# Readme.1_1

- In this exercise we are asked to use Linux's scheduling policies **SCHED_OTHER,SCHED_RR,SCHED_FIFO**. We have created three threads **Thread_A,Thread_B,Thread_C** each of which does counting calling their respective functions count_A Count_B Count_C from 1-2^32.
- 
- In the main program , we have created three threads using **p_thread_create()** for each thread which calls their respective launch functions thread_A_launch , thread__B_launch and thread_C_launch and joining these p_threads in the main program itself.
- 
- In **thread_launch_A:** we have first set the scheduling policy to **SCHED_OTHER** and then used the function s clock_gettime() with **timespec start** as one parameter and setting the CLOCK_REALTIME to start the time for function count_A and then stopped the time using the same function clock_gettime with **timespec stop** as one of the parameter
- .
- Similarly for t**hread_launch_B** with scheduling policy to **SCHED_RR** and **thread_launch_C** with scheduling policy to **SCHED_FIFO** with increasing the priority +2 every time starting from an integer value of 2 to 20 to get a total of 10 readings in total . We have saved all the time values(**accum**) in file Assignment_2_values.txt and tried to plot a histogram graph of it.
- It is observed in the graph that **SCHED_FIFO** has always taken **minimum time** to run the count function and **SCHED _OTHER** has taken **maximum** time among the three policies to run the count function in all the 10 readings. Also ,SCHED_RR takes less time than SCHED_OTHER and more time than SCHED_FIFO.