# Operating Systems

## Practice .5
## Scheduling simulation

AI Software, Gachon University

# Objective

- Implementation of scheduling method
  - First Come First Served (non-preemptive)
  - Shortest Job First (non-preemptive)
  - Shortest Remaining Time First (preemptive)

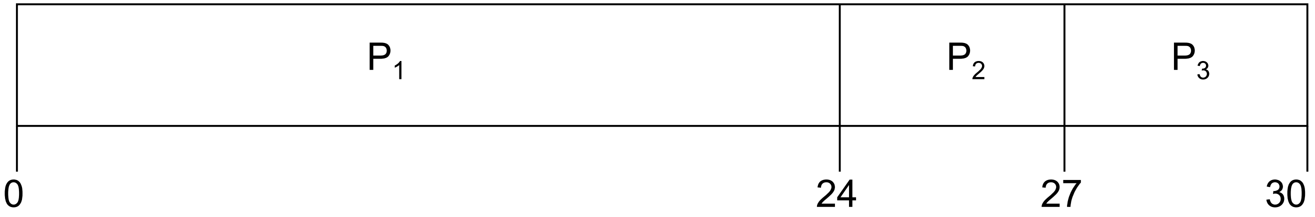
- And display the system performance results

- by fully understanding the scheduling algorithm,

# First-Come, First-Served (FCFS) Scheduling

| Process | CPU Burst Time |
|---|---|
| $P_1$ | 24 |
| $P_2$ | 3 |
| $P_3$ | 3 |

Suppose that the processes arrive in the order: $P_1$ , $P_2$ , $P_3$
The **Gantt Chart** for the schedule is:



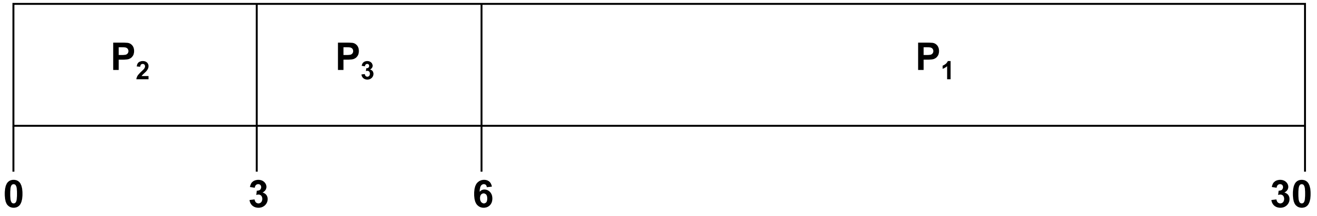
**Waiting time** for $P_1$ = 0; $P_2$ = 24; $P_3$ = 27
**Average waiting time**: (0 + 24 + 27)/3 = **17**

# FCFS Scheduling (Cont.)

Suppose that the processes arrive in the order:

$$P_2\ ,\ P_3\ ,\ P_1$$

- The Gantt chart for the schedule is:

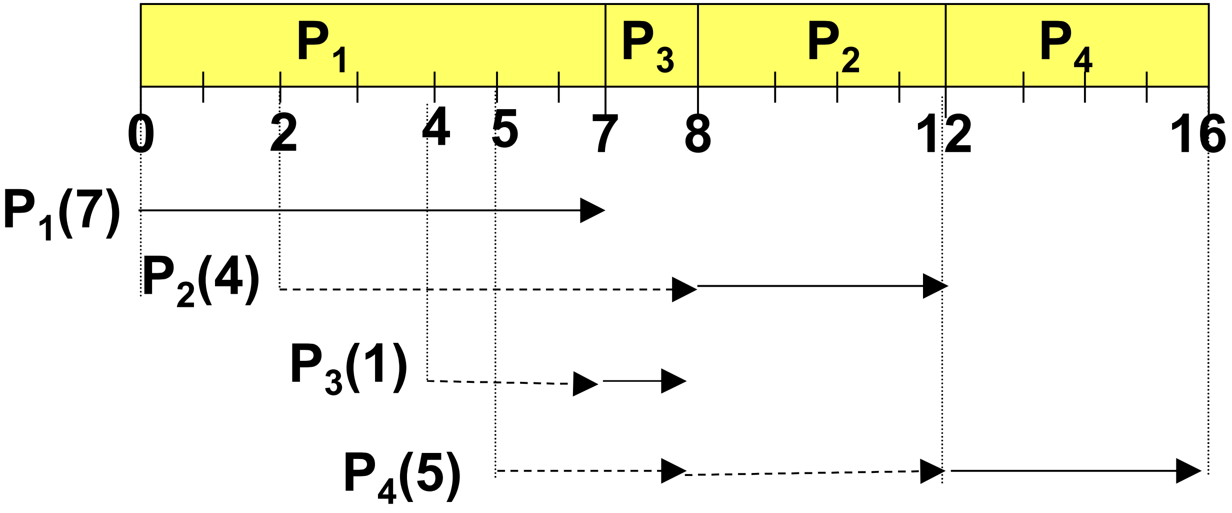


- Waiting time for $P_1 = 6; P_2 = 0; P_3 = 3$
- Average waiting time: (6 + 0 + 3)/3 = **3**
- Much better than the previous case

