OS Assignment 2 - Report for Q1b

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

1) Explanation of Logic:

In this question, we were asked to create 3 child processes from a parent process. Each child process would do the **exec call** to a unique bash script which will compile a Linux-5.19.8 kernel. The logic I am employing here is that I am checking using an if/else statement if my current process is parent or not and then only forking it to create a child process. Before using the **fork()** call to fork my program, I measure the start time of that particular child process using the **clock_gettime()** system call. I am then using a **waitpid()** system call for each of the three child processes by inputting their respective process IDs using the **getpid()** call to wait for each process and then subsequently note their time of finishing using **clock_gettime()** again. After this, I am printing out the time taken by each of the processes.

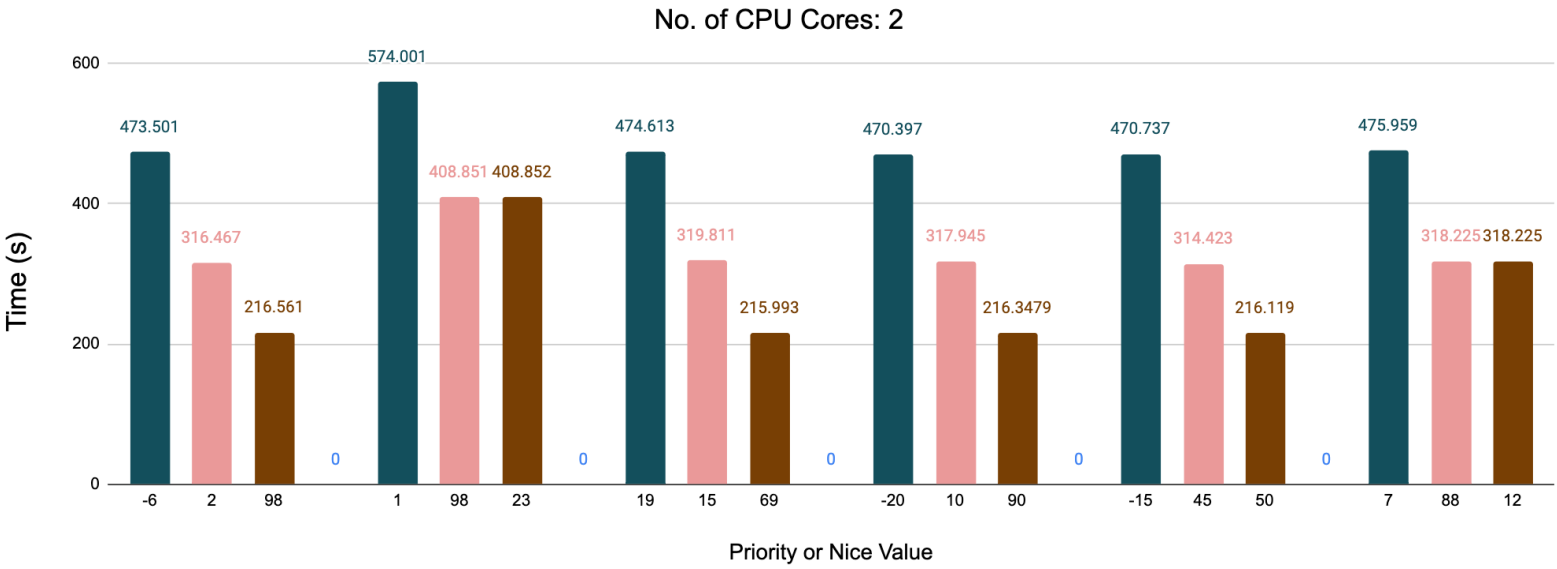
For scheduling my processes, I am using the system call **sched_setscheduler()** for SCHED_FIFO and SCHED_RR policy whereas I am setting the nice value for the process following the SCHED_OTHER policy using a call to the **nice()** system call. The exec call exits the child program which ensures that my child program is not being forked. I used **fflush(stdout)** before this execl call is executed to print anything on stdout before the child process is killed by the new execl process as it does not inherit any strings on the stdout and thus stdout has to be flushed to return print statements.

2) Outcomes and Measurements:

S No.	Nice Value	Priority_FIFO	Priority_RR	Time - (SCHED_OTHER)	Time - (SCHED_FIFO)	Time - (SCHED_RR)
1	-6	2	98	473.500737	316.467187	216.56142
2	1	98	23	574.000239	408.850752	408.850683
3	19	15	69	474.613198	319.811498	215.992921
4	-20	10	90	470.396634	317.945268	216.347868
5	-15	45	50	470.736695	314.422523	216.119631
6	7	88	12	475.959855	318.225256	318.225216

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. If SCHED_FIFO has high priority value, it and SCHED_RR still have almost the same time taken.

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