Operating Systems (CS F372)
Tutorial Sheet 10
Scheduling

In this tutorial sheet, we will further explore the concept of process scheduling, emphasizing thread scheduling.

Some useful commands:

`ps -eLo pid,ppid,class,pri` - This command will provide a list of all processes running on the system, displaying their Process ID (pid), Parent Process ID (ppid), scheduling class (class), and priority (pri).

`top -H -p <pid>` - This command provides insights into resource usage, thread activity, and overall system performance related to the specified process. Alternatively, you can use the `htop` CLI tool which offers a GUI-esque experience. You can install it using `sudo apt install htop`

Problem 1:

Write a C program that explores and assesses the complexities of thread scheduling. Your program should create two threads, each with its priorities and policies. While Thread 2 handles more complex computational tasks, Thread 1 is meant to carry out a simple counting task. Once the threads are created, measure and display each thread's execution time precisely.

Note: Use `sudo` command to run your program when working with Real-time Scheduling policies (FIFO, RR)

Problem 2:

Implement a C program for Priority Scheduling with preemptive behavior. The program should take user input for each process's arrival time, burst time, and priority. Simulate the scheduling of processes and calculate the waiting time and turnaround time for each process. Ensure that the processes are scheduled based on their priority, and if two processes have the same priority, the one with the lower arrival time is given preference.

Take Home Exercise:

Implement a C program for Priority Scheduling without preemption. The program should take user input for the number of processes, their burst time, priority, and arrival time. Simulate the scheduling of processes based on priority without allowing preemption. Calculate the waiting time and turnaround time for each process and display the results.