# Project: Process Scheduling

This project comes in two parts: **Simple Research**, and **Implementation**.

For the **Simple Research** part:
Conduct simple research to answer these 3 questions about the current status and future of operating systems. Have a minimum of 2 pages but there is no max limit. Use Arial font size 11, and single spacing. Please explain and elaborate. You must also have cited sources, and those sources must be cited. Please remember that chatGPT is not a valid source.
Here are the (guide) questions for the simple research:
1. What scheduling algorithm(s) are being used by operating systems today?
2. What active research areas are there today, in the context of improving CPU scheduling?
3. What active research areas are there today, in the context of improving operating systems
in general?

For the **Implementation** part:

Every scheduling algorithm is different and may favor (boost CPU performance
in the presence of) certain types of processes. Each algorithm will be worth 10 points (5 algorithms = HPS: 50 points).

To do this group project, (by your lab groups), you need to learn about scheduling algorithms. CPU scheduling involves looking at the ready queue and picking one (or more, if more than one processor is free) to be run. You are given an input file (all numbers are integers, no fractions):

- First line: A single number indicating the number of test cases.
- Second line: Start of first test case.

For each test case, at least 2 lines:

- First line: A number X greater than zero indicating number of processes, followed by a string: FCFS, SJF, SRTF, P, or RR. In the case of RR, another number Q greater than zero will follow.
- Next X lines: The X processes. Note that the first process' index is 1. Last process' index is therefore X and not X-1.

For each process, 1 line, 3 numbers:

- First number: Arrival time in ns, always zero or positive (test case start time assumed to be 0ns but processes may not have arrived yet).
- Second number: Burst time in ns, always greater than zero.
- Third number: Priority, range is -20 to +20, inclusive.

The string after X corresponds to the abbreviation of the algorithm to use:

- FCFS = First Come First Served
- SJF = Shortest Job First (non-preemptive)
- SRTF = Shortest Remaining Time First (SJF preemptive)
- P = Priority (preemptive),
- RR Q = Round-Robin (the number after the RR represents the time quantum).

Specifications for the Project:

1. Assume a uniprocessor and zero overhead (context-switching is instant).
2. Create your own test case
3. For each test case, output a text version of the resulting Gantt chart:
    a. First line: Test case number (start with 1).
    b. Second line: First “block”, corresponds to the first process to be run.
4. For each block:
    a. First number: Time elapsed so far in ns.
    b. Second number: Process index.
    c. Third number: CPU time used in ns. Add a capital X immediately at the end of this number if the process has completed at the end of this block.

Sample 1:

- Input (via standard input)

```
2
4 SRTF
0 50 2
40 2 3
20 3 1
30 55 1
2 FCFS
100 10 1
10 70 1
```

- Output (standard output)

```
1
0 1 20
20 3 3X
23 1 17
40 2 2X
42 1 13X
55 4 55X
2
10 2 70X
100 1 10X
```

Sample 2:
For round-robin, the behavior must mimic FCFS except that preempted processes are moved to the tail end of the queue and, if not currently running,
must give way to new arrivals.

- Input

```
1
4 RR 25
0 30 3
25 45 4
75 10 1
55 15 5
```

- Output

```
1
0 1 25
25 2 25
50 1 5X
55 4 15X
70 2 20X
90 3 10X
```

**Scheduling Criteria Output**

Compute the 5 scheduling criteria/measure for each case / scheduling. Place at the end of output.

Sample:

▶ Output

```
1

0 1 25
25 2 25
50 1 5X
55 4 15X
70 2 20X
90 3 10X
CPU Utilization: (%)

Throughput: (# of processes)
completed per unit of time
Waiting time: (time) cumulative

Turnaround time: (time)

Response time: (time)
```

Remember what we want to maximize:

1. CPU utilization: Must keep CPU busy. To calculate this, you must determine the amount of time the CPU was not doing anything.
    a. Example: (100ns of total time - 0ns of idle) / 100ns of total time = 100%
    b. Ideally, 100% might indicate an overloaded system but for this exercise, 100% is ok.
2. Throughput: Number of processes completed per unit time.
    a. Example: 4 processes / 100ns of total time = 0.04 processes/second
    b. Pipelining and multiple processors can only get you so far.
3. Waiting time: Total or sum of times waiting in the ready queue. For this project, display the 1) waiting time of each process; and 2) the average waiting time.

a. Example:
    i. Waiting Time:
        1. Process 1: 10ns
        2. Process 2: 20ns
    ii. Average waiting time = 15ns
b. Has NOTHING to do with waiting for I/O.

4. Turnaround time: Time it takes to execute the process, from submission (entry into the system) to completion (termination). For this project, output the turnaround time for each process.
    a. Example for FCFS:
        i. Turnaround Time:
            1. Process 1: 50ns
            2. Process 2: 30ns
        ii. Average turnaround time = 40ns
    b. Not the best criterion for an interactive system. Why?
5. Response time: Time it takes to start responding, from submission to first response.
    a. Some output can be produced early, after all.
    b. But this does NOT include the time it takes to actually output the ENTIRE response.
    c. Generally limited by the speed of the output device.

**Requirements**

- 3-7 mins demo video to present your code and output
- CANVAS submission of:
    - Simple Research PDF File
    - C++ code
    - Test file and output file
    - Readme File including video Via Google Drive Link
    - CoA

● Include the certificate of authorship. If you have questions, you may email your respective professor. FAQ may be made available if there are a lot of common questions. Good luck!