Gaulthier Gain & Benoît Knott

INFO0940
Operating Systems

Project 1:
CPU Scheduler Simulator

Academic year 2023-2024

Overview

- **Your job** will be to implement a CPU scheduler simulator.

- What is a **CPU scheduler**?

  It is the part of the operating system that decides which process to execute next.

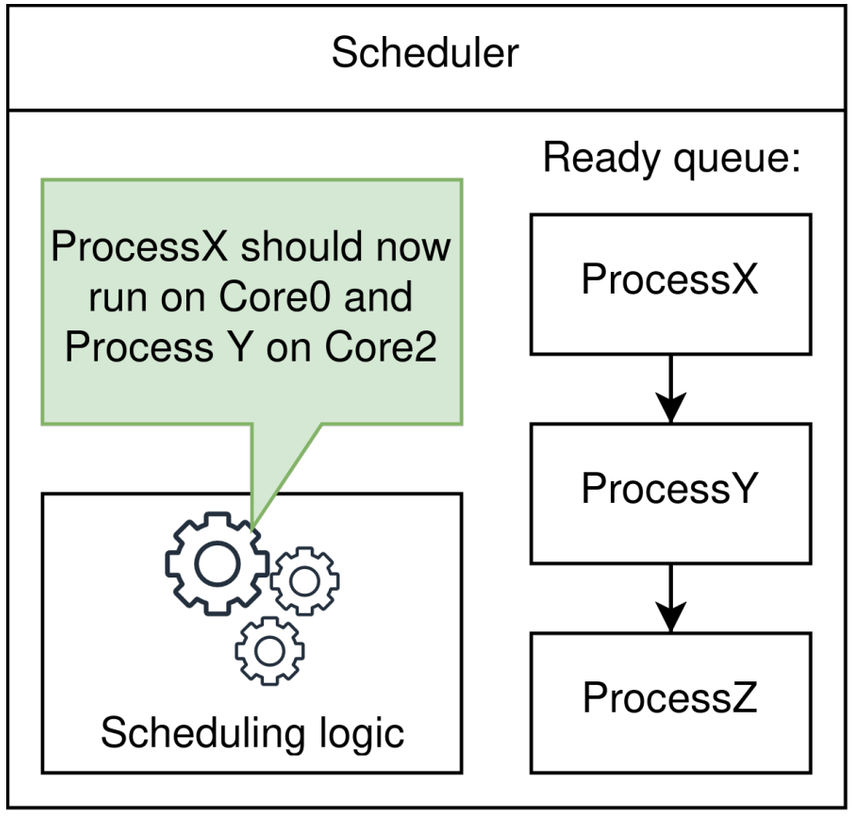
- Programming a real CPU scheduler is a complex task.

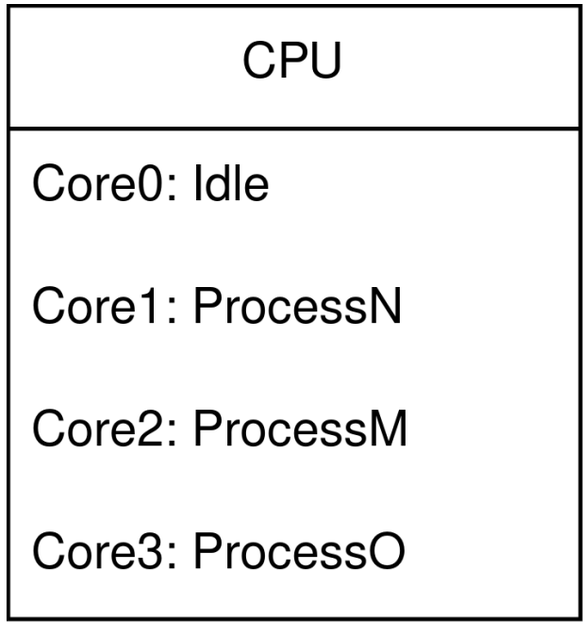
  → Instead, you are going to make a **CPU scheduler simulator**.

CPU Scheduler

A **CPU scheduler**:

- maintains a list of ready processes.
- can use different scheduling algorithms to make its decisions.

