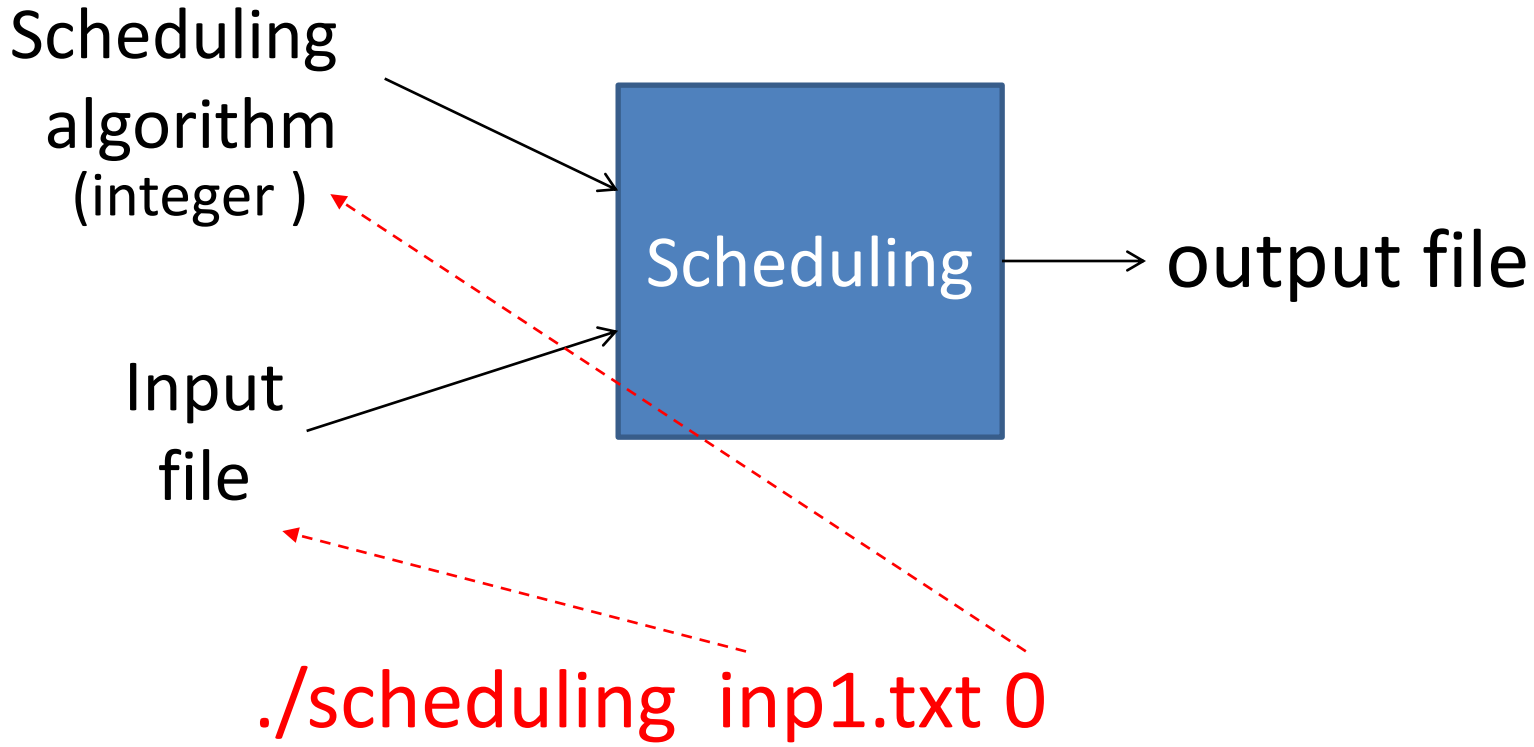
# Programming Assignment
# OS Scheduling

# What Will We Do?

- In this project we will test several scheduling algorithms



# Your Source Code

- scheduling.c
- compile with:

gcc –Wall –o scheduling –std=c99 scheduling.c

# Input File

- The first line in the file is the total number of processes.
- Each process will be represented by 4 integers:

A B C D:

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

**Note:** If more than one process arrives at the same time, give preference to the one with lower ID.


**How time is distributed for a process**

**Note**: We will use integers, not floating point. In case (0.5 * CPU Time) is float, round to following cycle (e.g. if cpu time is 7 then 7/2 = 3.5 -> 4).

All times are in cycles

# Scheduling Algorithms

- 0: First-Come-First-Served (nonpreemptive)
  - Queue of ready processes
  - Newly arriving 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)
  - At each cycle, you 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.

