CMSC412 DISCUSSION

Project 1 [Due Friday, September 21 @ 6:00pm]

Review:

- Questions about Project 0

Project 1 Requirements

- Add “detached” (background) processes
- Add asynchronous killing of processes
- Add the ability to print the process table (i.e., information about current processes)

Lifetime of a User Process

- The shell [src/user/shell.c] spawns a user process using Spawn_With_Path
- Spawn_With_Path [src/libc/process.c] → Spawn_Program (a.k.a. Sys_Spawn)
- Sys_Spawn [src/geekos/syscall.c] → Spawn
- Spawn [src/geekos/user.c] → Start_User_Thread
- Start_User_Thread [src/geekos/kthread.c] adds the thread to the struct Kernel_Thread list (s_runQueue)

Information about User Processes

- User processes terminate…
  - **…normally**, via Exit (called automatically when main finishes, as you discovered in Project 0)
  - **…abnormally,** via Sys_Kill which is the goal of Project 1
- Parent process can wait via Wait call (in fact they must to avoid a zombie process)
  - …perhaps the parent does not want to Wait on it's children. This is the point of *background* processes.

Implementation – Adding "Detached" Processes

- In /src/user/shell.c:
  - Parse '&'
  - If '&' detected – spawn in background, don't Wait
  - If '&' not detected – spawn normally, do Wait
- In /src/libc/process.c (and process.h):
  - Modify DEF_SYSCALL macro (and Spawn_Program definition) to handle extra parameter
- In /src/geekos/user.c + /src/geekos/syscall.c:
  - Accommodate extra background parameter in Sys_Spawn / Spawn
- Additional Notes:
  - Detached processes *cannot* be Wait()ed on
  - Detached processes *cannot* receive input from Get_Key (and neither can any children of a detached process(!))

Implementation – Killing Processes

- Add kill.c in /src/user/
- In /src/libc/process.c (and process.h):
  - Add Kill() function declaration and macro (wrapper for Sys_Kill)
- In /src/geekos/syscall.c (Sys_Kill):
  - Get the PID of the victim process
  - Lookup the victim's Kernel_Thread (see Lookup_Thread [src/geekos/kthread.c])
  - Dequeue thread from all queues and 'kill' it
    - Run queue, join queues, device queues, etc.
  - The currently running thread can kill itself

Implementation – Printing the Process Table

- Add ps.c in /src/user/
  - Takes information from Sys_PS and prints it (look at Project Specification)
- In /src/geekos/syscall.c (Sys_PS):
  - Prepare a struct Process_Info array (note: this is in *kernel space*)
  - Examine all threads: s_allThreadList [src/geekos/kthread.c], and fill out the above array
    - There are helper functions for traversing this list, use them!
  - Copy array into *user space*: Copy_To_User()
    - This is important! Look at the “Further Reading” to understand why.