Process Scheduling project guide

Malek Slokom - Mouna Rekik

1 Description of the project

Multiprogramming is a method of executing multiple processes simultaneously in the memory in order to minimize the average waiting time and efficiently use the available CPU resources. There are various CPU scheduling algorithms used to perform multiprogramming. In this program, we use C to simulate the following algorithms:

- First Come First Serve (FCFS)
- Shortest Job First (SJF)
- Non-Preemptive Priority
- Round Robin (RR)
- Shortest Remaining Time (SRT)
- Preemptive Priority

In this document we will go over how to install, build and execute the program. Finally, we will provide you with an example input/output of a successful run.

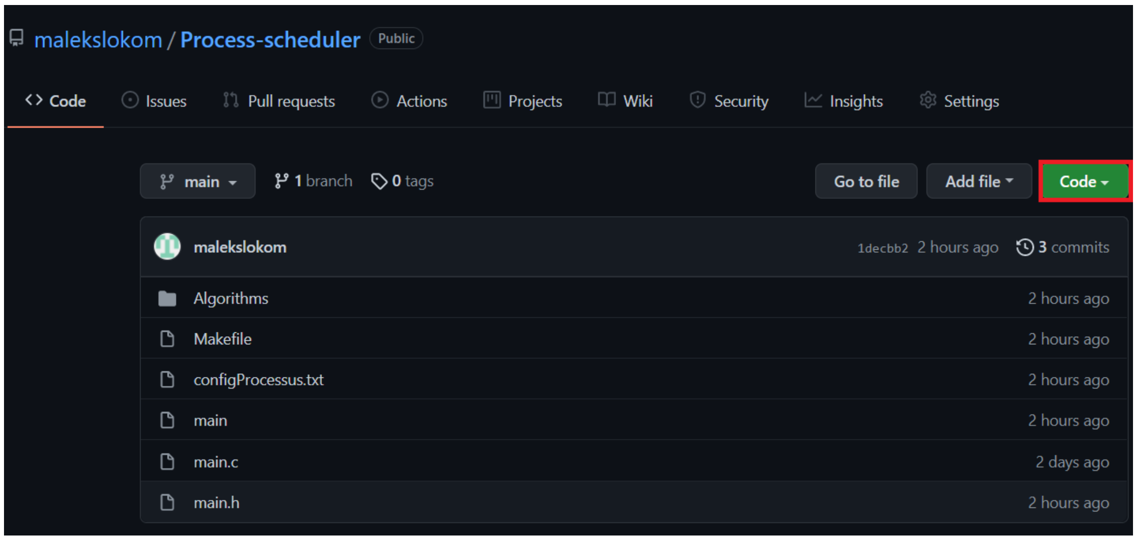
2 Requirements

The following prerequisites and requirements must be satisfied in order for the program to run successfully :

- Have a **Linux-based OS** (`https://ubuntu.com/download/desktop`)
- Install a **C language compiler** like gcc (`https://gcc.gnu.org/install/download.html`)
- (optional) Install the **Git** command line interpreter (`https://git-scm.com/downloads`)

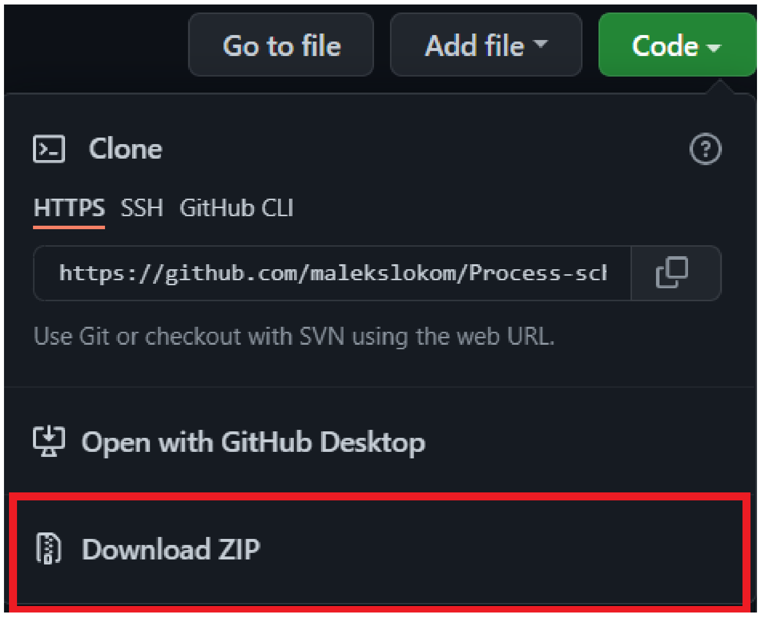
3 Installation

To install our project, first of all, you need to visit our *Github* repository at :
https://github.com/malekslokom/Process-scheduler

