# Project 4: IPC and Process Management

# Project Info

- Design Reviews: Sunday, 11/9 and Monday, 11/10. Sign up!
- OHs: Thursdays, 4:30-6:30pm
- Due Date: **Saturday**, 11/15, at 11:59pm

# General Notes

- Need to protect critical sections with synch devices (sync.h/c). This project is an exercise in synch mechanisms.
- Use the supplier scheduler (scheduler.c), which uses lottery scheduling. Don't break it!
  - total_ready_priority
  - ready_queue
- Look at the test cases (Robin Hood esp.) to get an idea of how everything fits together

# Implementation Checklist

- do_spawn: create new processes
- do_mbox_*: handful of mbox functions to enable IPC
- Handle keyboard input
  - putchar()
  - do_getchar()
- do_kill: kill a process
- do_wait: wait on a process
- This is a reasonable order in which to complete this project!!

# Spawn

- Kernel has a fixed array of PCBs
- What info do you need to initialize process?
  - PID
  - Allocate a stack
  - Entry point (ramdisk_find)
  - total_ready_priority
  - Do something with the ready_queue

# Message Boxes

- Read the (2) pages in Tanenbaum about message passing
- Literally the bounded-buffer problem
- Reclaim mboxes (refcount)

# Keyboard Input

- Implemented as a message box (initialized on kernel startup)
  - putchar() is a producer (puts character in message box). If buffer is full, discard
  - do_getchar() is a consumer (reads character from message box). You should replace the dummy implementation with your own code.
  - Keyboard interrupt handler is initalized in init_idt(). Keyboard.c translates keyboard signal to an ASCII character for you.

# Kill

- A process should be killed immediately
  - Which queue it is in (ready, blocked, sleeping, etc) doesn't matter-- kill it!
- Do not reclaim locks (this is extra credit)
- Reclaim memory!!
  - Occupied by the PCB
  - Look at robinhood test case to figure out what needs to be reclaimed
- Update total_ready_priority

# Wait

- Waits for a process to exit
  - Blocks until the process is killed or exits normally
- What to add to the PCB to implement this behavior?
- Return -1 on failure, 0 on success

# Hints/Tips

- List of functions to implement is straightforward. But realizing the implementation is tricky!
- Can't use anything in stdio.h/string.h/standard C libraries. Look to util.h and check out any of the header files in the project folder for a helper function you might want.
- Use the tasks script (./tasks robinhoodandlittlejohn) to copy 3 files over to the top-level project directory. Otherwise make will fail.