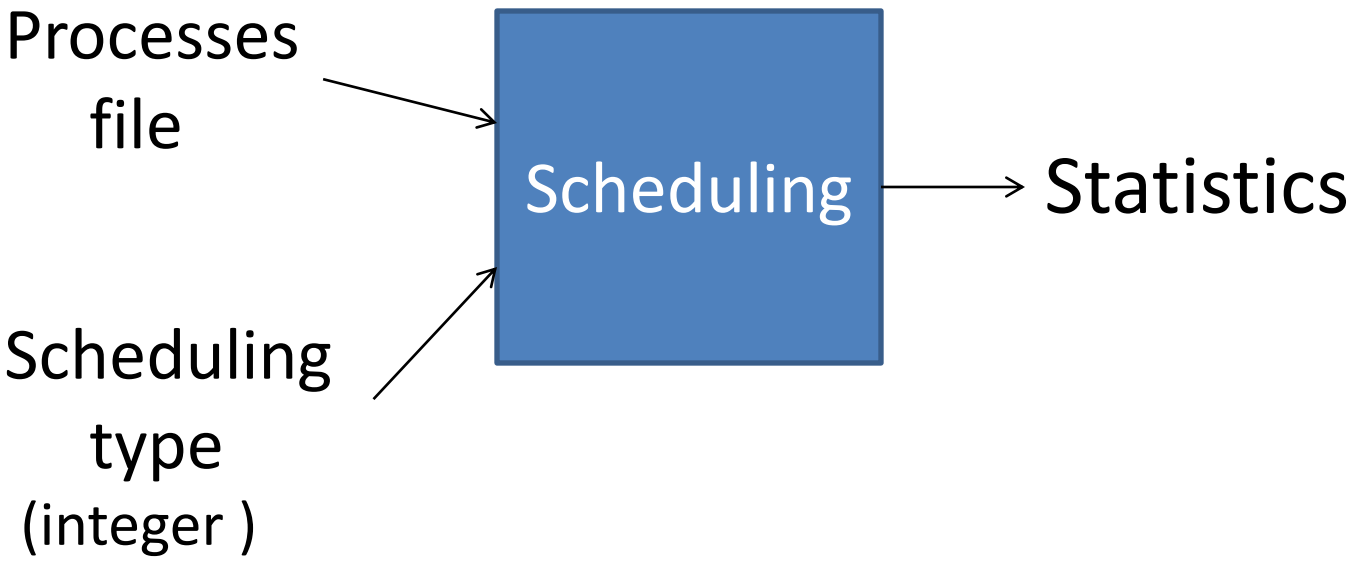
OS Scheduling

What Will We Do?

- In this project we will test several scheduling algorithm



Processes File

- Each process will be presented by 4 numbers:

A B C D:

  - A: process ID
  - B: CPU time
  - C: I/O time
  - Arrival time

Note: If more than one processes arrive at the same time, give preference to the one with lower ID.

How time is distributed for a process

Note: We will use integers, no floating points.
In case (0.5 * CPU Time) is float, round to following cycle (e.g. 5.5 -> 6).

Scheduling Algorithms

- 0: First-Come-First-Served (nonpreemptive)
  - Queue of ready processes
  - Newly income processes are added to the end of the queue
  - When a process is blocked, due to I/O, and then becomes ready, it is added to the end of the queue.
  - If two processes happen to be ready at the same time, give preference to the one with lower ID.

Scheduling Algorithms

- 1: Round-Robin with quantum 2
  - Another process scheduled if one of the following occurs:
    - Current running process terminates
    - Current running process is blocked on I/O
    - Current running process ran for 2 cycles
  - You can think of RR as a queue of ready processes. When a process goes from running to ready, it moves to the back of the queue.
  - If two processes become Ready at the same time, give preference to the one with smaller ID

Scheduling Algorithms

- 2: Shortest remaining job first (preemptive)
  - After each clock cycle calculate the remaining CPU time for all ready/running processes and run the one with shortest reaming time
  - If several processes have the same remaining CPU time, give preference to the process with lower ID.

