Exercise 1 (1/2)

- Write two programs **agent.c** (Agent) and **controller.c** (Controller) and create a file **text.txt** which contains some message.
- On startup, the agent program creates the file `/var/run/agent.pid` and writes its `PID` into the file. It then reads the contents of a file called **text.txt**, prints it on the console and finally sleeps indefinitely.
- The agent would be sensitive to two signal types: SIGUSR1 and SIGUSR2. When the agent receives a SIGUSR1, it reads the contents of the text from **text.txt** and prints it on the console; when it receives a SIGUSR2, it prints "Process terminating..." and then exits.

Exercise 1 (2/2)

- On startup, the controller checks for a running agent by fetching the agent's PID from the file `/var/run/agent.pid`. If it cannot find an agent, it prints "Error: No agent found." and then exits; otherwise, it prints "Agent found." and continues.
- Then in an infinite loop, the controller prints "Choose a command {"read", "exit", "stop", "continue"} to send to the agent" and then waits for user input. The user enters one of the commands.
- When the user enters his/her choice, the controller sends the corresponding signal {"read":"SIGUSR1", "exit":"SIGUSR2", "stop":"SIGSTOP", "continue":"SIGCONT"} to the agent. If the user chooses "exit", the controller should also terminate after sending the signal. If the user presses Ctrl+C, the controller should send a SIGTERM to the agent and then exit.
- Submit **agent.c** and **controller.c**.

Exercise 2 (1/15)

worker.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>
#include <unistd.h>
#include <signal.h>
#define TRI_BASE 1000000

// current process pid
pid_t pid;
// current process idx (starts from 0)
int process_idx;
// number of triangular numbers found
long tris;
// checks if n is triangular
bool is_triangular(int n){
    for (int i = 1; i <= n; i++){
        if (i * (i + 1) == 2 * n){
            return true;
        }
    }
    return false;
}
```

```c
// generates a big rand number n
long big_n() {
    time_t t; long n = 0;
    srand((unsigned) time(&t));
    while(n < TRI_BASE) n += rand();
    return n % TRI_BASE;
}

// Given: signal handler
void signal_handler(int signum);
```

Exercise 2 (2/15)

scheduler.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>

typedef struct {
    int idx;
    int at, bt, rt, wt, ct, tat;
    int burst; // remaining burst
} ProcessData;

// idx of running process
// -1 means no running processes
int running_process = -1;
// the total time of the timer
unsigned total_time;
// data of the worker processes
ProcessData data[PS_MAX];
// array of all worker processes
pid_t ps[PS_MAX];
// size of data array
unsigned data_size;

//TODO: read data from file and store
    it in data array
void read_file(FILE* file);
```

```c
//TODO: send signal SIGCONT to worker
    process
void resume(pid_t process);
//TODO: send signal SIGTSTP to worker
    process
void suspend(pid_t process);
//TODO: send signal SIGTERM to worker
    process
void terminate(pid_t process);

//TODO: create a process using fork
void create_process(int new_process);

// TODO: find next process for running
ProcessData find_next_process();

// Given: reports the metrics and
    simulation results
void report();

// Given: scheduler waits for all
    processes
void check_burst();

// TODO: schedule processes
void schedule_handler(int signum);
```

Exercise 2 (3/15)

- Assume that we have only one CPU with one core. Only one process can run simultaneously and no race conditions might occur. You **should** use the code templates in this gist. The code snippets in the previous slides contain only function signatures.
- Write a program **scheduler.c** (scheduler) to simulate the first come, first served (FCFS) algorithm using a timer. Write another program **worker.c** (worker process) which is responsible for finding and counting the **next** triangular numbers from the range (n, ∞) where n is a big number randomly generated by the function $long\ big\_n()$.
- In this simulation, each process has an index (idx), arrival time (at) and burst time (bt). The scheduler reads the data of all processes from a file $data.txt$ where it contains $(m + 1)$ rows and 3 columns **(idx, at, bt)**. Here, we can have at most 10 processes where the actual number of processes is m.