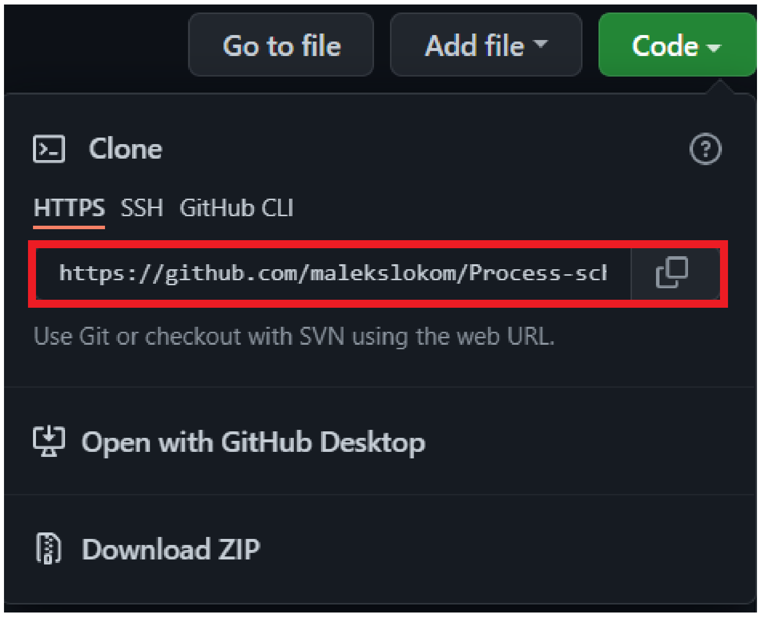


Then, you have two ways to download the project:

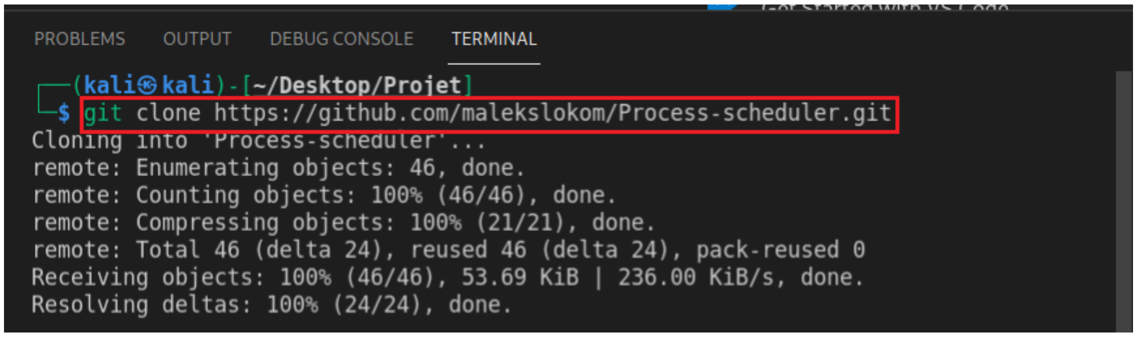
- **Download the project as a ZIP file**:
  To download the project as a ZIP file, while on the project page, click on the button "*Code*", then click on "*Download ZIP*".


Go to file
Add file
Code
Clone
HTTPS
SSH
GitHub CLI
https://github.com/malekslokom/Process-scl
Use Git or checkout with SVN using the web URL.
Open with GitHub Desktop
Download ZIP

- **Clone the project using Git**:
  To clone the project using Git, while on the project page, click on the button "*Code*" then copy the link you see in the figure below.