# Example of Non-Preemptive SJF

| Process | Arrival Time | Burst Time |
|---|---|---|
| $P_1$ | 0 | 7 |
| $P_2$ | 2 | 4 |
| $P_3$ | 4 | 1 |
| $P_4$ | 5 | 4 |

- SJF (non-preemptive)



$P_1$'s wating time = 0

$P_2$'s wating time = 6

$P_3$'s wating time = 3

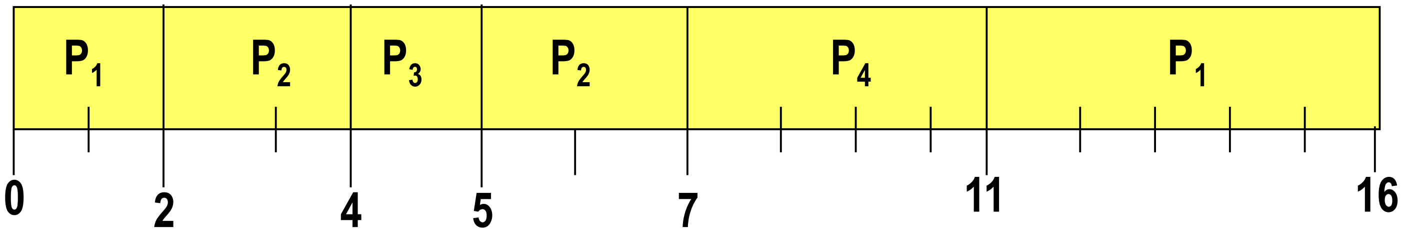
$P_4$'s wating time = 7

Average waiting time = (0 + 6 + 3 + 7)/4 = 4

# Example of Preemptive SJF (SRTF)

| Process | Arrival Time | Burst Time |
|---|---|---|
| $P_1$ | 0 | 7 |
| $P_2$ | 2 | 4 |
| $P_3$ | 4 | 1 |
| $P_4$ | 5 | 4 |

- SRTF (preemptive SJF)



$P_1$'s wating time = 9

$P_2$'s wating time = 1

$P_3$'s wating time = 0

$P_4$'s wating time = 2

Average waiting time = (9 + 1 + 0 +2)/4 = 3

# Mission

- Implementation of scheduling function
- Files provided
  - main.c
  - schedule.h
  - proc_list.txt

- You have to make **schedule.c** file and implement functions declared in *schedule.h*
- You should submit only “**schedule.c”** to CyberCampus

