OS Assignment 2 - Report for Q1a

By: Arnav Goel (2021519), Section B, Branch: CSAI

1) Explanation of Logic:

In this question, we were asked to make three functions countA(), countB(), countC(). All the three functions do the same function i.e. count from 1 to 2^32. We call the three functions from three different threads, thrA(), thrB(), thrC() which we create using the POSIX Standard API call - **pthread_create()**. For scheduling of the thread with policy SCHED_FIFO and SCHED_RR, use the system call provided by the POSIX API Library called **pthread_set_schedparam().** This takes the Thread ID, Policy and a pointer to a struct which stores the priority as its arguments. For the scheduling of the thread following the SCHED_OTHER policy, we use the **nice()** system call for setting its nice value.

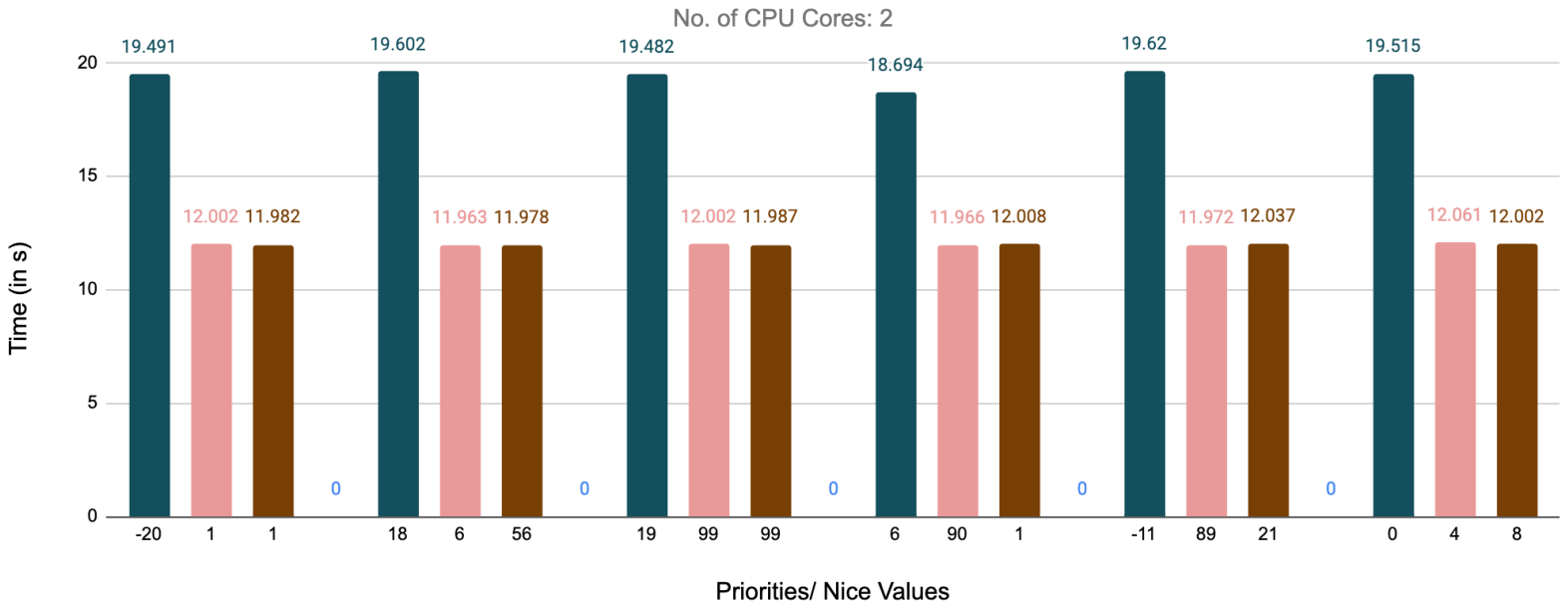
We do this scheduling in every function respectively before starting the counting. We use the **clock_gettime()** system call before the scheduling is done to note the start time for each thread and then use it at the end of the counting to note the end time of counting for each thread. After a thread finishes its execution, we call the **pthread_join()** system call which waits for each thread to end its execution. We then print out the time taken for each thread with its corresponding policy.

2) Outcomes and Measurements:

S No.	Nice Value	Priority_FIFO	Priority_RR	Time - (SCHED_OTHER)	Time - (SCHED_FIFO)	Time - (SCHED_RR)
1	-20	1	1	19.490389	12.002137	11.981757
2	18	6	56	19.601917	11.963405	11.978455
3	19	99	99	19.482482	12.002279	11.986687
4	6	90	1	18.693597	11.965901	12.008329
5	-11	89	21	19.620215	11.972198	12.037309
6	0	4	8	19.515184	12.058939	12.001161

The table above shows the various priority values/nice values for the three types of Scheduling Policies i.e. SCHED_OTHER, SCHED_FIFO, SCHED_RR and their corresponding runtimes. Sched_other as described in the Linux Manual Page is starved by the system scheduler and hence irrespective of high or low nice values, always gives a higher runtime than SCHED_FIFO and SCHED_RR. SCHED_FIFO and SCHED_RR give almost equal times because of us using 2 cores for processing our threads and if SCHED_FIFO is given a lower priority due to the nature of its scheduling policy it always takes more time to complete than SCHED_RR.

The following bar graph shows the plot for the readings of runtimes and priority/nice values:



Green: SCHED_OTHER
Pink: SCHED_FIFO
Brown: SCHED_RR