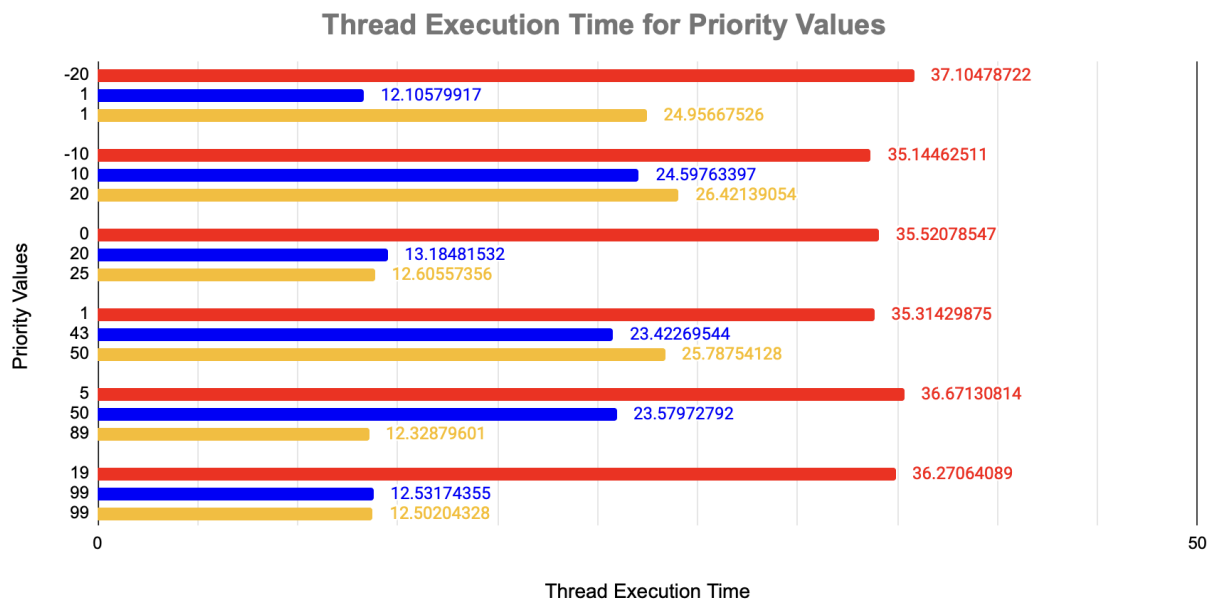# OS Assignment 2 - Q1

1.1 Thread Scheduling

The program creates 3 threads in parallel in order to run the three count functions parallely. Thread A calls the `countA()` function, thread B calls `countB()`, and thread C calls `countC()`. Thread A uses `SCHED_OTHER` scheduling policy, thread B uses `SCHED_FIFO` scheduling, and thread C uses `SCHED_RR` scheduling.

Each count function creates a `struct sched_param` to set the scheduling priority of each thread. Specifically for the thread that uses the `SCHED_OTHER` scheduling policy, `nice()` is called in order to set the nice value (-20 to +19). Other scheduling methods take priority values (+1 to +99) through the `struct sched_param`'s `scheduling_priority` attribute.

Time for each counting loop is measured within the `countX()` functions before the for loop starts, and immediately after the for loop ends, using the `clock_gettime()` function. The obtained times at different combinations of priority values for the three scheduling policies are as follows (red → `SCHED_OTHER`, blue → `SCHED_FIFO`, yellow → `SCHED_RR`):

Thread Execution Time for Priority Values

Priority Values	Thread Execution Time
-20 (red)	37.10478722
1 (blue)	12.10579917
1 (yellow)	24.95667526
-10 (red)	35.14462511
10 (blue)	24.59763397
20 (yellow)	26.42139054
0 (red)	35.52078547
20 (blue)	13.18481532
25 (yellow)	12.60557356
1 (red)	35.31429875
43 (blue)	23.42269544
50 (yellow)	25.78754128
5 (red)	36.67130814
50 (blue)	23.57972792
89 (yellow)	12.32879601
19 (red)	36.27064089
99 (blue)	12.53174355
99 (yellow)	12.50204328

(x-axis: Thread Execution Time, range 0 to 50)

1.2 Process Scheduling

The program creates 3 processes by calling `fork()` through nested if-else statements. The child process from each `fork()` call uses the `execl()` system call to execute three different bash scripts for three different processes.

Each bash script executes the commands to modify the `.config` for the kernel to the custom configuration (as required), and then compiles the modified kernel. The three child processes run parallely and take different scheduling policies; the first child process uses `SCHED_OTHER`, the second uses `SCHED_FIFO`, and the third uses `SCHED_RR` scheduling policies. The

scheduling policies are set using the `sched_setscheduler()` system call. The times for the execution of each process are measured by calling `clock_gettime()` before the parent forks, and after the wait() call ends. The `SCHED_OTHER` process uses the `nice()` call to set the nice value, while the other processes take priority values using the `struct sched_param`. The program was then executed, which thus called the bash scripts that compile three different copies of the stock kernel.

Stock Kernel Compilation Times for Priority Values

Priority Values	Kernel Compilation Time
10	1302.63
80	1152.87
80	1069.91
15	735.228
40	516.972
20	513.594
19	724.74
60	500.85
99	500.862
-19	730.842
35	727.008
15	726.378
-15	735.81
85	709.008
1	786.378
0	715.674
20	712.608
50	772.614