Go to file
Add file
Code
Clone
HTTPS
SSH
GitHub CLI
https://github.com/malekslokom/Process-scl
Use Git or checkout with SVN using the web URL.
Open with GitHub Desktop
Download ZIP

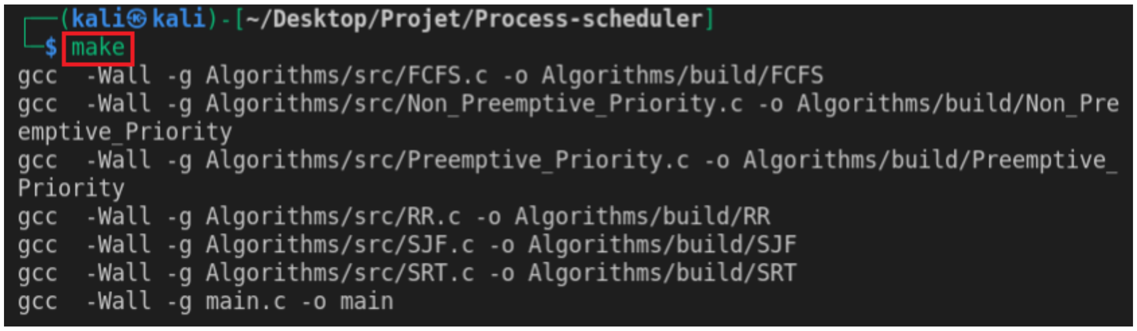
In your terminal, navigate to a folder where you want to download the project and run this command.



```
┌──(kali㉿kali)-[~/Desktop/Projet]
└─$ git clone https://github.com/malekslokom/Process-scheduler.git
Cloning into 'Process-scheduler'...
remote: Enumerating objects: 46, done.
remote: Counting objects: 100% (46/46), done.
remote: Compressing objects: 100% (21/21), done.
remote: Total 46 (delta 24), reused 46 (delta 24), pack-reused 0
Receiving objects: 100% (46/46), 53.69 KiB | 236.00 KiB/s, done.
Resolving deltas: 100% (24/24), done.
```

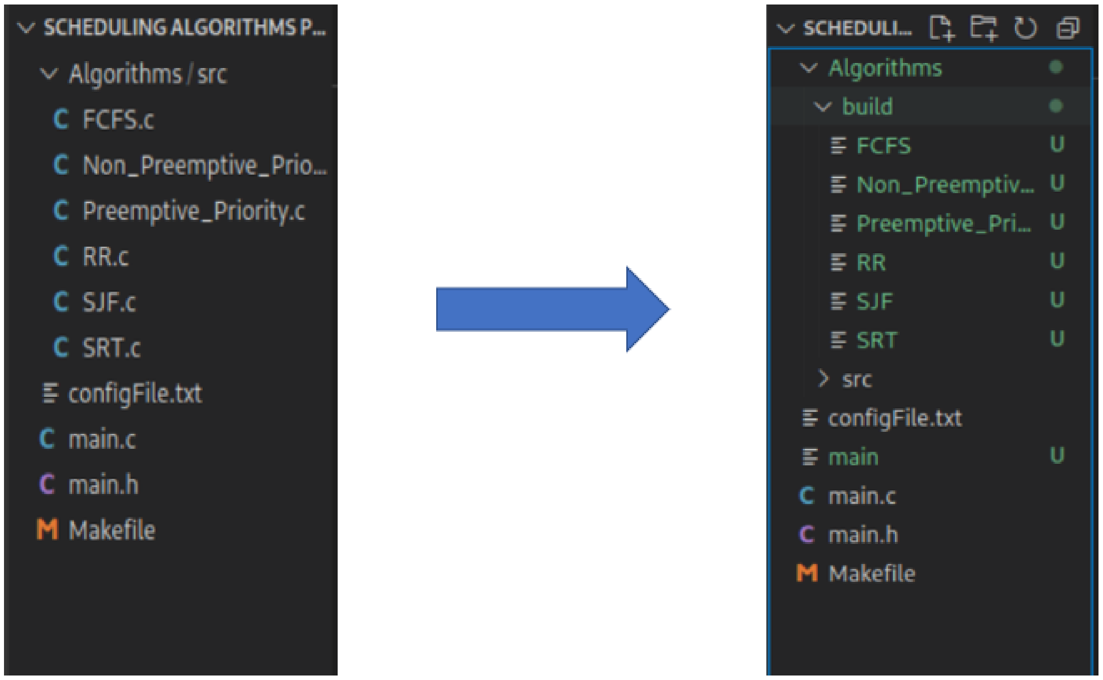
After downloading the project, you need **to build** it. In your terminal, while in the directory of your project, type the following commands:

- "***make***"

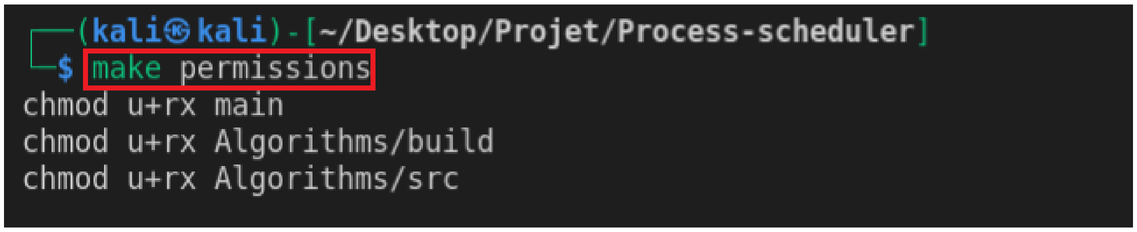


```
┌──(kali㉿kali)-[~/Desktop/Projet/Process-scheduler]
└─$ make
gcc  -Wall -g Algorithms/src/FCFS.c -o Algorithms/build/FCFS
gcc  -Wall -g Algorithms/src/Non_Preemptive_Priority.c -o Algorithms/build/Non_Pre
emptive_Priority
gcc  -Wall -g Algorithms/src/Preemptive_Priority.c -o Algorithms/build/Preemptive_
Priority
gcc  -Wall -g Algorithms/src/RR.c -o Algorithms/build/RR
gcc  -Wall -g Algorithms/src/SJF.c -o Algorithms/build/SJF
gcc  -Wall -g Algorithms/src/SRT.c -o Algorithms/build/SRT
gcc  -Wall -g main.c -o main
```

This will build the files in the "Algorithms/src" directory into executable files in the "Algorithms/build" directory so that your computer can utilize the different algorithms required. It will also generate a "main" file that you can execute to run the program.



- "*make permissions*"



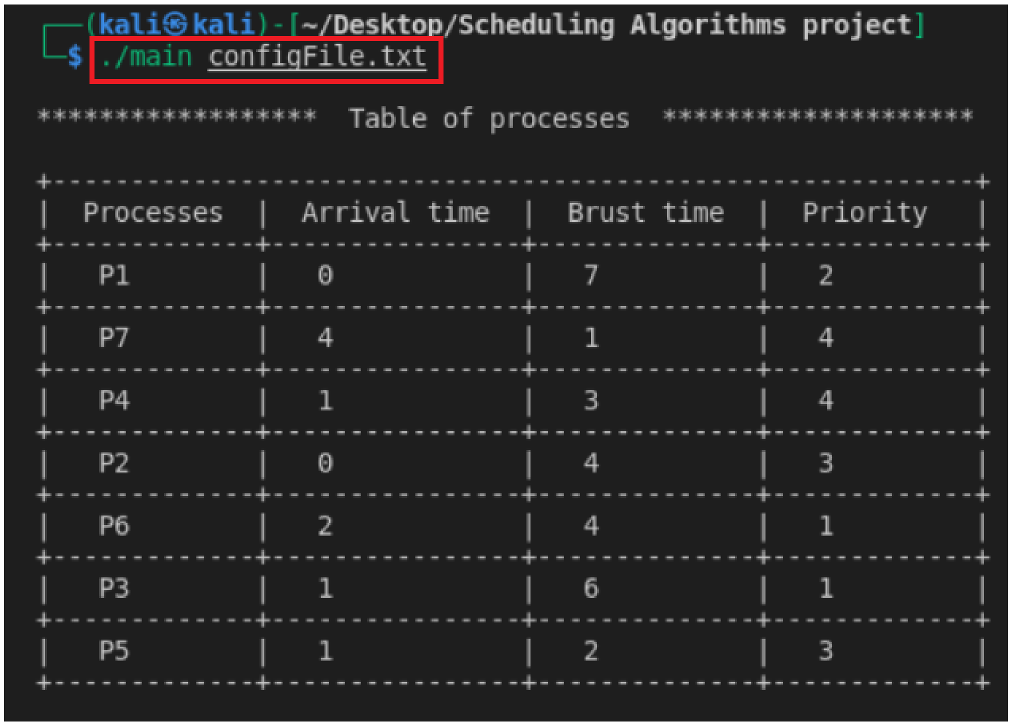
```
┌──(kali㉿kali)-[~/Desktop/Projet/Process-scheduler]
└─$ make permissions
chmod u+rx main
chmod u+rx Algorithms/build
chmod u+rx Algorithms/src
```

