

Homework Assignment #4

Due: Sunday, September 27, 2021

1. What are two differences between user-level threads and kernel-level threads? Under what circumstances is one type better than the other?
2. What resources are used when a thread is created? How do they differ from those used when a process is created?
3. Which of the following components of program state are shared across threads in a multithreaded process?
 - a. Register values
 - b. Heap memory
 - c. Global variables
 - d. Stack memory
4. Consider the following code segment:

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

 - a. How many unique processes are created?
 - b. How many unique threads are created?

5. The program shown below uses the Pthreads API. What would be the output from the program at LINE C and LINE P?

```
#include <pthread.h>
#include <stdio.h>
#include <types.h>
int value = 0;
void *runner(void *param); /* the thread */
int main(int argc, char *argv[])
{
    pid_t pid;
    pthread_t tid;
    pthread_attr_t attr;
    pid = fork();
    if (pid == 0) { /* child process */
        pthread_attr_init(&attr);
        pthread_create(&tid,&attr,runner,NULL);
        pthread_join(tid,NULL);
        printf("CHILD: value = %d",value); /* LINE C */
    }
    else if (pid > 0) { /* parent process */
        wait(NULL);
        printf("PARENT: value = %d",value); /* LINE P */
    }
}

void *runner(void *param) {
    value = 5;
    pthread_exit(0);
}
```

6. The Fibonacci sequence is the series of numbers 0, 1, 1, 2, 3, 5, 8, Formally, it can be expressed as:

$$fib_0 = 0$$

$$fib_1 = 1$$

$$fib_n = fib_{n-1} + fib_{n-2}$$

Write a multithreaded program that generates the Fibonacci sequence. This program should work as follows: On the command line, the user will enter the number of Fibonacci numbers that the program is to generate. The program will then create a separate thread that will generate the Fibonacci numbers, placing the sequence in data that can be shared by the threads (an array is probably the most convenient data structure). When the thread finishes execution, the parent thread will output the sequence generated by the child thread. Because the parent thread cannot begin outputting the Fibonacci sequence until the child thread finishes, the parent thread will have to wait for the child thread to finish.