Exercise 2 (4/15)

- Then, the scheduler runs a timer (`struct itimerval`) and sends a signal *SIGALRM* every second. You need to use a timer and set it to decrement in real time. You need to write the function *schedule_handler* which schedules the processes according to the given algorithm, in this exercise we have FCFS algorithm.
- The worker process **(worker.c)** in the main function will read its **idx** as a command line argument. Then it registers the same handler **signal_handler** for all three signals (SIGTSTP, SIGCONT, SIGTERM). The worker process generates a number n using **big_n** function. In an infinite loop, the worker checks the candiadate for the next triangular number **next_n** using **is_triangular**, when it finds the triangular number **next_n**, it needs to increments the counter **tris** and then checks again for the next number **(next_n + 1)** till termination by the scheduler.

Exercise 2 (5/15)

- The function `create_process`, firstly, stops any running workers, then, forks a new worker process and runs an instance of worker program **worker.c** using **execvp** where the argument `new_process` is the idx of the new process and passed as a command line argument to the worker program.
- The function `find_next_process`, will return the **ProcessData** of the *next process to run* according to the algorithm. In FCFS, you need to search in the **data** array for the next process based on their arrival time. In case, more than one process have the same arrival time then we can take the process with smallest idx as the next process to run. In case there are no processes arrived yet, then you need to find the next process again after incrementing the `total_time` till finding one process.
- The shcheduler terminates when the burst time of all processes becomes zero and prints a report before termination.

Exercise 2 (6/15)

- The implementation should calculate the following metrics and add to the **ProcessData** of each process: (**Note:** Go to the last slides to see the formula of the required metrics)
  - Response time (RT)
  - Completion time(CT)
  - Turn around time(TAT)
  - Waiting time(WT)
  - Average Turnaround time
  - Average waiting time
- Based on your implementation, answer the following questions and add your answers to a file **ex2.txt**.
  - Change the macro **TRI_BASE** to a small number like 100 and run the scheduler again. Compare your results before and after the change?
  - Change the arrival time of all processes to make all of them zero, then run the scheduler again. In which order will processes be executed?

Exercise 2 (7/15)

- Submit **worker.c**, **scheduler.c**, **ex2.txt** with a script **ex2.sh** to run the program.
- **Note:** The user runs only the scheduler program passing **data.txt** as a command line argument, then the scheduler will run workers based on the given data. Do not forget to compile the **worker.c** before running **scheduler.c**.

Exercise 2 (8/15)

Example: The input is a **data.txt** file as follows:

```
idx at bt
0   5    2
1        7          6
2        20   3
3        3  8
4  2                    4
5  3                    1
6  10                    6
```

which can be visualized as a table:

idx	at	bt
0	5	2
1	7	6
2	20	3
3	3	8
4	2	4
5	3	1
6	10	6

The output is in the next slides.

(Note: Some screenshots are taken intentionally starting from the last line of the previous screenshot for continuity)

Exercise 2 (9/15)

```
Scheduler: Runtime: 1 seconds.
Process 4: has not arrived yet.
Scheduler: Starting Process 4 (Remaining Time: 4)
Process 4 (PID=<252152>): has been started
Process 4 (PID=<252152>): will find the next trianguar number from (954860, inf)
Scheduler: Runtime: 3 seconds.
Scheduler: Process 4 is running with 3 seconds left
Process 4 (PID=<252152>): I found this trianguar number 955653
Scheduler: Runtime: 4 seconds.
Scheduler: Process 4 is running with 2 seconds left
Scheduler: Runtime: 5 seconds.
Scheduler: Process 4 is running with 1 seconds left
Scheduler: Runtime: 6 seconds.
Scheduler: Process 4 is running with 0 seconds left
Scheduler: Terminating Process 4 (Remaining Time: 0)
Process 4 (PID=<252152>): count of triangulars found so far is 1
Process 4: terminating....
Scheduler: Starting Process 3 (Remaining Time: 8)
Process 3 (PID=<252153>): has been started
Process 3 (PID=<252153>): will find the next trianguar number from (353906, inf)
Process 3 (PID=<252153>): I found this trianguar number 354061
Process 3 (PID=<252153>): I found this trianguar number 354903
Scheduler: Runtime: 7 seconds.
Scheduler: Process 3 is running with 7 seconds left
Process 3 (PID=<252153>): I found this trianguar number 355746
Scheduler: Runtime: 8 seconds.
Scheduler: Process 3 is running with 6 seconds left
Process 3 (PID=<252153>): I found this trianguar number 356590
Process 3 (PID=<252153>): I found this trianguar number 357435
Scheduler: Runtime: 9 seconds.
Scheduler: Process 3 is running with 5 seconds left
Process 3 (PID=<252153>): I found this trianguar number 358281
Process 3 (PID=<252153>): I found this trianguar number 359128
```

