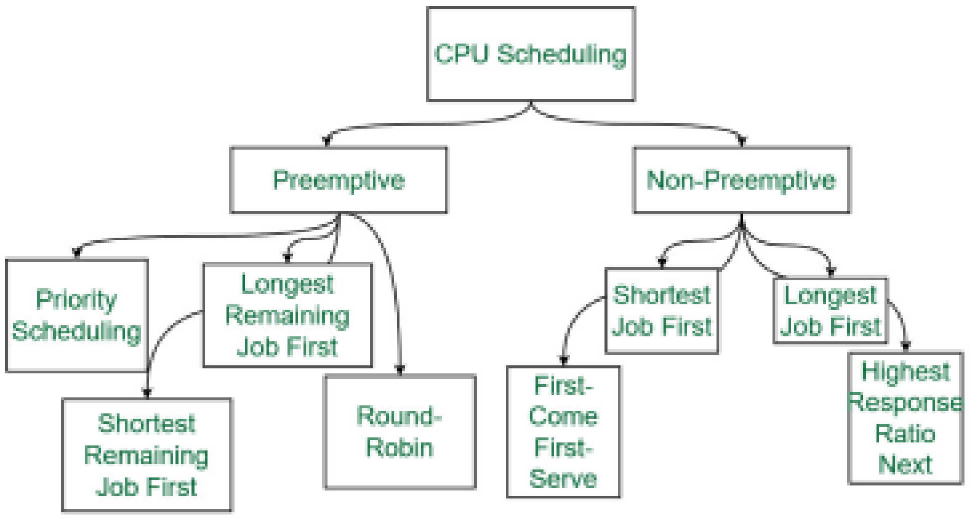
Operating Systems (CS F372)
Tutorial Sheet 9 (Scheduling)

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



CPU Scheduling is a process that allows one process to use the CPU while another process is delayed (in standby) due to unavailability of any resources such as I/O etc, thus making full use of the CPU. The purpose of CPU Scheduling is to make the system more efficient, faster, and fairer. Whenever the CPU becomes idle, the operating system must select one of the processes in the line ready for launch. Some of the scheduling policies offered by Linux are:

- SCHED_FIFO : Real-time scheduling policy that uses the First In-First Out scheduling algorithm.
- SCHED_RR: Real-time scheduling policy that uses the Round Robin scheduling algorithm.
- SCHED_OTHER: Default Linux scheduling policy(CFS - Completely Fair Scheduling) used for regular tasks.
- SCHED_BATCH : Scheduling policy designed for CPU-intensive tasks. It does not preempt nearly as often as regular tasks, allowing tasks to run longer but at the cost of interactivity. This is well suited for batch jobs.
- SCHED_IDLE: Scheduling policy intended for very low prioritized tasks.

Some important commands:

chrt -m - Lists the maximum and minimum valid priorities for all scheduling policies chrt -p pid - Shows what the current scheduling policy and priority are for the process with process id = pid
chrt [policy flag] -p priority pid - Changes the scheduling policy of the process with process id = pid, with a priority value = priority, to the scheduling policy represented by policy flag(-o for SCHED_OTHER, -f for SCHED_FIFO, -r for SCHED_RR, -b for SCHED_BATCH, -i for

SCHED_IDLE).

Eg. chrt -f -p 34 22885 - Changes the scheduling policy of the process with process id 22885 to SCHED_FIFO, with a priority value of 34. Refer this for more details on the chrt command and for the valid range of priority. You might also need to use sudo if you are using chrt with SCHED_FIFO and SCHED_RR.

Now we will look into the concept of nice values. When you use the chrt -m command to find the maximum and minimum values of priority values for SCHED_OTHER, SCHED_BATCH and SCHED_IDLE, you will notice that they are 0 and 0. These are all non-realtime scheduling policies and it means all of these processes have the same 'priority' and their only difference is in their nice value. nice value ranges from -20 to 19. Bigger is the nice value of a process, lower is its priority and the 'nicer' it is to other processes. Refer this for more details.

nice -n <nice_value> <command> - Replace <nice_value> with the priority level and <command> with the actual command or process you want to execute with that priority renice -n <nice_value> -p <process_id> - Replace <nice_value> with the new priority level you want to assign and <process_id> with the ID of the running process you wish to modify.

Eg. renice -n 43 -p 59036 - Changes the nice value of process with process id 59036 to 43 Note: Use ps ax -o pid,ni to find the nice values of available processes.

Problem 1:
Write a POSIX-compliant C program that takes in three command line arguments: the scheduling policy ('f' for FIFO, 'r' for RR, etc.), priority, and PID of the process whose scheduling policy and priority must be changed. The program should also print the valid priority range of the policy provided, and then change the scheduling policy and priority of the process provided.

Note: Use `sudo` to run your program if you are working with real-time scheduling policies (FIFO or RR). In this case, it may not be possible to run for the students.
Note: You can use **`sleep 10000&`** to create a process and get its PID.
Note: Use **`chrt -p <PID>`** to check if the program works correctly.

Problem 2:

Write a POSIX-compliant C program that calculates the timeslice allocated to an RR process. What happens when you run the program for a FIFO process?

Note: You can use **`sudo chrt --[rr|fifo] 1 ./[exec_name]`** to run the program with the required scheduling.

Problem 3:

Create a POSIX-compliant C program that prints the scheduling policy and priority of the thread and then modifies these values and prints the new policy and priority.