Inter-Process Communication and Process Management

"Election Edition"

Notes about the Timing

- Monday Nov. 12th - Design Review
  - Signup!
- 11:59p Sunday Nov. 18th – Project Due!

What You'll Be Doing

- Implementing "message boxes"
  - bounded buffer linter-process communication
- Keyboard Input
  - do_getchar()
- Process Management
  - spawn, kill, wait
- This is a reasonable order to do it in!

General Notes

- Still need to worry about interrupts! Use critical sections
  - Think about kernel init especially ! (be careful about leave_critical)
- The supplied scheduler uses lottery scheduling
  - you *may not break it (hint:* total_ready_priority)
- You will need to reclaim memory in this assignment!

Message Boxes

- Look to Tanenbaum (MOS)
- You should use locks and condition variables, not critical section
- You should reclaim mboxes

Keyboard Handlin’

- Use an mbox to capture keystrokes
- getchar() will be interacting with the mbox
- Discard characters when the buffer is full
- Will need to initialize at kernel startup
- The basic IRQ1 interrupt handling is setup in init_idt(), entry.S:irq1_entry and keyboard.c

Spawn

- Collect information for the task
  - Entry point -> look at ramdisk_find()
  - What about field task_type = ?
- Setting up resources and scheduling
  - Allocate a PCB
  - Assign a PID
  - Allocate stacks
  - Remember total_ready_priority !

kill

- A process should be killed immediately
  - Ready, blocked, or sleeping, doesn't matter, it should be killed.
- Other processes should be unaffected
- For starters, don't try to recover locks
- Reclaiming memory is important!
  - Look at the robinhood test case, and think about why it needs to have reclamation

wait

- Allows a process to block until a given process completes execution
- Basically, wake up on kill's and exit's
- What needs to happen to the PCB for efficient blocking?
- Return -1 on failure, 0 on success

Questions!

- Ask them.