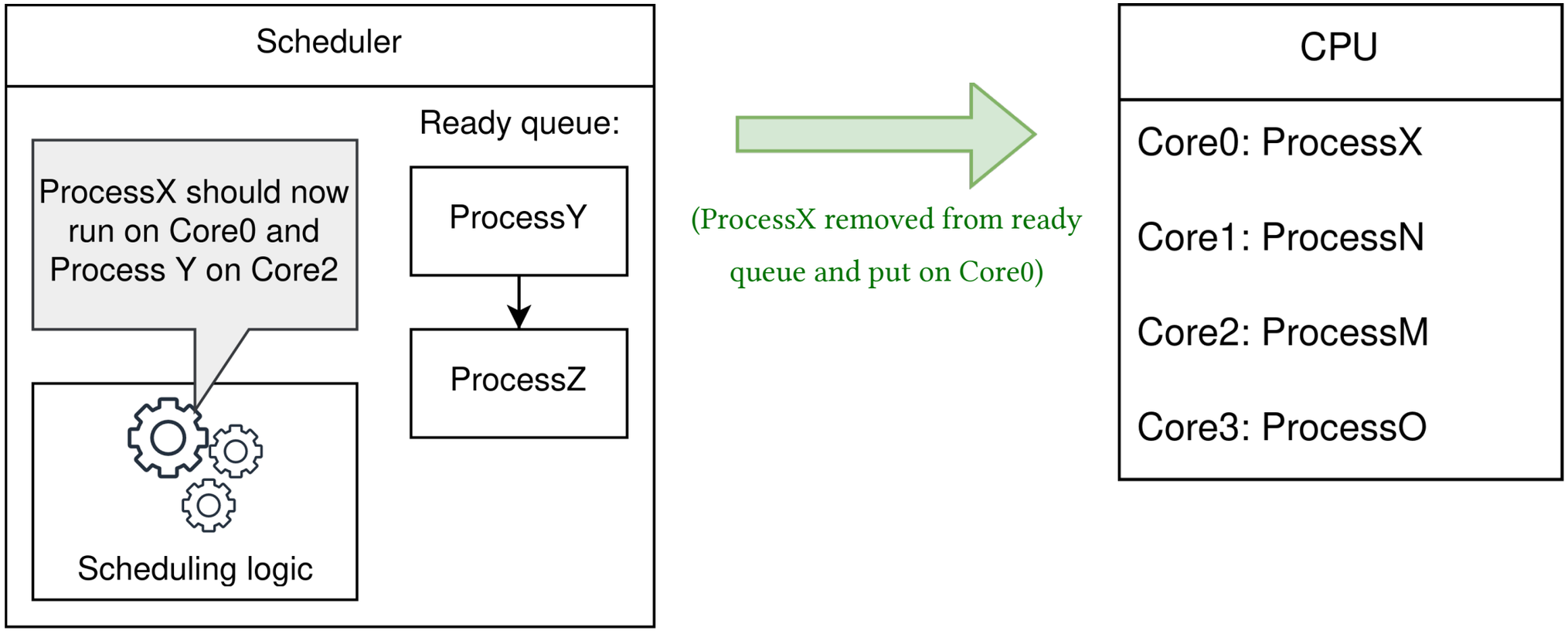




CPU Scheduler

A **CPU scheduler**:

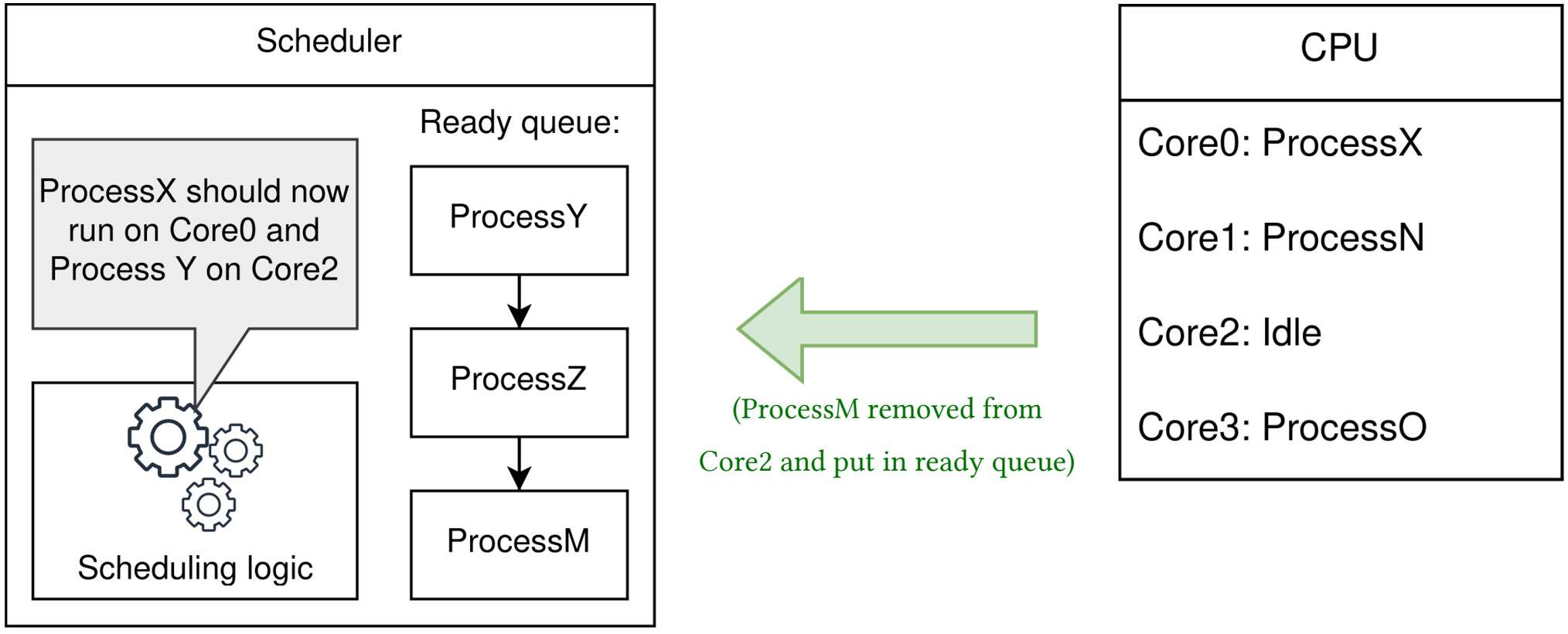
- maintains a list of ready processes.
- can use different scheduling algorithms to make its decisions.



CPU Scheduler

A **CPU scheduler**:

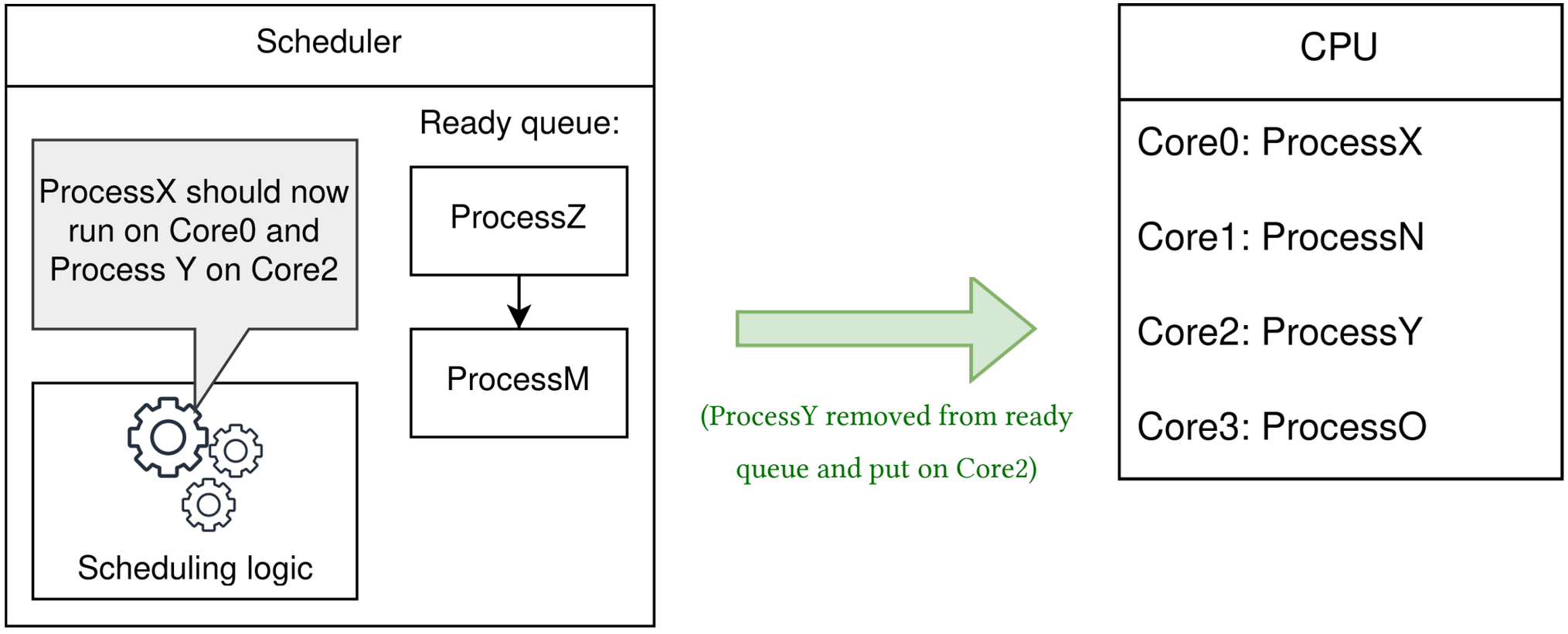
- maintains a list of ready processes.
- can use different scheduling algorithms to make its decisions.



CPU Scheduler

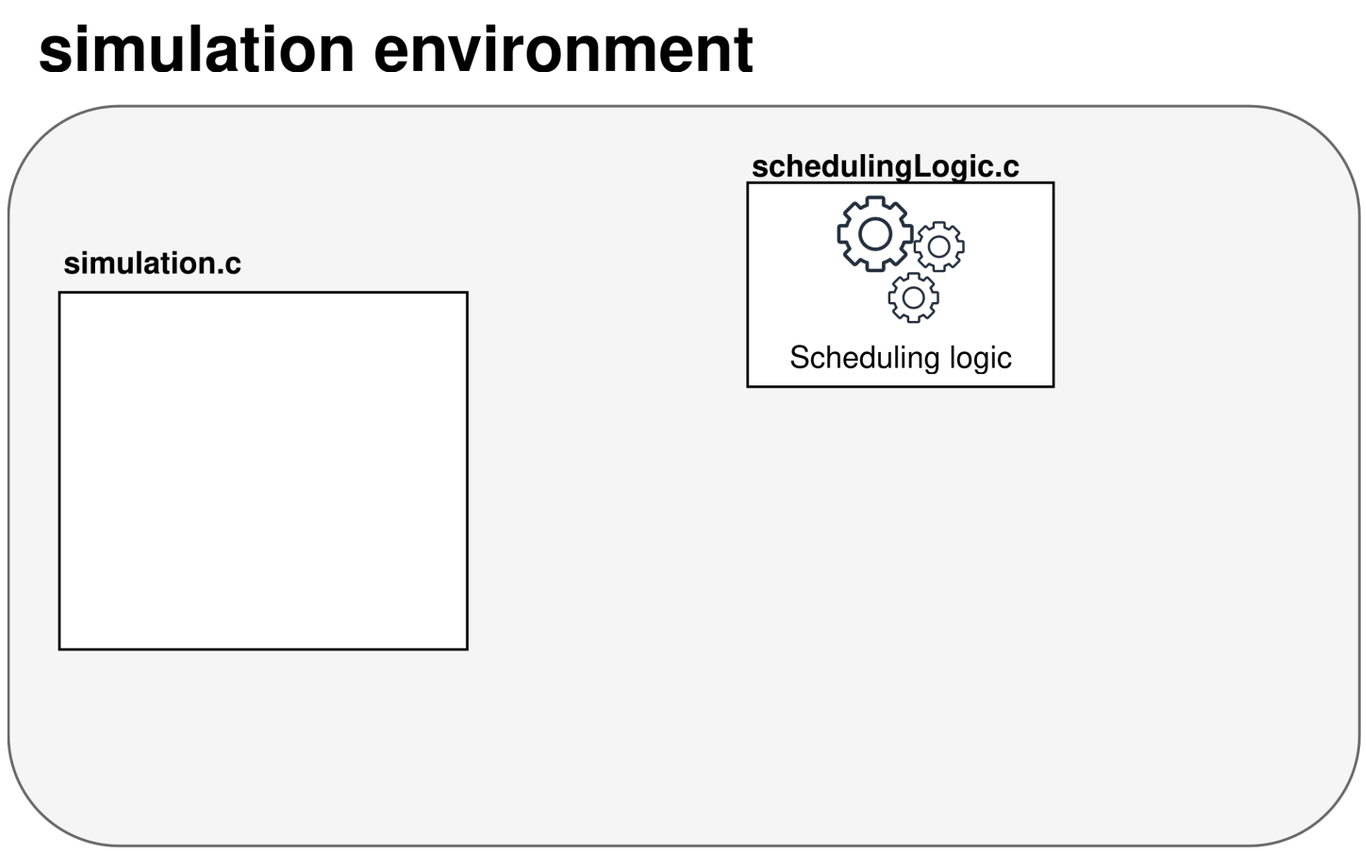
A **CPU scheduler**:

- maintains a list of ready processes.
- can use different scheduling algorithms to make its decisions.