This command will allow a normal user to **read and run** the program, even if he doesn't have administrator privileges.

4 Running the project

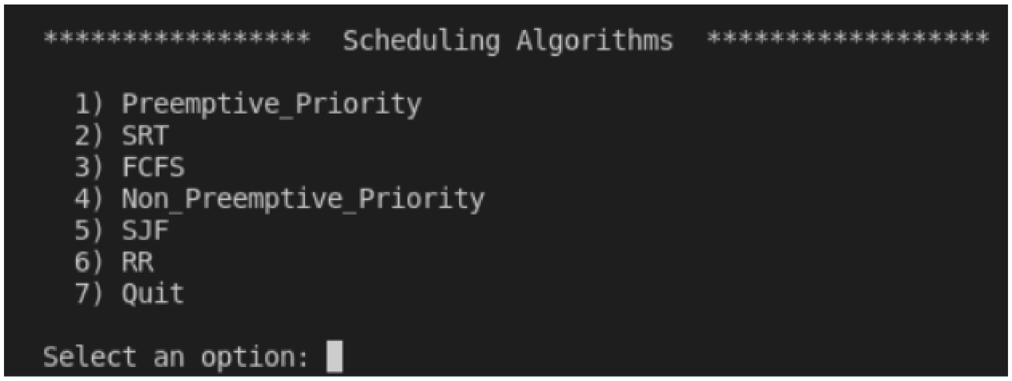
First of all, to run the project, you need to create a *configuration file* containing all the processes. To start, you can use the "configFile.txt" file we provide as an example.
Using your terminal, follow these steps:

- Type the command "***./main configFile.txt***"



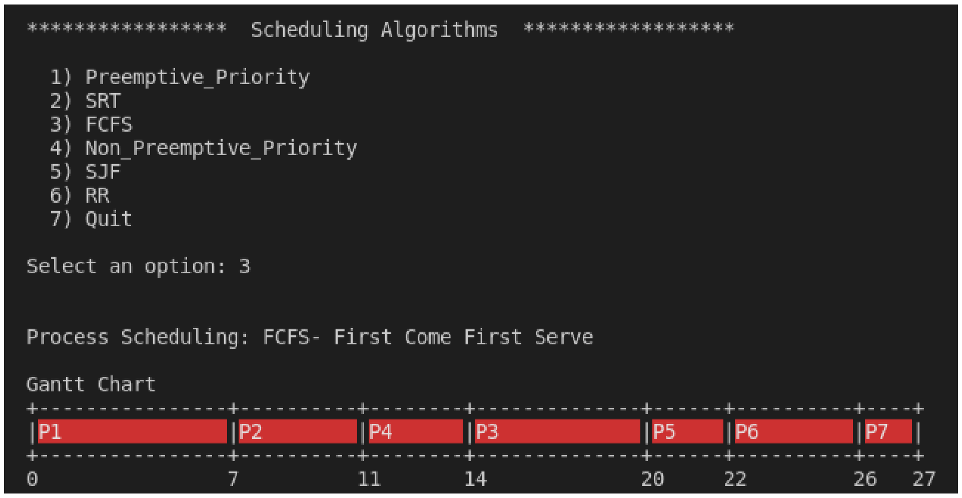
```
┌──(kali㉿kali)-[~/Desktop/Scheduling Algorithms project]
└─$ ./main configFile.txt

******************  Table of processes  ********************

+-------------+----------------+-------------+-------------+
|  Processes  |  Arrival time  |  Brust time |  Priority   |
+-------------+----------------+-------------+-------------+
|   P1        |   0            |   7         |   2         |
+-------------+----------------+-------------+-------------+
|   P7        |   4            |   1         |   4         |
+-------------+----------------+-------------+-------------+
|   P4        |   1            |   3         |   4         |
+-------------+----------------+-------------+-------------+
|   P2        |   0            |   4         |   3         |
+-------------+----------------+-------------+-------------+
|   P6        |   2            |   4         |   1         |
+-------------+----------------+-------------+-------------+
|   P3        |   1            |   6         |   1         |
+-------------+----------------+-------------+-------------+
|   P5        |   1            |   2         |   3         |
+-------------+----------------+-------------+-------------+
```