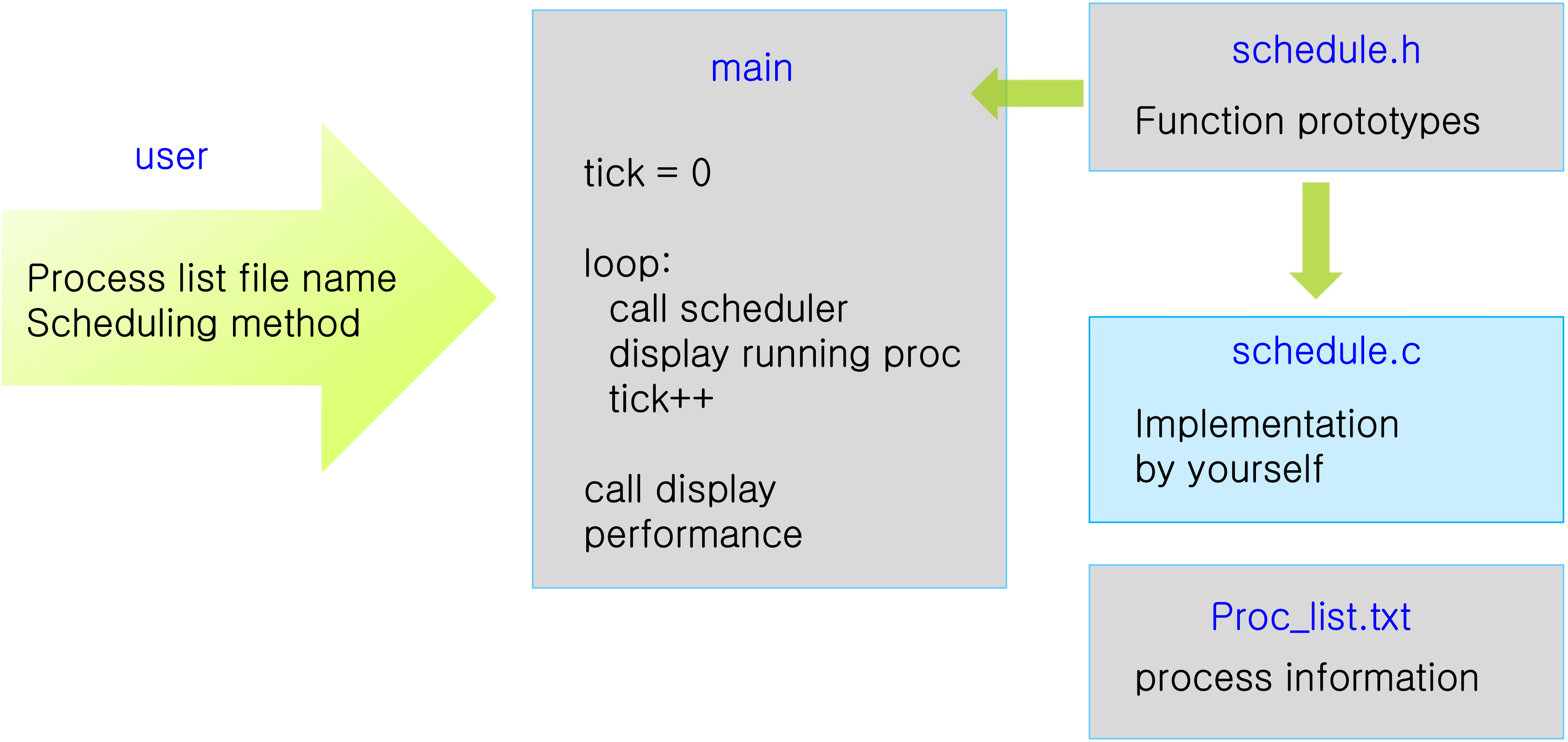
# Mission

given

homework

include

- structures



user

Process list file name
Scheduling method

main

```
tick = 0

loop:
  call scheduler
  display running proc
  tick++

call display
performance
```

schedule.h

Function prototypes

schedule.c

Implementation
by yourself

Proc_list.txt

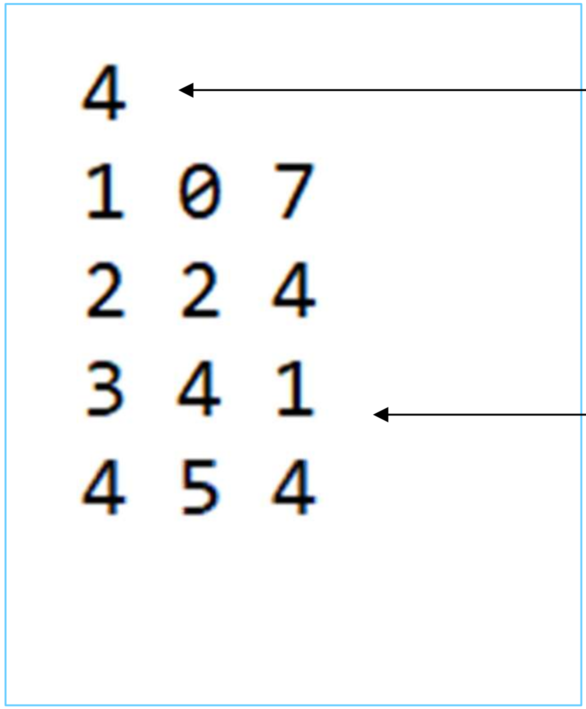
process information

# Process list

- Structure of a process list follows as below:
  - First line: the number of process
  - From second line
    - Process ID (1~10)
    - Process beginning time (0~100)
    - CPU burst time (1~20)

# Process list (Example)

- proc_list.txt



```
4
1 0 7
2 2 4
3 4 1
4 5 4
```

the number of process

| pid | beginning time | burst time |
|---|---|---|
| 1 | 0 | 7 |
| 2 | 2 | 4 |
| 3 | 4 | 1 |
| 4 | 5 | 4 |

# Code summary

- main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "schedule.h"

// global variables
int tick = 0;

int main(int argc, char* argv[])
{
    if (argc < 3) {
        printf("[usage] ./run [file_name] [scheduling_method]");
        return 0;
    }
    // set scheduling method
    set_schedule(atoi(argv[2]));
    // set processes
    read_proc_list(argv[1]);

    while(1){
        int res = do_schedule(tick);
        if (res < 0 || tick > 50) break;
        printf("[tick:%04d] CPU is allocated to process No.%02d₩n", tick, res);
        tick++;
    }
    print_performance();
    return 0;
}
```

Yellow-marked lines
are from schedule.h

# Code summary

- schedule.h

