OS ASSIGNMENT2 PART 1-

Ayush Mahant
2019353

Explaining the fork file-

The file is named as “throughfork.c”.
At the starting, the appropriate packages have been included. Then declare and initialize the global variable with value 10.
Now, in the main, I create a child process using a fork,and I have the pid.
If the pid < 0 then , the child process is not created. If pid==0, then the child process is created, and the child process is executing. The condition for the global variable is to reach -90 from 10 which is done using the while loop. Then the parent process is executed,with a while loop for global variable to reach 100 and then waitpid. Then ,I exit using exit system call.

Explaining the thread file-

The filename is “throughthread.c”.
The file is named as “throughfork.c”.
At the starting, the appropriate packages have been included. Then declare and initialize the global variable with value 10.
Now, in the main, using pthread_create, a child thread is created. In the child thread, the global variable is reaching a value -90 through while loop.
After the child thread is created, there is a while loop for the global variable to reach 100.
Then there is pthread_join, to wait for the execution of child thread.

Reasons behind the differences-

Threads are not stand-alone processes so they access global variables in a global manner. The global variable cannot be accessed in a global manner in processes. Each process, access the global variable uniquely .
So in the child process, the global variable starts from 10 to -90 ,and in the parent process, it starts from 10 to 100.
So in the child thread and the parent thread , the global variable is shared so it increments or decrements concurrently. In the output we are taking the processes and the threads are running concurrently.

In threads race condition occurs when two or more threads can access shared data and they try to change it at the same time.Now the thread scheduling algorithm is available to swap between threads at any time, we do not know the order in which the threads access the shared data. The result of the change in data is dependent on the thread scheduling algorithm.
So basically what we see in threads, is that if the global value is incremented upto for e.g, 50 in parent, the starting value of the global variable in the child thread should be 50, but it isn't. THIS is the race condition.

The only significant difference here is that the global variable in the child process starts from 10 and in the process process from 10. In child thread and the parent thread, the global variable is globally shared. Also notice that the child process and the parent process always start from 10. In the child thread and the parent thread, it is random between -90 and 100.