This command will run the program and use the configuration file at the path provided as input for the processes. This will display the list of processes in a table, then will allow you to choose one of several scheduling algorithm options generated dynamically using the filenames in the "*Algorithms/src*" directory.



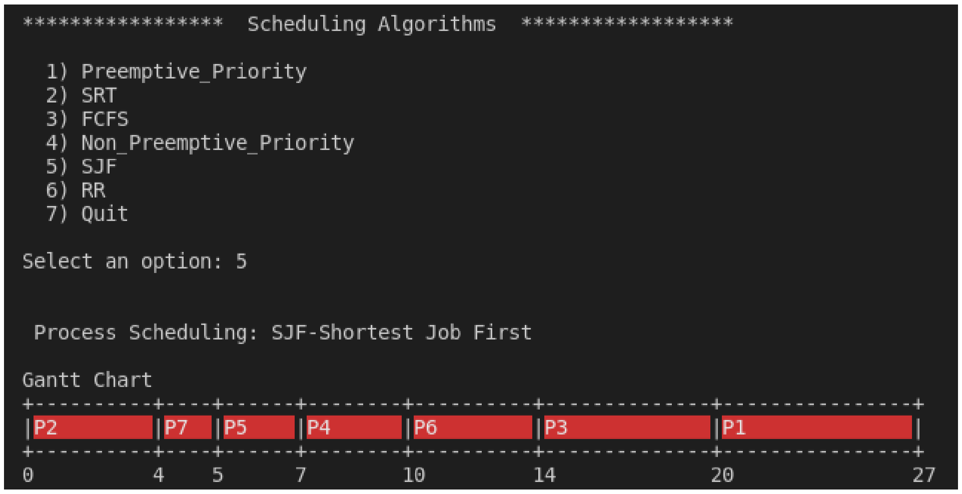
```
*****************  Scheduling Algorithms  ******************

  1) Preemptive_Priority
  2) SRT
  3) FCFS
  4) Non_Preemptive_Priority
  5) SJF
  6) RR
  7) Quit

Select an option: █
```

- Pick an option from the menu:
  - **First Come First Serve (FCFS)**:



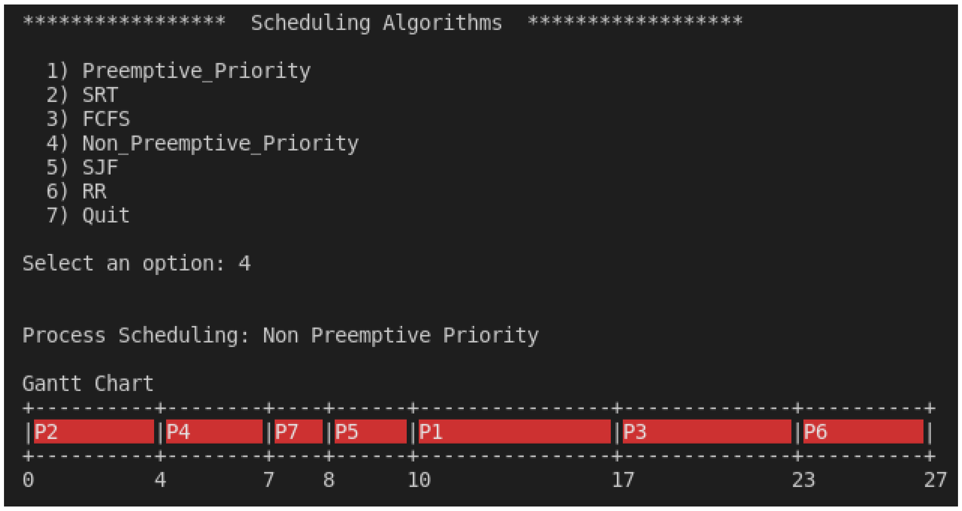
```
*****************  Scheduling Algorithms  ******************

  1) Preemptive_Priority
  2) SRT
  3) FCFS
  4) Non_Preemptive_Priority
  5) SJF
  6) RR
  7) Quit

Select an option: 3


Process Scheduling: FCFS- First Come First Serve

Gantt Chart
+----------------+----------+--------+--------------+------+----------+----+
|P1              |P2        |P4      |P3            |P5    |P6        |P7  |
+----------------+----------+--------+--------------+------+----------+----+
0                7          11       14             20     22         26   27
```