```
#ifndef __schedule_h__
#define __schedule_h__

void read_proc_list(const char* file_name);

void set_schedule(int method);

int do_schedule(int tick);

void print_performance();

#endif
```

# Code summary

- schedule.h

```
// fn: read_proc_list
// desc: read process file list
// param
//   file_name: process list name
void read_proc_list(const char* file_name);
```

  - The function reads the file and initializes processes

# Code summary

- schedule.h

```
// fn: set_schedule
// desc: set scheduling method
//
// param: method
//  scheduling method
//  1. FCFS (Nonpreemptive)
//  2. Shortest Job First (Nonpreemptive)
//  3. Shortest Remaining Time First (Prremptive)
//
// return none
void set_schedule(int method);
```

# Code summary

- schedule.h

```
// fn: do_schedule
// desc: scheduling function called every tick from main
// param
//  tick: time tick beginning from 0
// return
//     -1: when all process are terminated
//      0: CPU is idle
// others: PID of running state
int do_schedule(int tick);
```

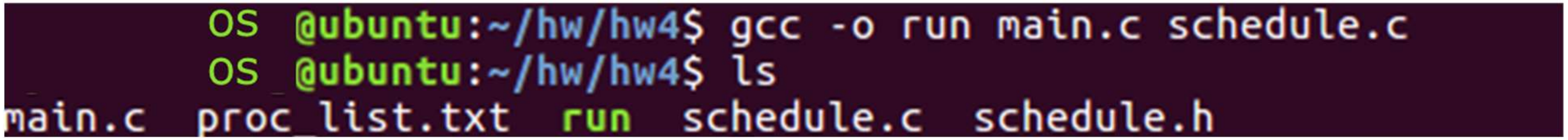
# Code summary

- schedule.h

```
// fn: print_performance();
// desc: print system performance
void print_performance();
```

# Compile

- gcc –o run main.c schedule.c

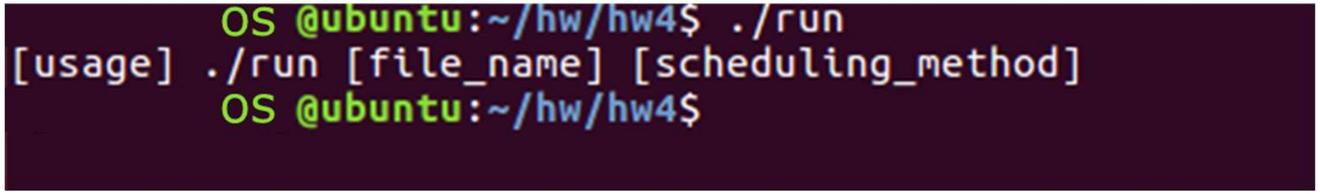


```
          OS  @ubuntu:~/hw/hw4$ gcc -o run main.c schedule.c
          OS  @ubuntu:~/hw/hw4$ ls
main.c  proc_list.txt  run  schedule.c  schedule.h
```

- Note, schedule.c is not included in provided files.
- You should create the file first!!

# Run

- ./run [process_file] [scheduling method]
  - process_file: text file name
  - scheduling method: 1-FCFS, 2-SJF, 3-SRTF
- Example.
  - ./run proc_list.txt 1
- Failure case



