE or -EBADF if there are no readers when you try to write
- -EBADF anywhere if a fd is not used/valid
- -EFAULT as always if user-addresses are OOB
- Read man pages for dup/dup2/pipe errors
  (-EMFILE/EBADF/EINVAL)

- Remember to filter bad fd's when taking them as argument
  - **my_fds[41000101]?**
  - **They are user inputted, you don't want to segfault anywhere :)**

Good luck on your final!

- I can answer questions on content or lab3