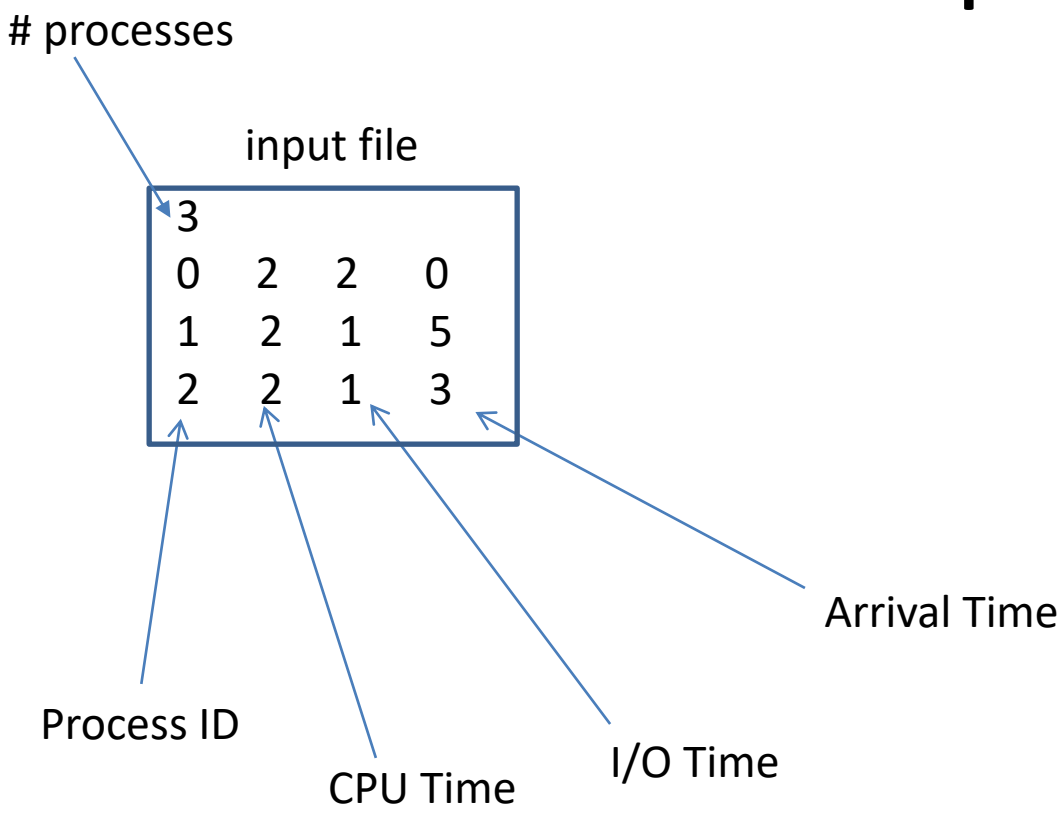
# Output

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

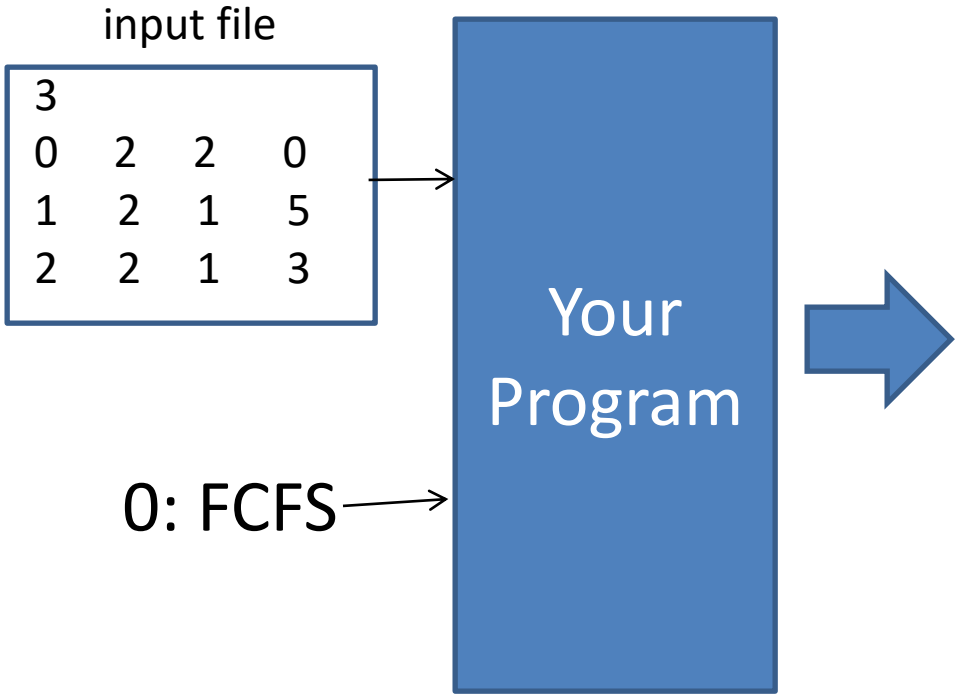
# Output

- Timing snapshot: at every line show:
  - Cycle number
  - State of each process (**running**, **ready**, or **blocked**)
    - example: **1:blocked** (i.e. process 1 is in blocked state)
    - Print processes, in the same line, 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)
    - When there is a cycle where none of the processes is running, then the CPU is considered idle.
  - For each process:
    - Turnaround time (i.e. cycle this process finished – cycle it started + 1)

# Example


# processes
input file
3
0 2 2 0
1 2 1 5
2 2 1 3
Process ID
CPU Time
I/O Time
Arrival Time

# 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
(empty line)
Finishing time: 7
CPU utilization: 0.75
Turnaround process 0: 4
Turnaround process 1: 3
Turnaround process 2: 4

# What To Submit

Your source code: single file with the name

scheduling.c

# Avoid The Following Mistakes (Penalty applied for each)

- Code does not run on CIMS machines (-5)
- Late submission (-20% for each day)
- Output with different format (-5)
- The work is not your own (zero!)

# Excuses not Accepted

- I submitted the wrong file.
- I submitted one minute after the deadline.
  - We highly suggest that you upload a version, even if not yet complete, each time you implement something and do not wait till you finish the whole program.

# One last thing

- To help you start, we are proving you with a C file (skeleton-lab1.c) that:
  - Reads arguments from command line
  - Checks that the arguments are correct
  - Forms the name of the output file
- You can use this file, part of it, or none at all. It is up to you as long as your submitted program works correctly.

# To test your code

- You are provided with two zip files 2processes.zip and 3processes.zip
  - Each one contains an input file (2processes.txt and 3processes.txt) and the output for each of the three different scheduling algorithms
- You can also use these example files to formulate other test cases, solve them on a piece of paper first, then check the output.

All the Best!