```
          OS @ubuntu:~/hw/hw4$ ./run
[usage] ./run [file_name] [scheduling_method]
          OS @ubuntu:~/hw/hw4$
```

# Result format

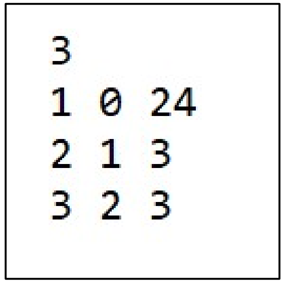
- For every tick, running process should be shown
  - [tick:000X] CPU is allocated to process No. XX
  - This is implemented in main function. You have nothing to do.

- After all processes are terminated.
  - Display system performance as below: (0 means a digit)

| PID | Begin (Arrival) | Finish | Turn around time | Waiting time | Response time |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 |
| … | 0 | 0 | 0 | 0 | 0 |
| | average | | 00.00 | 00.00 | 00.00 |

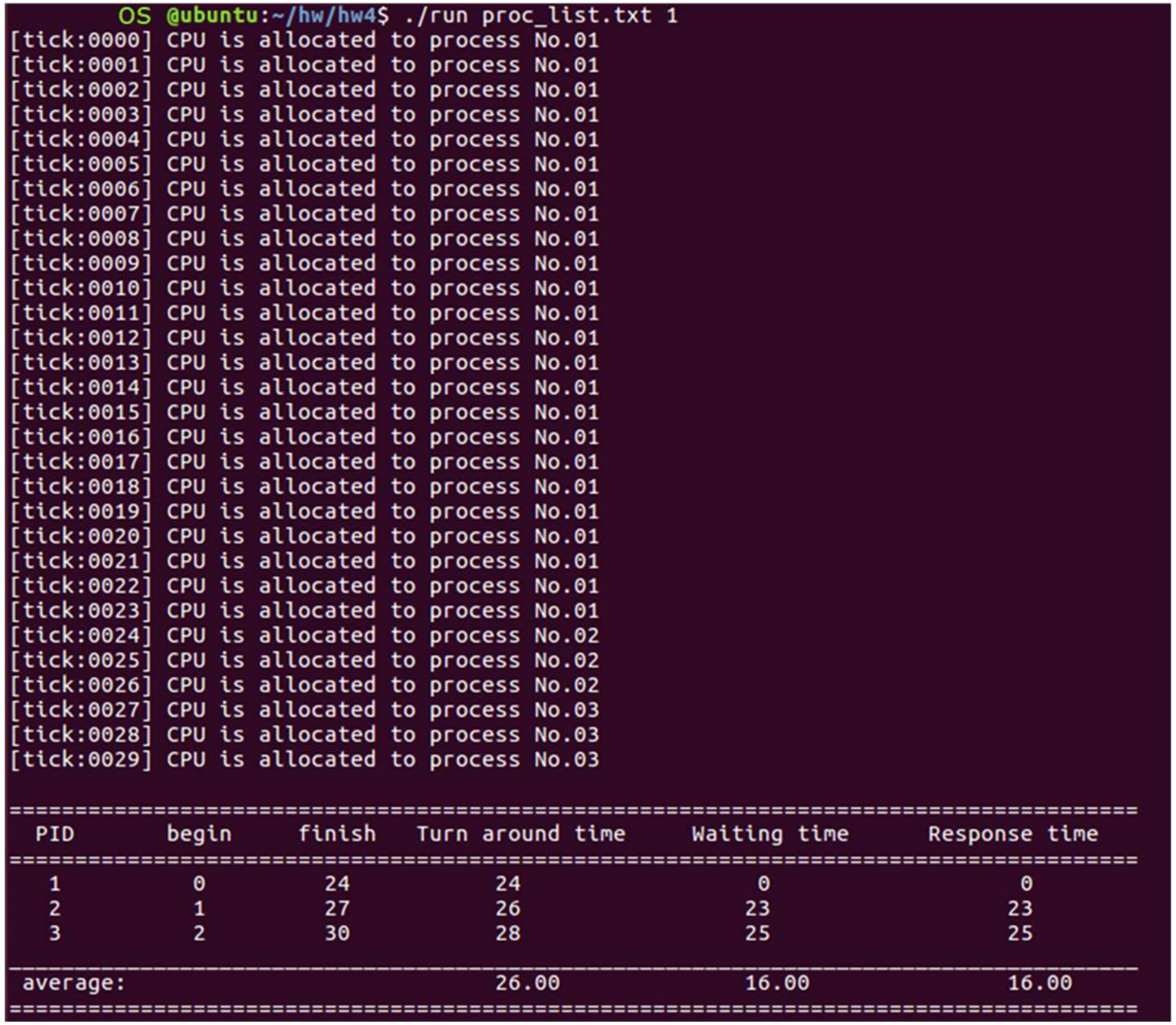
# Result Example

- Scheduling using FCFS



```
3
1 0 24
2 1 3
3 2 3
```

proc_list.txt

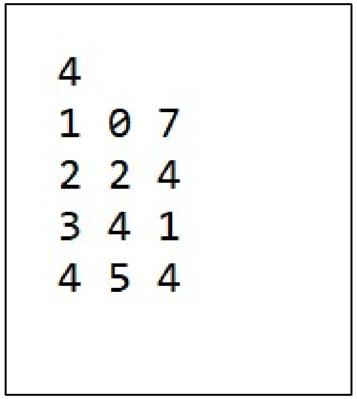


