Gebze Technical University
Department Of Computer Engineering
CSE 312 /CSE 504
Operating Systems
Spring 2019

Homework #03
Due Date: May 3th 2019
Interprocess Communication, Synchronization

In this homework, we will use our 8080 Emulator with a more advanced OS that supports simple Inter-process communication and synchronization . As you recall, in our last homework we were able to enhance our OS with multitasking capabilities. Since we did understand this concept. We will explore another crucial feature of modern operating systems which comes very handy. By your hands(Literally) our OS will support synchronization and communication between two rivaling processes. For IPC, your operating system will support Message boxes. You can think these boxes as mailboxes in your homes. Processes can use these boxes in order to facilitate interactions among themselves.

MAILBOX SPECIFICATION

Your OS can hold arbitrary number of mailboxes during its runtime, each mailbox should have at least one owner and it is created by the operating system on demand of a user process. Any user processes can access to the given mailbox if they have the **handle** of it. **Handle** can be thought as an unique integer between 0 and 255.

Mailbox Structure:

Mailboxes have fixed 50 byte data buffer size (It can hold 50 items at most.) and start from address 0x2000D. Your mailbox also includes 1 byte-length one mutex and two semaphores which will occupy the first three bytes, Thus a mailbox have a length of 53 byte. Values of these bytes are initialized to zero by the OS . Thus a mailbox implementation should support synchronization by spesifications. Mailbox starting at 0x2000D will have the id 1, next one will start at at 0x2053D and will have id 2 an so on...

System Calls: You will keep the previous system calls from the first homework and add the new ones below:

Call	**Params**	**Function**	**Cycle**
RAND_INT	REG A = 12	Returns random integer into Reg B	60
WAIT	REG A = 13, REG B = ID of MB, REG C = Address of Cond Var	Waits on the condition var	200
SIGNAL	REG A = 14, REG B = ID of MB, REG C = Address of Cond Var	Releases the lock on the variable	200

OBJECTIVE

Your objective is to code to process that will work cooperatively in order to achieve a coherent result. First process will generate random numbers between 0 and 255 in a loop. Loop will terminate 400. You can use loop with in a loop to implement this objective. Generated number will be placed into the mail box unless message box is fullor being used by another process. If that is the case process will wait until a slot opening or mail box is being used by another process.
Second process will connect to same mail box and read the last numbers entered in the mail box. Readed values will be copied in to a local(process wide) array and will be deleted from the mail box. If mail box is empty or mail box is being used by another process this process will wait until it is released or there is an element in the box. In every context switching event your operating system will save the mailbox contents and the local list contents to files called **mailbox.txt** and **locallist.txt.** These processes should have different assembly files, named **sender.asm** and **receiver.asm.**

You will supply us with two different implementation. In the first implementation you will show us the race condition between two processes without using the synchronization mechanism. And in the second one you we expect you to resolve the issue with using your version of mailbox synchronization.

PS: You can directly access to the mail box by calculating the static addresses of user processes and the address of the mailbox. Remember that wait call may cause in corrections and updates in the user process table and require a context switching event.

Tips & Tricks

1. If you have failed to implement HW2, We will supply you with a OS implementation that supports Multitasking. Otherwise you can continue using your own code base developed during the HW2.
2. Study the given code and try to understand it
3. Start early,code often!!!
4. Since Mailbox is a resource your process table have to have a reference to it
5. Remember Your OS can access any part of the memory it wants.
6. **We have changed the Homework guidelines, pay attention to those instructions.**
7. Do **not** use **HLT in user processes**, Do **not** use **PCHL in user processes**, remember this is a special instruction.
8. We expect you to deliver asm files, for processes and the both kernel implementations and rest of the necessary files to compile and run your homeworks. Remember **do not send** us any com files.

GENERAL HOMEWORK GUIDELINES

1. No cheating, No copying, No peaking to other people homework

2. Follow the instructions very carefully.
3. Send required files only. Do not share your whole file system with us.
4. If you fail to implement one of the requirements, leave it be. Do not send an empty file
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.
7. We expect you to provide