The Simulator

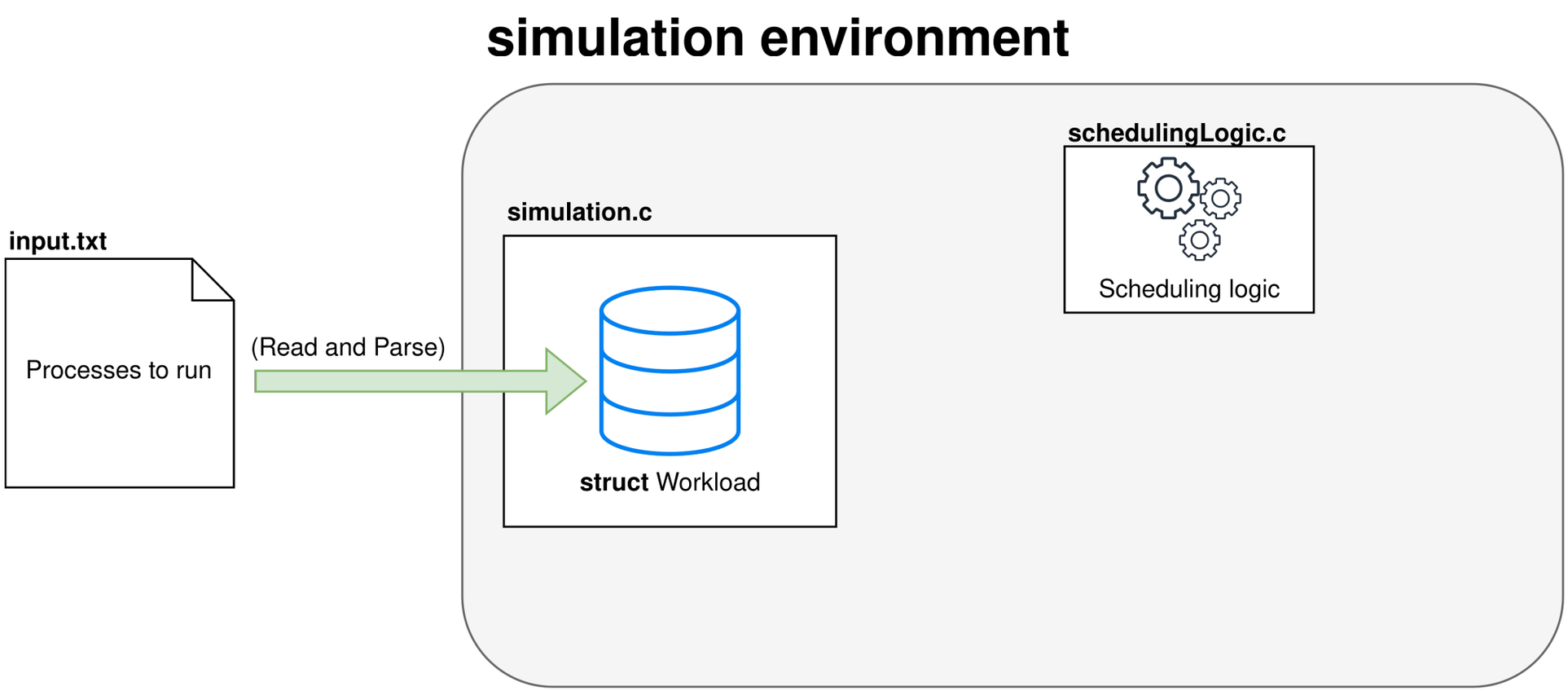
The scheduling logic you will implement will be part of a simulator.



The Simulator

The scheduling logic you will implement will be part of a simulator.

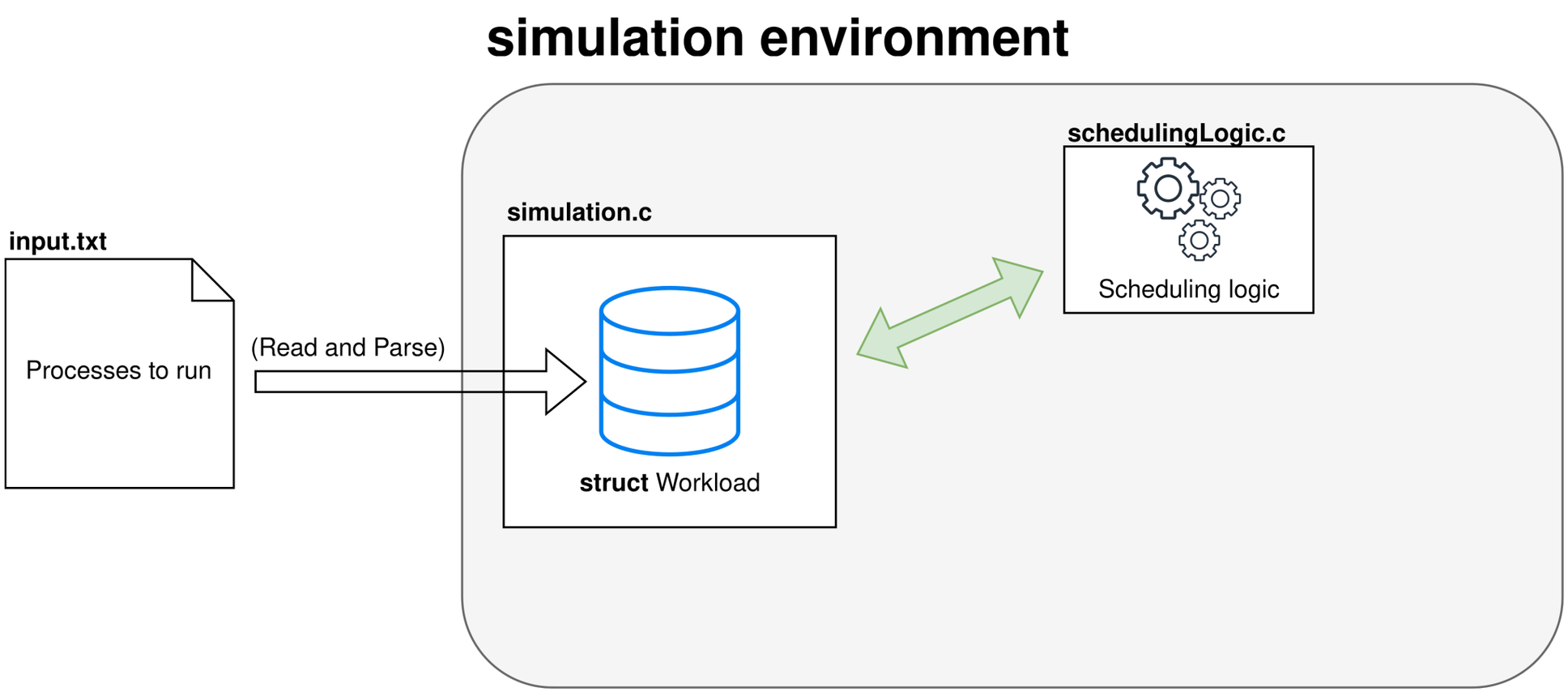
- The simulator will take processes information as input.