Exercise 2 (10/15)

```
Process 3 (PID=<252153>): I found this trianguar number 359128
Scheduler: Runtime: 10 seconds.
Scheduler: Process 3 is running with 4 seconds left
Process 3 (PID=<252153>): I found this trianguar number 359976
Scheduler: Runtime: 11 seconds.
Scheduler: Process 3 is running with 3 seconds left
Process 3 (PID=<252153>): I found this trianguar number 360825
Process 3 (PID=<252153>): I found this trianguar number 361675
Scheduler: Runtime: 12 seconds.
Scheduler: Process 3 is running with 2 seconds left
Process 3 (PID=<252153>): I found this trianguar number 362526
Process 3 (PID=<252153>): I found this trianguar number 363378
Scheduler: Runtime: 13 seconds.
Scheduler: Process 3 is running with 1 seconds left
Process 3 (PID=<252153>): I found this trianguar number 364231
Scheduler: Runtime: 14 seconds.
Scheduler: Process 3 is running with 0 seconds left
Scheduler: Terminating Process 3 (Remaining Time: 0)
Process 3 (PID=<252153>): count of triangulars found so far is 13
Process 3: terminating....
Scheduler: Starting Process 5 (Remaining Time: 1)
Process 5 (PID=<252154>): has been started
Process 5 (PID=<252154>): will find the next trianguar number from (110156, inf)
Process 5 (PID=<252154>): I found this trianguar number 110215
Process 5 (PID=<252154>): I found this trianguar number 110685
Process 5 (PID=<252154>): I found this trianguar number 111156
Process 5 (PID=<252154>): I found this trianguar number 111628
Process 5 (PID=<252154>): I found this trianguar number 112101
Process 5 (PID=<252154>): I found this trianguar number 112575
Process 5 (PID=<252154>): I found this trianguar number 113050
Process 5 (PID=<252154>): I found this trianguar number 113526
Scheduler: Runtime: 15 seconds.
Scheduler: Process 5 is running with 0 seconds left
```

Exercise 2 (11/15)

```
Scheduler: Process 5 is running with 0 seconds left
Scheduler: Terminating Process 5 (Remaining Time: 0)
Process 5 (PID=<252154>): count of triangulars found so far is 8
Process 5: terminating....
Scheduler: Starting Process 0 (Remaining Time: 2)
Process 0 (PID=<252155>): has been started
Process 0 (PID=<252155>): will find the next trianguar number from (520288, inf)
Process 0 (PID=<252155>): I found this trianguar number 520710
Scheduler: Runtime: 16 seconds.
Scheduler: Process 0 is running with 1 seconds left
Process 0 (PID=<252155>): I found this trianguar number 521731
Scheduler: Runtime: 17 seconds.
Scheduler: Process 0 is running with 0 seconds left
Scheduler: Terminating Process 0 (Remaining Time: 0)
Process 0 (PID=<252155>): count of triangulars found so far is 2
Process 0: terminating....
Scheduler: Starting Process 1 (Remaining Time: 6)
Process 1 (PID=<252156>): has been started
Process 1 (PID=<252156>): will find the next trianguar number from (121458, inf)
Process 1 (PID=<252156>): I found this trianguar number 121771
Process 1 (PID=<252156>): I found this trianguar number 122265
Process 1 (PID=<252156>): I found this trianguar number 122760
Process 1 (PID=<252156>): I found this trianguar number 123256
Process 1 (PID=<252156>): I found this trianguar number 123753
Process 1 (PID=<252156>): I found this trianguar number 124251
Process 1 (PID=<252156>): I found this trianguar number 124750
Scheduler: Runtime: 18 seconds.
Scheduler: Process 1 is running with 5 seconds left
Process 1 (PID=<252156>): I found this trianguar number 125250
Process 1 (PID=<252156>): I found this trianguar number 125751
Process 1 (PID=<252156>): I found this trianguar number 126253
Process 1 (PID=<252156>): I found this trianguar number 126756
Process 1 (PID=<252156>): I found this trianguar number 127260
```