```
       OS @ubuntu:~/hw/hw4$ ./run proc_list.txt 1
[tick:0000] CPU is allocated to process No.01
[tick:0001] CPU is allocated to process No.01
[tick:0002] CPU is allocated to process No.01
[tick:0003] CPU is allocated to process No.01
[tick:0004] CPU is allocated to process No.01
[tick:0005] CPU is allocated to process No.01
[tick:0006] CPU is allocated to process No.01
[tick:0007] CPU is allocated to process No.01
[tick:0008] CPU is allocated to process No.01
[tick:0009] CPU is allocated to process No.01
[tick:0010] CPU is allocated to process No.01
[tick:0011] CPU is allocated to process No.01
[tick:0012] CPU is allocated to process No.01
[tick:0013] CPU is allocated to process No.01
[tick:0014] CPU is allocated to process No.01
[tick:0015] CPU is allocated to process No.01
[tick:0016] CPU is allocated to process No.01
[tick:0017] CPU is allocated to process No.01
[tick:0018] CPU is allocated to process No.01
[tick:0019] CPU is allocated to process No.01
[tick:0020] CPU is allocated to process No.01
[tick:0021] CPU is allocated to process No.01
[tick:0022] CPU is allocated to process No.01
[tick:0023] CPU is allocated to process No.01
[tick:0024] CPU is allocated to process No.02
[tick:0025] CPU is allocated to process No.02
[tick:0026] CPU is allocated to process No.02
[tick:0027] CPU is allocated to process No.03
[tick:0028] CPU is allocated to process No.03
[tick:0029] CPU is allocated to process No.03

==============================================================================================
  PID        begin      finish    Turn around time      Waiting time        Response time
==============================================================================================
   1           0          24            24                   0                    0
   2           1          27            26                  23                   23
   3           2          30            28                  25                   25
----------------------------------------------------------------------------------------------
 average:                             26.00                16.00                16.00
==============================================================================================
```

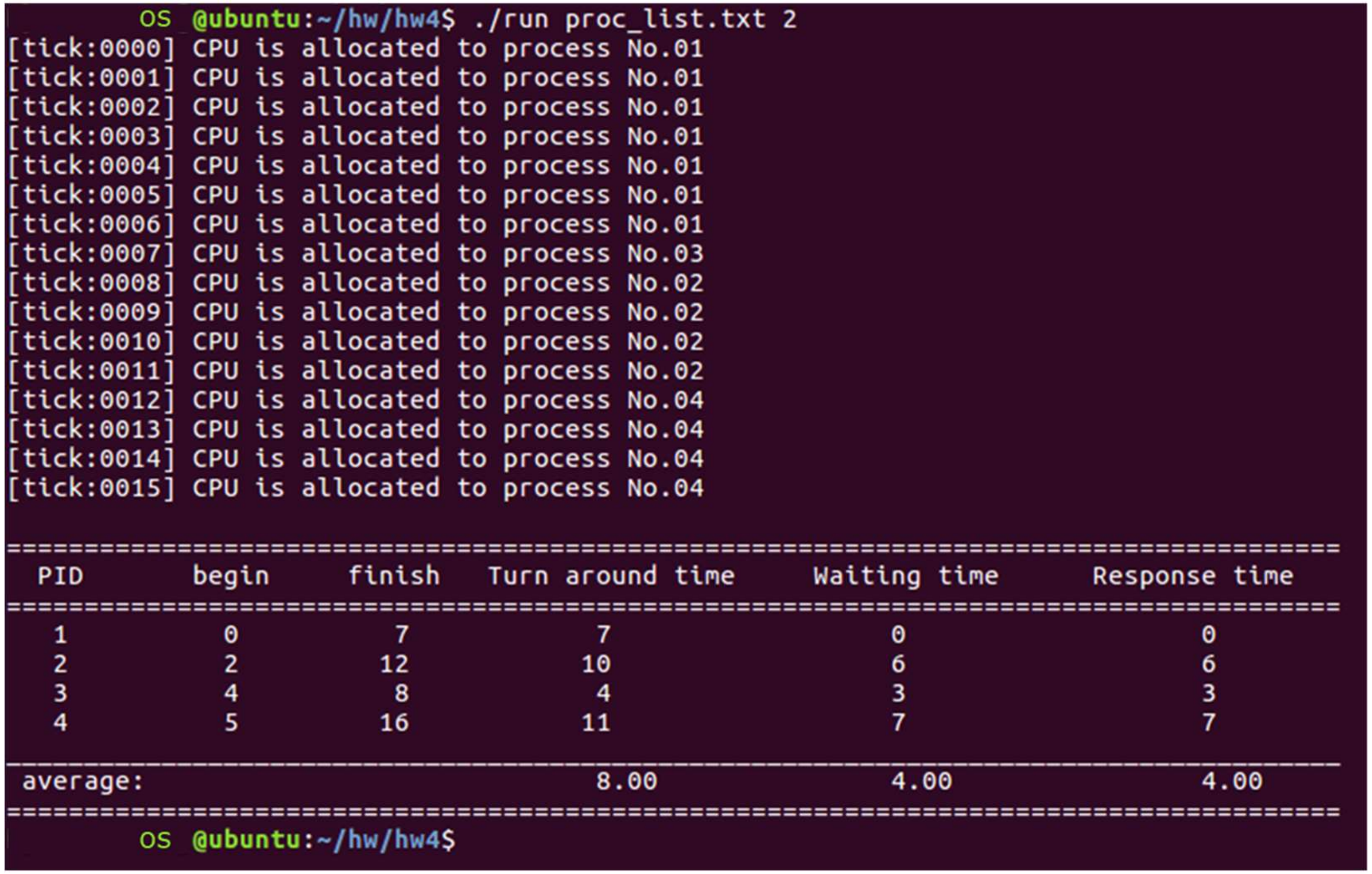
# Result Example

- Scheduling using SJF