The Simulator

The scheduling logic you will implement will be part of a simulator.

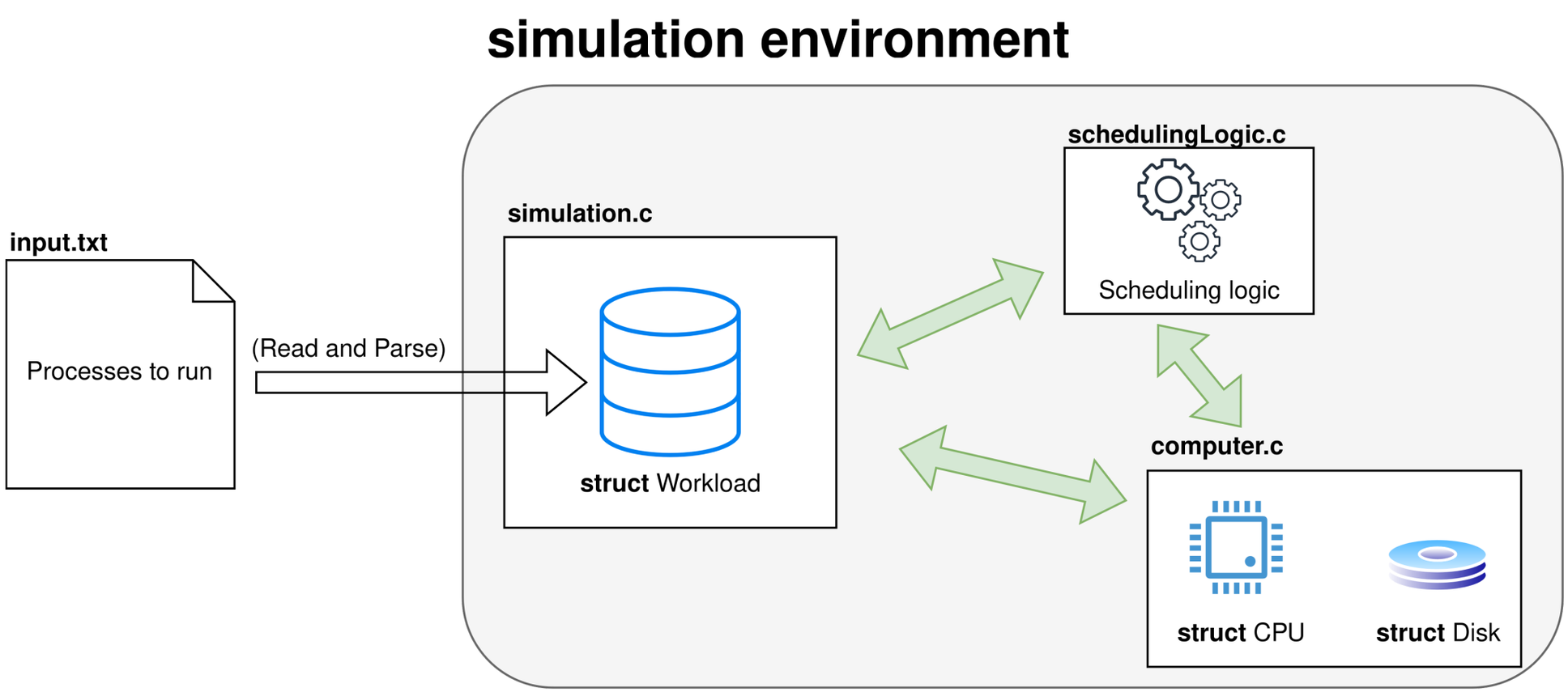
- The simulator will take processes information as input.
  - It will then use the scheduling logic to "execute" these processes.



The Simulator

The scheduling logic you will implement will be part of a simulator.