Exercise 2 (12/15)

```
Process 1 (PID=<252156>): I found this trianguar number 127260
Process 1 (PID=<252156>): I found this trianguar number 127765
Scheduler: Runtime: 19 seconds.
Scheduler: Process 1 is running with 4 seconds left
Process 1 (PID=<252156>): I found this trianguar number 128271
Process 1 (PID=<252156>): I found this trianguar number 128778
Process 1 (PID=<252156>): I found this trianguar number 129286
Process 1 (PID=<252156>): I found this trianguar number 129795
Process 1 (PID=<252156>): I found this trianguar number 130305
Process 1 (PID=<252156>): I found this trianguar number 130816
Scheduler: Runtime: 20 seconds.
Scheduler: Process 1 is running with 3 seconds left
Process 1 (PID=<252156>): I found this trianguar number 131328
Process 1 (PID=<252156>): I found this trianguar number 131841
Process 1 (PID=<252156>): I found this trianguar number 132355
Process 1 (PID=<252156>): I found this trianguar number 132870
Process 1 (PID=<252156>): I found this trianguar number 133386
Process 1 (PID=<252156>): I found this trianguar number 133903
Process 1 (PID=<252156>): I found this trianguar number 134421
Scheduler: Runtime: 21 seconds.
Scheduler: Process 1 is running with 2 seconds left
Process 1 (PID=<252156>): I found this trianguar number 134940
Process 1 (PID=<252156>): I found this trianguar number 135460
Process 1 (PID=<252156>): I found this trianguar number 135981
Process 1 (PID=<252156>): I found this trianguar number 136503
Process 1 (PID=<252156>): I found this trianguar number 137026
Process 1 (PID=<252156>): I found this trianguar number 137550
Scheduler: Runtime: 22 seconds.
Scheduler: Process 1 is running with 1 seconds left
Process 1 (PID=<252156>): I found this trianguar number 138075
Process 1 (PID=<252156>): I found this trianguar number 138601
Process 1 (PID=<252156>): I found this trianguar number 139128
Process 1 (PID=<252156>): I found this trianguar number 139656
```

Exercise 2 (13/15)

```
Process 1 (PID=<252156>): I found this trianguar number 139656
Process 1 (PID=<252156>): I found this trianguar number 140185
Process 1 (PID=<252156>): I found this trianguar number 140715
Scheduler: Runtime: 23 seconds.
Scheduler: Process 1 is running with 0 seconds left
Scheduler: Terminating Process 1 (Remaining Time: 0)
Process 1 (PID=<252156>): count of triangulars found so far is 38
Process 1: terminating....
Scheduler: Starting Process 6 (Remaining Time: 6)
Process 6 (PID=<252157>): has been started
Process 6 (PID=<252157>): will find the next trianguar number from (594687, inf)
Scheduler: Runtime: 24 seconds.
Scheduler: Process 6 is running with 5 seconds left
Process 6 (PID=<252157>): I found this trianguar number 595686
Scheduler: Runtime: 25 seconds.
Scheduler: Process 6 is running with 4 seconds left
Process 6 (PID=<252157>): I found this trianguar number 596778
Scheduler: Runtime: 26 seconds.
Scheduler: Process 6 is running with 3 seconds left
Process 6 (PID=<252157>): I found this trianguar number 597871
Scheduler: Runtime: 27 seconds.
Scheduler: Process 6 is running with 2 seconds left
Scheduler: Runtime: 28 seconds.
Scheduler: Process 6 is running with 1 seconds left
Process 6 (PID=<252157>): I found this trianguar number 598965
Scheduler: Runtime: 29 seconds.
Scheduler: Process 6 is running with 0 seconds left
Scheduler: Terminating Process 6 (Remaining Time: 0)
Process 6 (PID=<252157>): count of triangulars found so far is 4
Process 6: terminating....
Scheduler: Starting Process 2 (Remaining Time: 3)
Process 2 (PID=<252158>): has been started
Process 2 (PID=<252158>): will find the next trianguar number from (767457, inf)
```

