CSC 139 Operating System Principles
Homework 1

Fall 2019

Posted on Sep. 27, due on Oct. 6 (11:59 pm). Write your own answers. Late submission will be penalized (turn in whatever you have).

Exercise 1. (10%) What are the differences between user-level and kernel-level threads? Under what circumstances is one type better than the other?

Exercise 2. (OSC 3.11) (10%) Including the initial parent process, how many processes are created by the program shown below? Justify your answer.

```
#include <stdio.h>
#include <unistd.h>
int main()
{
    int i;
    for (i = 0; i < 4; i++)
        fork();
    return 0;
}
```

Exercise 3. (10%) Register state is considered to be per-thread rather than shared across an entire process. Why?

Exercise 4. (10%) Explain why system calls are needed to set up shared memory between two processes. Does sharing memory between multiple threads of the same process also require system calls to be set up?

Exercise 5. (10%) Describe the similarities and differences of doing a context switch between two processes as compared to doing a context switch between two threads in the same process.

Exercise 6. (OSC 3.13) (10%) Using the program below, identify the values of pid at lines A, B, C, and D. (Assume that the actual pids of the parent and child are 2600 and 2603, respectively.)

```c
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
int main()
{
pid_t pid, pid1;
    /* fork a child process */
    pid = fork();
    if (pid < 0) { /* error occurred */
        fprintf(stderr, "Fork Failed");
        return 1;
    }
    else if (pid == 0) { /* child process */
        pid1 = getpid();
        printf("child: pid = %d",pid); /* A */
        printf("child: pid1 = %d",pid1); /* B */
    }
    else { /* parent process */
        pid1 = getpid();
        printf("parent: pid = %d",pid); /* C */
        printf("parent: pid1 = %d",pid1); /* D */
        wait(NULL);
    }
    return 0;
}
```

Exercise 7. (OSC 3.16) (10%) Using the program shown below, explain what the output will be at lines X and Y.

```c
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
#define SIZE 5
int nums[SIZE] = {0,1,2,3,4};
int main()
{
int i;
pid_t pid;
    pid = fork();
    if (pid == 0) {
        for (i = 0; i < SIZE; i++) {
            nums[i] *= -i;
            printf("CHILD: %d ",nums[i]); /* LINE X */
        }
    }
    else if (pid > 0) {
        wait(NULL);
```

```
        for (i = 0; i < SIZE; i++)
            printf("PARENT: %d ",nums[i]); /* LINE Y */
    }
    return 0;
}
```

Exercise 8. (OSC 4.17) (10%) Consider the following code segment:

```
pid_t pid;
pid = fork();
if (pid == 0) { /* child process */
    fork();
    thread_create( . . .);
}
fork();
```

Answer the following questions and justify your answers.

- How many unique processes are created?
- How many unique threads are created?

Exercise 9. (20%) Five batch jobs A through E, arrive at a computer center at almost the same time. They have estimated running times of 11, 6, 2, 4, and 8 minutes. Their priorities are 3, 5, 2, 1, and 4, respectively, with 5 being the highest priority. For each of the following scheduling algorithms, determine the average process turnaround time. Ignore context switching overhead.

- Round-Robin with time quantum = 1 minute
- Priority scheduling
- First come, first served (run in order 11, 6, 2, 4, 8)
- Shortest job first

For Round-Robin scheduling, assume that the system is multiprogrammed, and that each job gets its fair share of the CPU. For other scheduling algorithms, assume that only one job runs at a time, until it finishes. All jobs are completely CPU bound.

Please complete the following survey questions:

1. How much time did you spend on this homework?
2. Rate the overall difficulty of this homework on a scale of 1 to 5 with 5 being the most difficult.
3. Provide your comments on this homework (e.g., amount of work, difficulty, relevance to the lectures, form of questions, etc.)