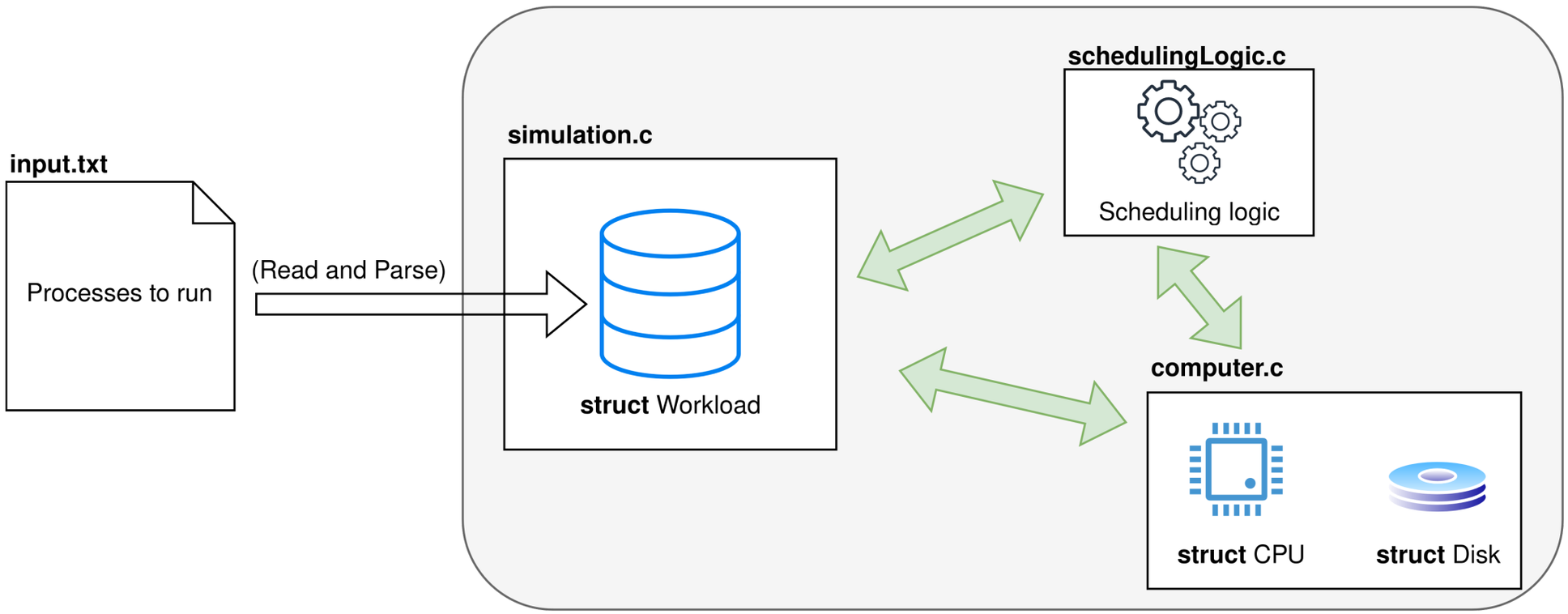
- The simulator will take processes information as input.
  - It will then use the scheduling logic to "execute" these processes.
  - With the use of structures representing the disk and CPU.



The Simulator (2)
Your task

- Boring parts of the simulation are already done for you (parsing arguments, input file, etc.).
- Your task is to implement the main simulation loop and the scheduling logic, which will need to support different scheduling algorithms.

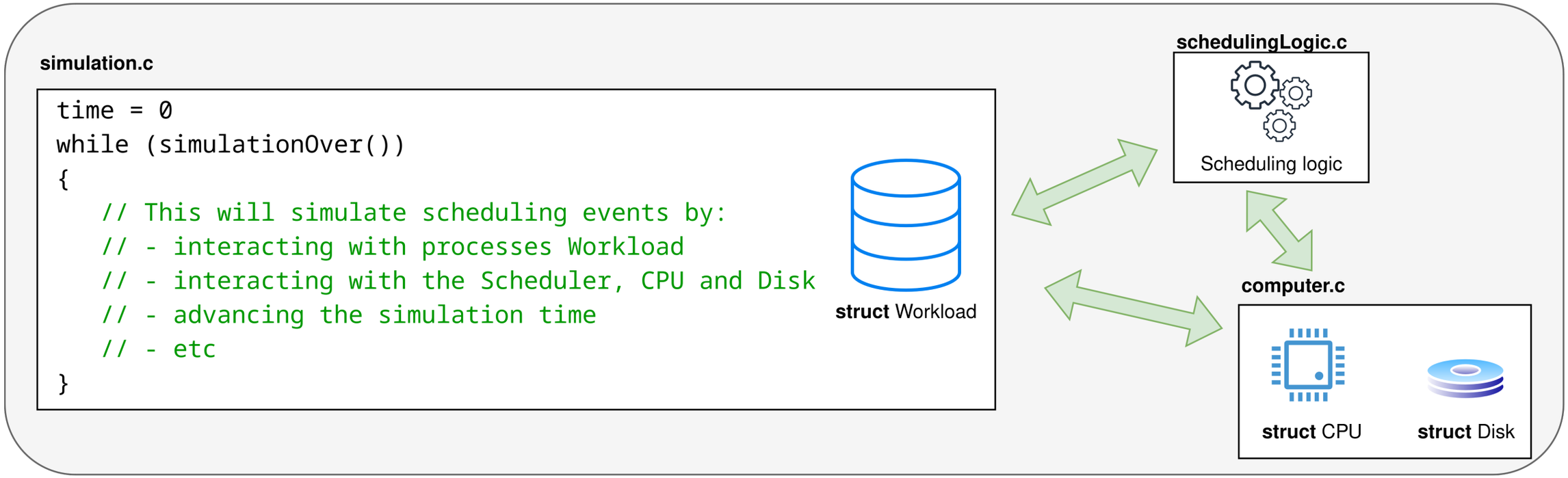
simulation environment



The Main Simulation Loop

The main simulation loop is the core of the simulation, which interacts with the different resources and simulates the progression of time.

