Gebze Technical University

Department Of Computer Engineering

CSE 312 /CSE 504
Operating Systems Spring 2020

Homework #02
Due Date: April8th 2020
Interrupts,
Processes,Multiprogramming

In this homework you are going to develop a kernel that will support multi- programming, interrupt handling and just a bit of memory management. Please refer to your course textbook for multi-programming and study SPIM MANUAL for how SPIM architecture handles interrupts. (http://pages.cs.wisc.edu/~larus/SPIM/spim_documentation.pdf)

We have modified the SPIM packageto make it easier for your too add these functionalities. We have added a new function named void SPIM_timerHandler() that gets called every time there is a timer interrupt. The timer interrupt happens about every 100ms.

Your task is to develop a part of the kernel called SPIMOS_GTU_X.s that will perform interrupt handling, multi-programming and context switching. Your new kernel will be loaded as an assembly file. In other words, instead of loading assembly files, you will load SPIMOS_GTU_1.s, SPIMOS_GTU_2.s, or SPIMOS_GTU_3.s

What we expect from your new OS is

1) Implementing these POSIX system calls: fork, waitpid, execve, any other POSIX call that you need. These system calls will be in syscall.cpp. Do not modify any other CPP files from the SPIM package.
2) Loading multiple programs into memory: Kernel will be able to load multiple programs into memory. This operation will be a system call.
3) Handling multi-programming: you need to develop a **Process Table** that will hold the necessary information about the processes in the memory. You should study what **Process Tables** holds. You can read carefully throughout the chapter 2 of the course book or any other onlineresource).
4) Handling Interrupts: Our simulator will generate interrupts, and your kernel will handle and respond by modifying SPIM_timerHandler function in syscall.cpp
5) Perform Round Robin scheduling: Every time a timer interrupt occurs, there is a chance to make a process switch.
6) Whenever a context scheduling occurs, you will print all the information about the processes in process table including but not limited to the entries in the listbelow.

   a. ProcessID
   b. ProcessName,
   c. Programcounter
   d. Stack Pointeraddress

Life-Cycle

You will implement 3 different flavors of MicroKernel(SPIMOS_GTU_1.s, SPIMOS_GTU_2.s, and SPIMOS_GTU_3.s). Don't worry 90 percent of the code is same between the Micro Kernels. We further explain the details below.

When your kernel is loaded your OS will start a process named **init** with **process id 0.** In different Micro Kernels Init process will load programs into memory differently

- In the first strategy **init** process will initialize Process Table, load 3 different programs (listed below) to the memory start them and will enter an infinite loop until all the processes terminate.

- Second strategy is randomly choosing one of the programs and loads it into memory 10 times (Same

I

program 10 different processes), start them and will enter an infinite loop until all the processes terminate.

- Final Strategy is choosing 2 out 3 programs randomly and loading each program 3 times start them and will enter an infinite loop until all the processes terminate.
- When SPIM starts, it immediately loads the **MicroKernel file** given by commandline parameters example EX: **./**`spim –efexceptions.s -file SPIMOS_GTU_1.s`
- For every timer interrupt, **OS** should handle the interrupt and perform round robin scheduling.
- Your programs finish execution it will acknowledge its termination by calling POSIX **PROCESS_EXIT**.
- Emulator will shut down only after all the programs in memory terminate.

Assembly Files

You will supply us with three different Assembly Files.

- BinarySearch.s
- LinearSearch.s

which were delivered to you with first homework, Update the System call's.

Ex;
Input : {10, 20, 80, 30, 60, 50, 110, 100, 130, 170}
x = 110;
Output : 6

Input : {10, 20, 80, 30, 60, 50,
110, 100, 130, 170}
x = 175;
Output : -1

- Collatz.asm: You are going to find collatz sequence for each number less than 25. You can find information about (Collatz conjecture on internet). For each number you will show the number being interested in, and its collatz sequence and go to next number **Ex Output;**7: 22 11 34 17 52 26 13 40 20 10 5 16 8 4 21

Your homework template includes several code and sample files. Here is a description for them. Do not change spim files except syscall.cpp and syscall.h and Do not send these files with your homework. Study these files to understand SPIM.

syscall.h : sample header for SPIM OS. You can rewrite this file
syscall.cpp : sample implementation for SPIM OS. You will rewrite this file
SPIMOS_GTU_1.s, SPIMOS_GTU_2.s, and SPIMOS_GTU_3.s: OS Kernel Flavors
LinearSearch.asm, Collatz.asm, BinarySearch.asm : Test Programs,
README : sample running instructions and report your work briefly

Pay attention to file naming Tips &

Tricks

1. You should study what to do, when an interrupt occurs.

II

2. You should study what to do, when context scheduling happens (What to save, what to update).
3. Start early, code often!!!
4. Do not forget to change Process States during context switching
5. Remember Your OS can access any part of the memory itwants.
6. During interrupt handling, be careful about interrupts happening again.
7. Please inspect the interrupt code and changes in the run.cpp to understand it.

General Homework Guidelines

1. No cheating, No copying, No peaking to other peoplehomework
2. Follow the instructions verycarefully.
3. Send required files only. Do not share your whole file system withus.
4. If you fail to implement one of the requirements, leave it be. Do not send an emptyfile
5. Respect the file names! Our HW grading is case-sensitive.
6. Failing to comply any of the warnings above will result in getting a **0** for your current homework.

Homework Instructions

1. Download and Install Vmware Player from Officialsite.
2. Download and install our virtual machine from the same location as HW1 https://drive.google.com/open?id=1YppX3lNkyTsHV_lvA4w9TomNCUkpLeEg
3. Download the SPIM package from the same location as HW1, replace run.cpp, syscall.h and syscall.cpp with our source codes.