Some exercises on Threads

1. Modify the program thread3.c such that it uses 8 threads rather than 4 threads. Each thread must also print the content of the variables mytid, sum and mysum after the for loop but before calling the lock synchronization primitive. Now try again, this time print the values of these variables immediately after the instruction sum = sum + mysum;. What differences do you see and how can you explain those differences.

2. The program ComputePrimeNumber.c in the folder Threads-C-programs is a C program that finds the prime number in a range from 1 to N. Rewrite this program using at least two new threads to perform the same task concurrently.

3. The program MatrixMultiplication.c in the folder Threads-C-programs is a C program that multiplies two square 4×4 matrices A and B and stores the result in a matrix C. Write a program that will use two new threads to perform the matrix multiplication. One thread will compute the first 2 columns of C while the other computes the last two columns of C. Do you need the lock synchronizaton primitive in your implementation? Do you need two different functions in your program to implement the matrices multiplications? Does your design exploit functional or data parallelism?

4. The program thread3-pro.c creates two child processes, each of these two processes creates two threads (total four threads). There is a global declaration "int sum=0". Together, among the 3 processes and 4 threads, how many "sum" variables there are?

5. List the fields of a thread control block (TCB) and explain the purpose of each field.

6. Explain why each thread control block has entries (fields) to store the contain of its run time registers but does not have entries to store the contain of the dynamically allocated variables (variables that are allocated memory on the heap section of a process).

7. Unlike static variables, threads do not share their stack with other threads. Why then the stack is not saved in the TCB when a context switch is performed (thread is de-scheduled)?

8. Can a thread write into the memory stack of another thread in a same process?

9. 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?

10. Can a thread forks a new process? If yes, what will be the parent of the new process?

11. In a multi-threaded user process, if a user thread T (which is not detached) calls pthread_exit(), thread T will become zombie. When can the user space stack of thread T get deleted?

12. Assume a thread t_1 creates a child thread t_2. Does both threads share the same address space?

13. What are some of the resources used when a thread is created? How do they differ from those used when a process is created?

14. Assume the image of a process is swapped out from main memory to the external memory. The process has 3 active threads, two in the ready queue and one in the running state. What will happen to those 3 threads?

15. What are two differences in terms of context switch and scheduling between user-level threads and kernel-level threads?

16. Describe some of the actions taken by a kernel to context switch 1- Among threads; 2- Among processes

17. Describe some of the actions taken by a kernel to context-switch between kernel level threads

18. What are some of the resources used when a thread is created? How do they differ from those used when a process is created?