  - **Shortest Job First (SJF)**:



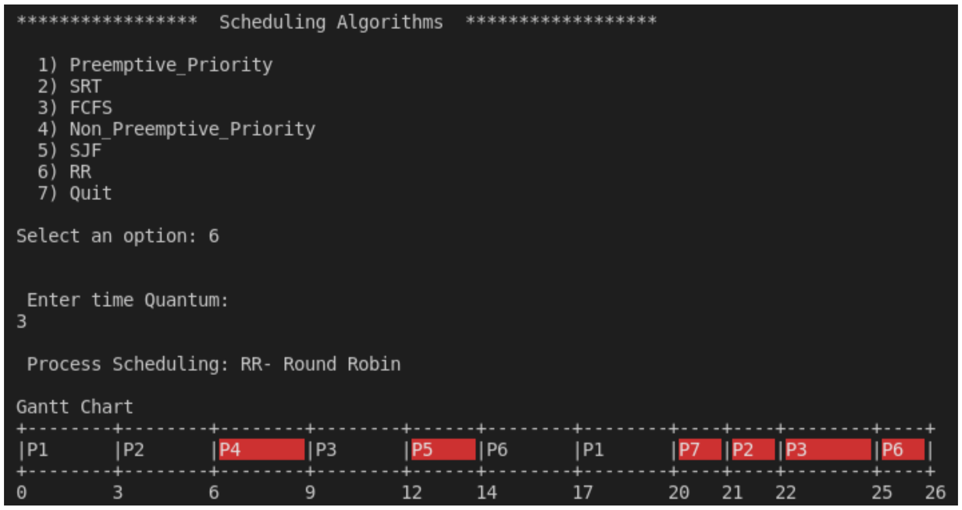
```
*****************  Scheduling Algorithms  ******************

  1) Preemptive_Priority
  2) SRT
  3) FCFS
  4) Non_Preemptive_Priority
  5) SJF
  6) RR
  7) Quit

Select an option: 5


 Process Scheduling: SJF-Shortest Job First

Gantt Chart
+----------+----+------+--------+----------+--------------+----------------+
|P2        |P7  |P5    |P4      |P6        |P3            |P1              |
+----------+----+------+--------+----------+--------------+----------------+
0          4    5      7        10         14             20               27
```

  - **Non-Preemptive Priority**:



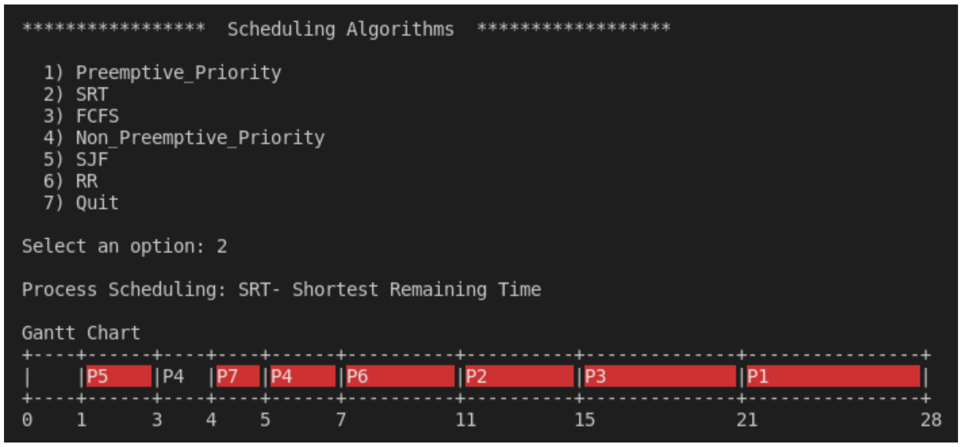
```
*****************  Scheduling Algorithms  ******************

  1) Preemptive_Priority
  2) SRT
  3) FCFS
  4) Non_Preemptive_Priority
  5) SJF
  6) RR
  7) Quit

Select an option: 4


Process Scheduling: Non Preemptive Priority

Gantt Chart
+----------+--------+----+------+----------------+--------------+----------+
|P2        |P4      |P7  |P5    |P1              |P3            |P6        |
+----------+--------+----+------+----------------+--------------+----------+
0          4        7    8      10               17             23         27
```