Output

- You output a file with name: inputfilename-s.txt
  - inputfilename if the name of the input file without the extension
  - s is the type of scheduling:0, 1, or 2
  - Example: if input file is inp1.txt, your output file for FCFS shall be: inp1-0.txt
- Your output file has two parts
  - Timing snapshot (starting from cycle 0)
  - Statistics

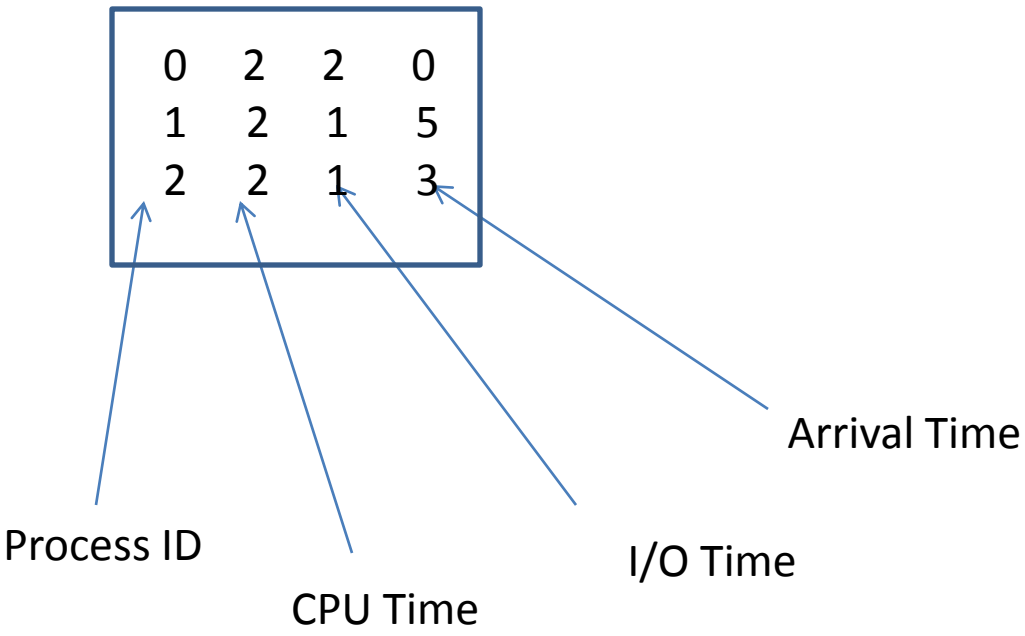
Output

- Timing snapshot: at every line show:
  - Cycle time
  - State of each process (**running**, **ready**, or **blocked**)
    - example: 1:blocked (i.e. process 1 is in blocked state)
    - Print processes ordered by their process ID
  - Be careful: do not show processes that have not yet arrived, or those that have terminated.
- Statistics:
  - Finishing time (i.e. last cycle)
  - CPU utilization (#cycles CPU was doing work / total number of cycles)
  - For each process:
    - Turnaround time (i.e. cycle this process finished – cycle it started + 1)

