**EECE 4029: Operating Systems**

**Spring 2021**

**HOMEWORK I**

**Given: Jan. 29, 2021**

**Due: Feb. 10 (Wednesday), 2021 (NO LATER THAN 11:59PM)**

**SUBMISSION INSTRUCTIONS:**

**SUBMIT ONLY ON-LINE FILES ON CANVAS BEFORE MIDNIGHT.**

**NO HARD COPY WILL BE ACCEPTED.**

**Hints on programming problem #7:**

**1. UNIX/LINUX Environments**

a. Linux Virtual Machine is available at the text book website:

<http://www.os-book.com/>

b. The UCFileSpace system includes a Unix shell environment:

<https://ucfilespace.uc.edu/wiki/services>

**2. If you do not know how to execute your code, please refer to "README" and "makefile" files provided in the source code package.**

1. Describe the differences between symmetric and asymmetric multiprocessing. What are three advantages and one disadvantage of multiprocessor systems?

2. What are the two models of inter-process communication? What are the strengths and weaknesses of the two approaches?

3. What is the main advantage of the microkernel approach to system design? How do user programs and system services interact in a microkernel architecture? What are the disadvantages of using the microkernel approach?

4. What is the main advantage of the layered approach to system design? What are the disadvantages of using the layered approach?

5. Describe the differences among short-term, medium-term, and long-term scheduling.

6. Explain the circumstances when the line of code marked printf("LINE J") in Fig. 2 is reached.



Fig. 2 Code for Question 6

7. Programming problem:

An operating system’s pid manager is responsible for managing process identifiers. When a process is first created, it is assigned a unique pid by the pid manager. The pid is returned to the pid manager when the process completes execution, and the manager may later reassign this pid. Process identifiers are discussed more fully in Section 3.3.1 of the textbook. What is most important here is to recognize that process identifiers must be unique; no two active processes can have the same pid.

Use the following constants to identify the range of possible pid values:

#define MIN PID 300

#define MAX PID 5000

You may use any data structure of your choice to represent the availability of process identifiers. One strategy is to adopt what Linux has done and use a bitmap in which a value of 0 at position i indicates that a process id of value i is available and a value of 1 indicates that the process id is currently in use.

Implement the following API for obtaining and releasing a pid:

• int allocate map(void)—Creates and initializes a data structure for representing pids; returns -1 if unsuccessful, 1 if successful

• int allocate pid(void)—Allocates and returns a pid; returns -1 if unable to allocate a pid (all pids are in use)

• void release pid(int pid)—Releases a pid

**Please submit your source code for this question.** **To receive full credits for this problem, please include screen shots of your test results.**