* **Round Robin (RR):**



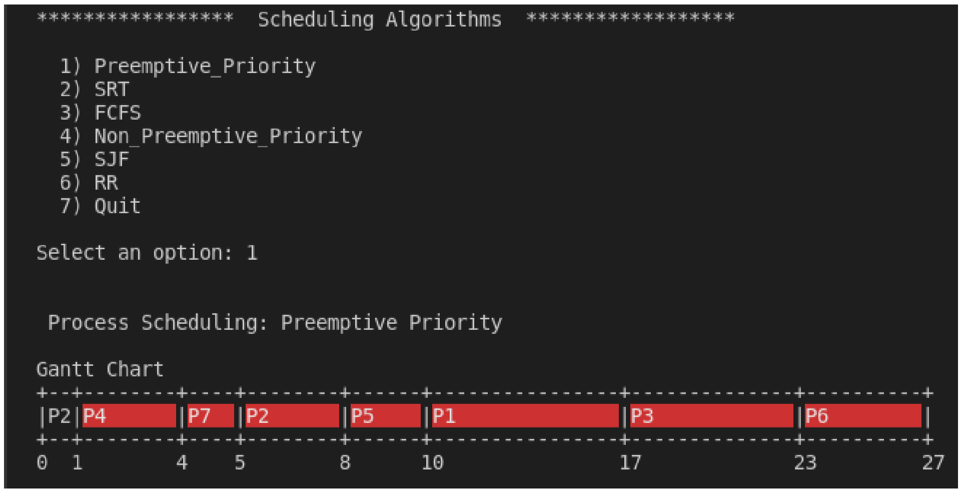
```
*****************  Scheduling Algorithms  ******************

  1) Preemptive_Priority
  2) SRT
  3) FCFS
  4) Non_Preemptive_Priority
  5) SJF
  6) RR
  7) Quit

Select an option: 6


 Enter time Quantum:
3

 Process Scheduling: RR- Round Robin

Gantt Chart
+--------+--------+--------+--------+------+--------+--------+----+----+--------+----+
|P1      |P2      |P4      |P3      |P5    |P6      |P1      |P7  |P2  |P3      |P6  |
+--------+--------+--------+--------+------+--------+--------+----+----+--------+----+
0        3        6        9        12     14       17       20   21   22       25   26
```

* **Shortest Remaining Time (SRT):**



```
*****************  Scheduling Algorithms  ******************

  1) Preemptive_Priority
  2) SRT
  3) FCFS
  4) Non_Preemptive_Priority
  5) SJF
  6) RR
  7) Quit

Select an option: 2

Process Scheduling: SRT- Shortest Remaining Time

Gantt Chart
+----+------+----+----+------+----------+----------+--------------+----------------+
|    |P5    |P4  |P7  |P4    |P6        |P2        |P3            |P1              |
+----+------+----+----+------+----------+----------+--------------+----------------+
0    1      3    4    5      7          11         15             21               28
```

* **Preemptive Priority:**



```
*****************  Scheduling Algorithms  ******************

  1) Preemptive_Priority
  2) SRT
  3) FCFS
  4) Non_Preemptive_Priority
  5) SJF
  6) RR
  7) Quit

Select an option: 1


 Process Scheduling: Preemptive Priority

Gantt Chart
+--+--------+----+--------+------+----------------+--------------+----------+
|P2|P4      |P7  |P2      |P5    |P1              |P3            |P6        |
+--+--------+----+--------+------+----------------+--------------+----------+
0  1        4    5        8      10               17             23         27
```