HW1: This assignment is due by Monday, September 27th, midnight. Please submit to Brightspace.
NOTE: You must submit the questions with your solutions. No handwritten work accepted.

Homework Questions (1 – 12 worth 25% of grade)

1) What is stored in the Systems Call Table?
    a. The addresses of system calls

2) Why are system calls provided?
    a. System calls provide a way for programs running in User space to communicate with the OS

3) What is the purpose of a Trap instruction?
    a. A trap instruction allows a context switch from user mode to kernel mode to happen.

4) Would a divide by 0 error generate a non-maskable interrupt? Why or why not?
    a. A divide by 0 error would generate a maskable interrupt, meaning it can be handled by the OS because it's not a critical or hardware level error.

5) What is the last register restored in a context switch and why?
    a. The program counter is restored, because it is the next instruction to execute. During a context switch, the state is saved. This makes sure that when the CPU starts executing instructions again it will be in the correct location.

6) Consider a single CPU system. What is the OS doing while a user process us executing?
    a. The OS is still monitoring and managing the execution process

7) What does a process become when it is not executing?
    a. Blocked, waiting, or ready

8) How is it ensured that a process stays within its own address space?

   a. Memory Protection

9) What are five major activities of an operating system regarding process management?

   a. Process scheduling
   b. Process creation and termination
   c. Process Synchronization
   d. Context switching
   e. Process Communication

10) What are three major activities of an operating system regarding memory management?

   a. Allocation
   b. Deallocation
   c. Protection

11) Which of the following instructions should only be executed in kernel mode (please circle or highlight):

a) [✓ Set the system timer.]

b) [✓ Turn off interrupts.]

c) Issue a trap instruction.

d) [✓ Access a device driver.]

e) Write to the terminal.

f) Divide by 0.

12) Write a C program that creates a new process with the fork() system call. After the fork(), the parent process should write a message stating that it is the parent and the child should write out a message saying it is the child. Please provide your code below.

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>

int main(){
    int pidd;
    pidd = fork();

    if(pidd == 0){
        printf("I am the child!\n");
    }
    else{
        printf("I am the parent!\n");
    }

}
```

Write a very basic shell: part 1 (75% of grade)

Write a C program that implements the beginnings of a simple shell. Your program should implement an infinite loop that prints out a prompt for the user and waits for the user to enter a program name. It then uses the read system call to read the program name off the command line. It then spawns a child process using fork. The child process should print out the name of the requested program, set up the argument vector and call execv to start the execution of the requested program. The child process should make sure that the execv system call did not fail. If it does fail, it should print out an error message and exit.

The parent process should wait until the child process completes its execution, go back to the top of the loop and repeat the process.

For testing purposes only attempt to execute programs in the current directory. As noted above, if the program does not exit the child should print out an error message and exit.

DO NOT INCLUDE TEST PROGRAMS. We will do our own testing of your shell with programs in our directory.

NOTE: you can assume there are no parameters to the program name entered on the command line.