simulation environment



Scheduler Features

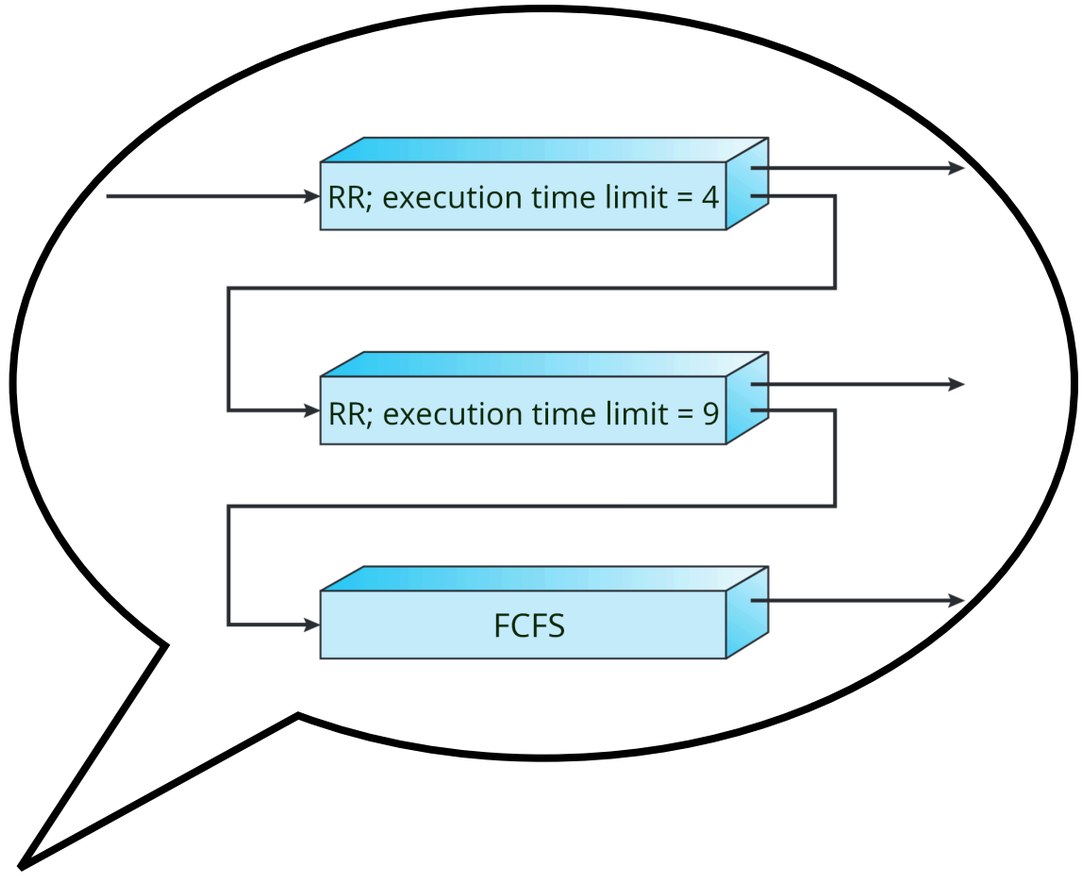
Your scheduler must support:

- Different scheduling algorithms:
    1. FCFS (First-Come-First-Served)
    2. SJF (Shortest-Job-First)
    3. RR (Round-Robin)
    4. PRIORITY
- Multiple cores
- Multilevel feedback queue scheduling

Scheduler Features

Your scheduler must support:

- Different scheduling algorithms:
  1. FCFS (First-Come-First-Served)
  2. SJF (Shortest-Job-First)
  3. RR (Round-Robin)
  4. PRIORITY
- Multiple cores
- Multilevel feedback queue scheduling



Stats and Graphs

- During the simulation, you should compute some statistics of the running processes and update a summary graph.
  → They will be printed at the end of your execution.

- For example, here is the graph of three processes which ran on core 0 one after the other during 29 time slots:

```
Process    0    5   10   15   20   25   30
      1    0000000000
      2    ----------000000000000000
      3    ------------------------00000
X = RUNNING on core X    . = WAITING IO    - = READY
```

- And here are the corresponding stats:

PID	PRIORITY	ARRIVAL	FINISH	TURNAROUND	CPU	WAITING	RESPONSE	C. SWITCHES
1	1	0	10	10	10	0	0.00	0
2	3	0	25	25	15	10	10.00	0
3	2	0	30	30	5	25	25.00	0

Demonstration

Demo

File: **input.txt**

```
# pid, start_time, duration, priority, [list of timestamps and events] (IO, CPU)
1, 0, 10, 1, [(0, CPU)]
2, 0, 15, 3, [(0, CPU)]
3, 0,  5, 2, [(0, CPU)]
```

Simulation takes an input file (`input.txt`)

Demo

```
 code git:(master) ✗ ./bin/cpuScheduler input.txt -c 1 -q 1 --algorithm=FCFS
---------------- Stats ----------------
PID | PRIORITY | ARRIVAL | FINISH | TURNAROUND | CPU | WAITING | RESPONSE | C. SWITCHES
  1 |        1 |       0 |     10 |         10 |  10 |       0 |     0.00 |           0
  2 |        3 |       0 |     25 |         25 |  15 |      10 |    10.00 |           0
  3 |        2 |       0 |     30 |         30 |   5 |      25 |    25.00 |           0
---------------------------------------
---------------- Graph ----------------
Process     0    5   10   15   20   25   30
      1     0000000000
      2     ----------000000000000000
      3     -------------------------00000

   Disk

X = RUNNING on core X    . = WAITING IO - = READY
---------------------------------------
```

FCFS algorithm

Demo

```
 code git:(master) ✗ ./bin/cpuScheduler input.txt -c 1 -q 1 --algorithm=SJF
---------------- Stats ----------------
PID | PRIORITY | ARRIVAL | FINISH | TURNAROUND | CPU | WAITING | RESPONSE | C. SWITCHES
  1 |        1 |       0 |     15 |         15 |  10 |       5 |     5.00 |           0
  2 |        3 |       0 |     30 |         30 |  15 |      15 |    15.00 |           0
  3 |        2 |       0 |      5 |          5 |   5 |       0 |     0.00 |           0
---------------------------------------
---------------- Graph ----------------
Process     0    5   10   15   20   25   30
      1     -----0000000000
      2     ----------------000000000000000
      3     00000

   Disk

X = RUNNING on core X    . = WAITINGIO - = READY
---------------------------------------
```