```
4
1 0 7
2 2 4
3 4 1
4 5 4
```

proc_list.txt



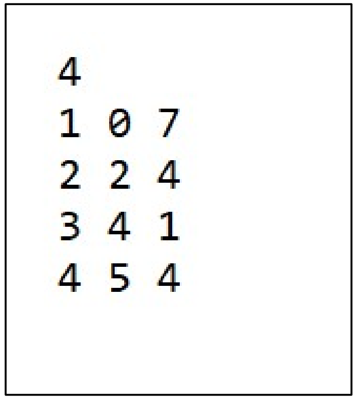
```
        OS  @ubuntu:~/hw/hw4$ ./run proc_list.txt 2
[tick:0000] CPU is allocated to process No.01
[tick:0001] CPU is allocated to process No.01
[tick:0002] CPU is allocated to process No.01
[tick:0003] CPU is allocated to process No.01
[tick:0004] CPU is allocated to process No.01
[tick:0005] CPU is allocated to process No.01
[tick:0006] CPU is allocated to process No.01
[tick:0007] CPU is allocated to process No.03
[tick:0008] CPU is allocated to process No.02
[tick:0009] CPU is allocated to process No.02
[tick:0010] CPU is allocated to process No.02
[tick:0011] CPU is allocated to process No.02
[tick:0012] CPU is allocated to process No.04
[tick:0013] CPU is allocated to process No.04
[tick:0014] CPU is allocated to process No.04
[tick:0015] CPU is allocated to process No.04

======================================================================================
  PID        begin      finish   Turn around time    Waiting time      Response time
======================================================================================
   1           0          7            7                  0                   0
   2           2         12           10                  6                   6
   3           4          8            4                  3                   3
   4           5         16           11                  7                   7

--------------------------------------------------------------------------------------
 average:                               8.00               4.00                4.00
======================================================================================
        OS  @ubuntu:~/hw/hw4$
```

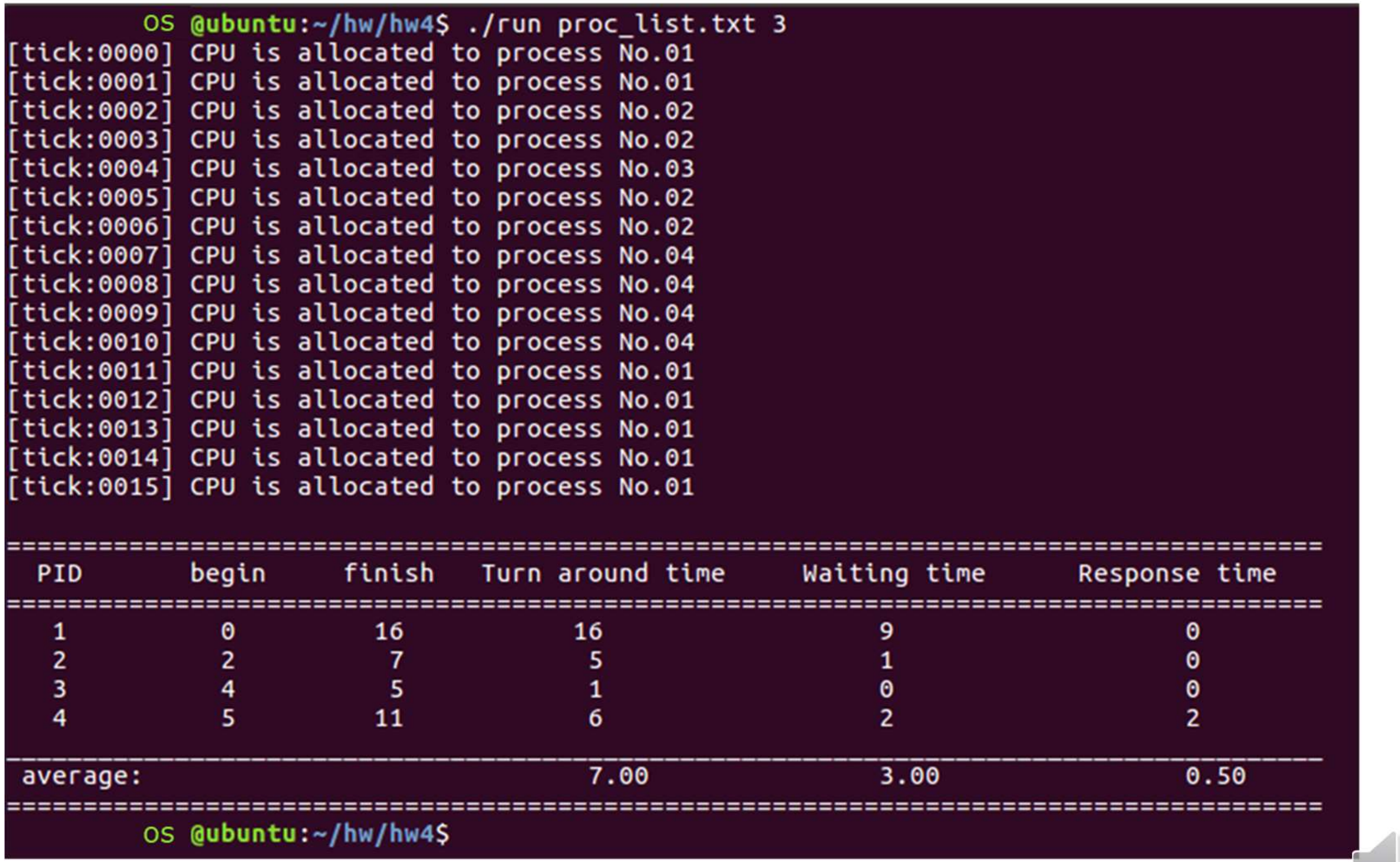
# Result Example

- Scheduling using SRTF



