Homework 1
Released: Sep 26, 2024
Due date: Oct 3, 2024, 11:59pm
Total points: 100

Problem 1 (10 points) What is the meaning of the term busy-waiting? What other kinds of waiting are there in an operating system? Can busy waiting be avoided altogether? Explain your answers.

Problem 2 (10 points) Explain why implementing synchronization primitives by disabling interrupts is not appropriate in a single-processor system if the synchronization primitives are to be used in user-level programs.

Problem 3 (10 points) Given the following program that uses three memory segments in an address space as described in class (code segment, data segment, and stack segment):

```
char a[100];
main(int argc, char ** argv)
{
   int d;
   static double b;
   char *s = "boo", * p;

   p = malloc(300);
   return 0;
}
```

Identify the segment in which each variable resides and indicate if the variable is private to the thread or is shared among threads.

Problem 4 (10 points) Which of the following instructions should be privileged:

a. set the value of a timer
b. read the clock
c. clear memory
d. issue a trap instruction
e. turn off interrupts
f. modify entries in device-status table
g. switch from user to kernel mode
h. access I/O device

Problem 5 (10 points) Given the following piece of code:

```
main(int argc, char ** argv)
{
        int child = fork();
        int c = 5;

        if(child == 0)
        {

                c += 5;
        }
        else
        {
                child = fork();
                c += 10;
                if(child)
                {
                     c += 5;
                }
        }
}
```

How many different copies of the variable *c* are there? What are their values?

Problem 6 (10 points) Assume that a system has multiple processing cores. For each of the following scenarios, describe which is a better locking mechanism - a spinlock or a mutex lock where waiting processes sleep while waiting for the lock to become available:

- The lock is to be held for a short duration.
- The lock is to be held for a long duration.
- A thread may be put to sleep while holding the lock.

Problem 7 (10 points) A multithreaded web server wishes to keep track of the number of requests it services (known as ***hits***). Consider the two following strategies to prevent a race condition on the variable hits. The first strategy is to use a basic mutex lock when updating hits:

```
        int hits;
        mutex lock hit lock;

        hit lock.acquire();

        hits++;
        hit lock.release();
```

A second strategy is to use an atomic integer:

```
        atomic t hits;

        atomic_inc(&hits);
```

Explain which of these two strategies is more efficient.

Problem 8 (10 points) Servers can be designed to limit the number of open connections. For example, a server may wish to have only N socket connections at any point in time. As soon as N connections are made, the server will not accept another incoming connection until an existing connection is released. Illustrate how semaphores can be used by a server to limit the number of concurrent connections.

Problem 9 (20 points) Write a multithreaded program that outputs prime numbers. This program should work as follows: The user will run the program and will enter a number on the command line. The program will then create a separate thread that outputs all the prime numbers less than or equal to the number entered by the user.