SJF algorithm

Demo

```
  code git:(master) ✗ ./bin/cpuScheduler input.txt -c 1 -q 1 --algorithm=RR --RRSlice=5
---------------- Stats ----------------
PID | PRIORITY | ARRIVAL | FINISH | TURNAROUND | CPU | WAITING | RESPONSE | C. SWITCHES
  1 |        1 |       0 |     20 |         20 |  10 |      10 |     5.00 |           1
  2 |        3 |       0 |     30 |         30 |  15 |      15 |     7.50 |           1
  3 |        2 |       0 |     15 |         15 |   5 |      10 |    10.00 |           0
---------------------------------------
---------------- Graph ----------------
Process    0    5   10   15   20   25   30
      1    00000----------00000
      2    -----00000----------0000000000
      3    ----------00000

   Disk

X = RUNNING on core X    . = WAITINGIO - = READY
---------------------------------------
```

RR algorithm

Demo

```
  code git:(master) ✗ ./bin/cpuScheduler input.txt -c 1 -q 1 --algorithm=PRIORITY
----------------- Stats ------------------
PID | PRIORITY | ARRIVAL | FINISH | TURNAROUND | CPU | WAITING | RESPONSE | C. SWITCHES
  1 |        1 |       0 |     10 |         10 |  10 |       0 |     0.00 |           0
  2 |        3 |       0 |     30 |         30 |  15 |      15 |    15.00 |           0
  3 |        2 |       0 |     15 |         15 |   5 |      10 |    10.00 |           0
-----------------------------------------
----------------- Graph ------------------
Process     0    5   10   15   20   25   30
      1     0000000000
      2     ----------------000000000000000
      3     ----------00000

   Disk

X = RUNNING on core X    . = WAITING IO - = READY
-----------------------------------------
```

Priority algorithm

Demo

```
  code git:(master) ✗  ./bin/cpuScheduler input_with_IO.txt -c 1 -q 1 --algorithm=FCFS
---------------- Stats ----------------
PID | PRIORITY | ARRIVAL | FINISH | TURNAROUND | CPU | WAITING | RESPONSE | C. SWITCHES
  1 |        1 |       0 |     29 |         29 |   8 |      18 |     9.00 |           1
  2 |        3 |       0 |     18 |         18 |  15 |       2 |     2.00 |           0
  3 |        2 |       0 |     23 |         23 |   5 |      18 |    18.00 |           0
---------------------------------------
---------------- Graph ----------------
Process    0    5   10   15   20   25
      1    00...------------------000000
      2    --00-0000000000000
      3    ------------------00000

   Disk      11

X = RUNNING on core X    . = WAITING (IO)    - = READY
---------------------------------------
```

IO operations (different input file than before)

Demo

File: **input_multilevel.txt**

```
1  # pid, start_time, duration, priority, [list of timestamps and events] (IO, CPU)
2  1,  0, 30, 1, [(0, CPU)]
3  2,  0, 20, 1, [(0, CPU)]
4  3, 12, 10, 1, [(0, CPU)]
```

Input file contains multiple queues

```
 code git:(master) ✗ ./bin/cpuScheduler input_multilevel.txt -c 1 -q 3 --algorithm=RR --RRSlice=2 --limit=4 \
>                                                                    --algorithm=RR --RRSlice=3 --limit=9 \
>                                                                    --algorithm=FCFS
Process    0    5   10   15   20   25   30   35   40   45   50   55   60
      1    00--00--000-----000------000--------0000000000000000
      2    --00--00---0-------000------000---00-----------------0000000
      3                0000------000------000
```

Multiple queues with different scheduling algorithms

Requirements

Additional Information:

- Group of **two** that you will **keep** the whole semester (see discussion - eCampus).
- Submit a *tar.gz* archive on the submission platform (C code, Makefile *&* report.pdf).
- You are asked to write a **very short** report (max **1** page) in which you briefly explain your implementation and answer some specific questions.
- The statement and skeleton code will be available by the end of this week on eCampus. Further information in the statements.

Do not forget: We want clean code without error.

Do not forget too: We can detect **plagiarism** so do not try…
Plagiarism = **0 for the course!**

Deadline: 13th April 2024

Happy Coding!

Late submissions are accepted but with a penalty of $2^N - 1$ marks per day after the deadline.

Team composition

Team	Member1	Member2
Team 01	Mathieu Pasart	Denis Zolotariov
Team 02	de Thibault Adrien	Differdange Jarod
Team 03	Sébastien Laurent	Thibaud Vanmechelen
Team 04	Corentin Michaux	Raoul Saad
Team 05	Francois Grosjean	Théo Karras
Team 06	Maxim Piron	Nicolas Schneiders
Team 07	Nathan Kamps	Julie Ngamia Djabiri
Team 08	Pouria Katouzian	Manon Gerard
Team 09	Maé Klinkenberg	Victor Lecâne
Team 10	Dario Rinallo	Lei Yang
Team 11	Jeffrey Bienvenue	Florent Volvert
Team 12	Sélim Kadioglu	Théo Faingnaert
Team 13	Hansen Julien	Smagghe Clément
Team 14	Antoine Grosjean	Florent Hervers
Team 15	Gregory Voskertchian	Lionel La Rocca
Team 16	Jamaà Jair	Robin Fonbonne
Team 17	Jean-Philippe Kleijkers	José-Luis Diaz Thiele
Team 18	Robin Rademaker	Hiba Qouiqa
Team 19	Yassir Labib	Tom Cheniaux
Team 20	Thomas Rotheudt	Pierre Lorenzen
Team 21	Edward Pirnay	Andréa Stistrup
Team 22	Vermeylen Clément	Baguette Brice
Team 23	Jiaxiang Yao	Alyssia Kayembe
Team 24	Corentin Van Putte	Ayman Labrahimi
Team 25	Lukas Di Girolamo	Loic Denis
Team 26	David Fiorucci	Luca Heudt
Team 27	Eri Van de Vyver	Louan Robert
Team 28	Alexandre Andries	Abdelilah Khaliphi
Team 29	Raul-Mihai Talmacel	Morgan Phemba
Team 30	Jean Grifnee	Loup Grondal
Team 31	Alfonso Cifuentes Darriba	/

For the submission platform: use either *team_ID* or *group_ID* where ID is your team number.