```
4
1 0 7
2 2 4
3 4 1
4 5 4
```

proc_list.txt



```
        OS @ubuntu:~/hw/hw4$ ./run proc_list.txt 3
[tick:0000] CPU is allocated to process No.01
[tick:0001] CPU is allocated to process No.01
[tick:0002] CPU is allocated to process No.02
[tick:0003] CPU is allocated to process No.02
[tick:0004] CPU is allocated to process No.03
[tick:0005] CPU is allocated to process No.02
[tick:0006] CPU is allocated to process No.02
[tick:0007] CPU is allocated to process No.04
[tick:0008] CPU is allocated to process No.04
[tick:0009] CPU is allocated to process No.04
[tick:0010] CPU is allocated to process No.04
[tick:0011] CPU is allocated to process No.01
[tick:0012] CPU is allocated to process No.01
[tick:0013] CPU is allocated to process No.01
[tick:0014] CPU is allocated to process No.01
[tick:0015] CPU is allocated to process No.01

==========================================================================================
  PID        begin      finish    Turn around time     Waiting time       Response time
==========================================================================================
   1           0          16            16                  9                    0
   2           2           7             5                  1                    0
   3           4           5             1                  0                    0
   4           5          11             6                  2                    2
------------------------------------------------------------------------------------------
 average:                               7.00                3.00                 0.50
==========================================================================================
        OS @ubuntu:~/hw/hw4$
```

# Rules

- Do not modify **main.c** or **schedule.h**
  - We will evaluate using the provided version of the two files.
  - Modification by you will not affect at all.

- When two processes have the same priority
  - The process in running state use CPU continuously (due to cost of context switching)
  - If both processes are in ready state, it doesn't matter which one you choose. (the result will be same).

# TIP #1

- System performance
  - Turn around time
    - finish tick – beginning tick
  - Waiting time
    - finish tick – beginning tick – burst tick
  - Response time
    - first tick the CPU was allocated – beginning tick

# TIP #2

- If you use the **structure well**, you can easily solve the problem.

- Example

  Struct PCB

  {

    - int pid;
    - Int begin_tick;
    - Int burst_tick;
    - …

  }

# Scoring criteria

- Total: 200
  - Submission: 30
  - Compile success: 30
  - Evaluation using a random process list with three scheduling method
    - For each Scheduling method
      - Correct result of running process per every tick (10)
      - Correct result for turn around time (10)
      - Correct result for waiting time (10)
      - Correct result for response time (10)
    - Subtotal:  (3 x 40 = )120
  - Code review: 20, **Word file needs to be submitted.**

# Due

- May 11, 23:59 (around 2 weeks after mid-term exam)
  - Submit to the CyberCampus
    - “schedule.c” file only: insert your ID + name in the first line of the code
  - No late submission will be allowed