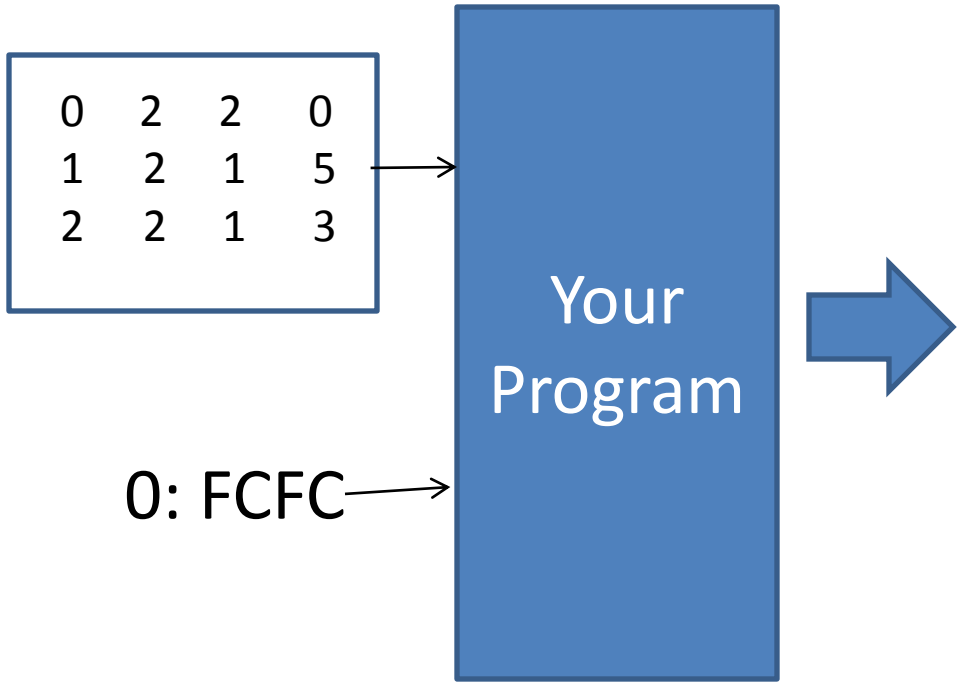
Example

0	2	2	0
1	2	1	5
2	2	1	3

Example



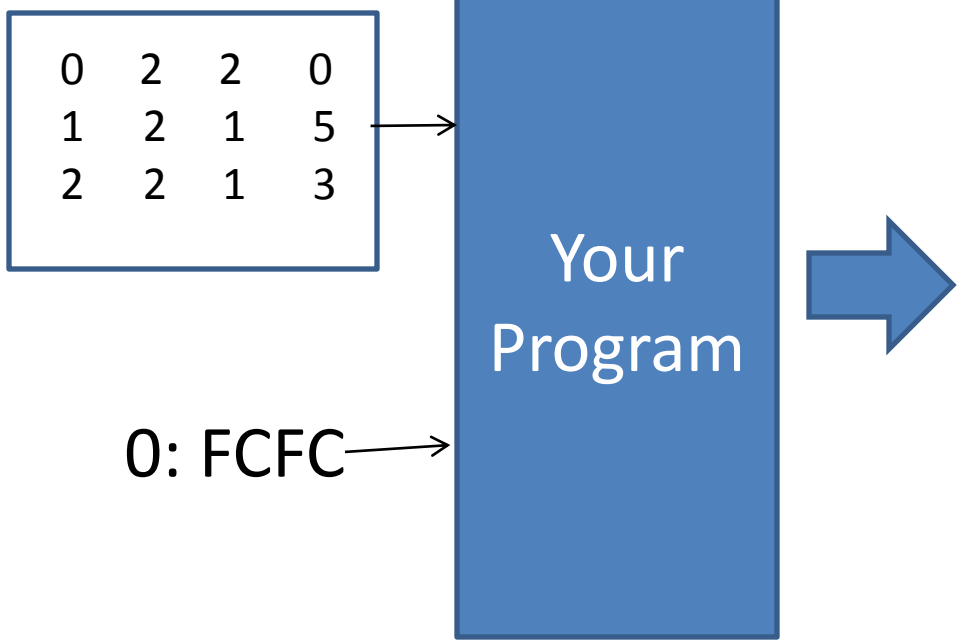
Example



0 0:running
1 0:blocked
2 0:blocked
3 0:running 2:ready
4 2:running
5 1:running 2:blocked
6 1:blocked 2:running
7 1:running

Finishing time: 7
CPU utilization: 0.75
Turnaround process 0: 4
Turnaround process 1: 3
Turnaround process 2: 4

Example



Clock cycle

State of each process

0 0:running
1 0:blocked
2 0:blocked
3 0:running 2:ready
4 2:running
5 1:running 2:blocked
6 1:blocked 2:running
7 1:running

Finishing time: 7
CPU utilization: 0.75
Turnaround process 0: 4
Turnaround process 1: 3
Turnaround process 2: 4

What To Submit

- Readme file, contains the following info:
  - How to compile and run your code on CIMS machines
  - e.g. gcc –o name file1.c file2.c
  - e.g. mycode 0 inputfile
- Source code

How to Submit

- Email your corresponding grader
- subject line: OS Scheduler Submission
- Make a zip file with your submission with the following naming convention:
  - lastname-firstname.zip
- Attach the file to the email and send it.
- Deadline: Due date at 11:59pm

Avoid The Following Mistakes (Penalty applied for each)

- TA must not change your source code (-5)
- Code does not run on CIMS machines (-5)
- You used a tool to compile your code not present on CIMS machines (-5)
- Late submission (-1 for each day)
- Output of different format (-2)
- The work is not your own (zero!)