Exercise 2 (14/15)

```
Process 2 (PID=<252158>): will find the next trianguar number from (767457, inf)
Scheduler: Runtime: 30 seconds.
Scheduler: Process 2 is running with 2 seconds left
Process 2 (PID=<252158>): I found this trianguar number 768180
Scheduler: Runtime: 31 seconds.
Scheduler: Process 2 is running with 1 seconds left
Scheduler: Runtime: 32 seconds.
Scheduler: Process 2 is running with 0 seconds left
Scheduler: Terminating Process 2 (Remaining Time: 0)
Process 2 (PID=<252158>): count of triangulars found so far is 1
Process 2: terminating....
Simulation results.....
process 0:
        at=5
        bt=2
        ct=17
        wt=10
        tat=12
        rt=10
process 1:
        at=7
        bt=6
        ct=23
        wt=10
        tat=16
        rt=10
process 2:
        at=20
        bt=3
        ct=32
        wt=9
        tat=12
        rt=9
```

Exercise 2 (15/15)

```
Process 2 (PID=<252158>): will find the next trianguar number from (767457, inf)
Scheduler: Runtime: 30 seconds.
Scheduler: Process 2 is running with 2 seconds left
Process 2 (PID=<252158>): I found this trianguar number 768180
Scheduler: Runtime: 31 seconds.
Scheduler: Process 2 is running with 1 seconds left
Scheduler: Runtime: 32 seconds.
Scheduler: Process 2 is running with 0 seconds left
Scheduler: Terminating Process 2 (Remaining Time: 0)
Process 2 (PID=<252158>): count of triangulars found so far is 1
Process 2: terminating....
Simulation results.....
process 0:
        at=5
        bt=2
        ct=17
        wt=10
        tat=12
        rt=10
process 1:
        at=7
        bt=6
        ct=23
        wt=10
        tat=16
        rt=10
process 2:
        at=20
        bt=3
        ct=32
        wt=9
        tat=12
        rt=9
```

Exercise 3

- Modify the program **scheduler.c** and write the solution of this exercise in **scheduler_sjf.c**, in which you should implement shortest job first algorithm (non-preemptive version).
- The implementation should follow the same requirements as the previous exercise. Use the same code snippets with some modification.
- Compare the output with the outputs of the previous exercise and save the explanation in **ex3.txt**
- Submit **ex3.txt**, **scheduler_sjf.c** with a supplement script **ex3.sh** to run the program.
- **Note:** You do not need to change **worker.c**. To implement SJF here, you need to modify only the functions **find_next_process** and **schedule_handler**.

Exercise 4

- Modify the program **scheduler.c** and write the solution of this exercise in **scheduler_rr.c**, in which you should implement round-robin algorithm.
- The implementation should follow the same requirements as the previous exercise. Use the same code snippets with some modification.
- Compare the output with the outputs of the previous two exercise and save the explanation in **ex4.txt**
- Submit **ex4.txt**, **scheduler_rr.c**, and script **ex4.sh** to run the program. **Note:** for this algorithm, the quantum should be specified by the user from stdin in **scheduler_rr.c**.
- **Note:** You do not need to change **worker.c**. To implement Round robin algorithm here, you need to modify the functions **find_next_process** and **schedule_handler**. You may also need to add another field to **ProcessData** struct for storing the remaining slice of quantum.