République Tunisienne

Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique

Université de Tunis El Manar

Institut Supérieur d'Informatique d'El Manar

Tunis El Manar University
HIGH INSTITUTE OF COMPUTER SCIENCE
SE Project

Process Scheduler

Project Analysis

Performed by:
Tayssir Baatour
Amani salah
Mokhtar Sallami
Mariem Najar

Supervised by:
M. Yousra Najar

12 December 2023

Contents

Liste des figures

1 Introduction

Our project focuses on developing a multitasking algorithm simulation system using the C programming language. The primary objective is to create a flexible and extensible environment for studying and comparing the performance of multitasking scheduling algorithms under a Linux operating system.

2 Project Environment

The development will take place in a Linux context, providing a robust and widely used platform for testing and implementing solutions related to process management.

3 Project Objectives

The project aims to achieve the following objectives clearly and definitively:

- **Efficient Simulation:**

  Design a high-performance simulation environment capable of generating random tasks.

- **Algorithm Evaluation:**

  Implement multiple multitasking scheduling algorithms to assess their performance.

- **Flexibility and Extensibility:**

  Provide a flexible and extensible simulation tool for studying and comparing the performance of multitasking scheduling algorithms.

- **Random Task Generation:**

  Generate random tasks with diverse characteristics such as execution time, priorities, and arrival times.

- **Algorithm Evaluation:**

  Allow the configuration of simulation parameters, including the number of tasks, arrival intervals, scheduling parameters, etc.

- **Data Collection and Visualization:**

  Collect data on task execution and algorithm performance metrics. Visualize simulation results in the form of graphs and statistics.

- **Implementation of Scheduling Policies:**

  Implement scheduling policies such as FIFO, SRT, Round-robin, Priority, and Multi-Level.

- **Dynamic Support for Scheduling Policies:**

  Enable the dynamic selection of scheduling policies from a menu, with dynamic consultation of the appropriate directory.

- **Graphical Display of Results:**

  Develop a graphical interface for real-time simulation tracking.

- **Dynamic Loading of Functions:**

  Implement dynamic loading of functions, with source code placed in the directory dedicated to scheduling policies.

- **Calculation of Comparative Metrics:**

  Perform calculations of metrics seen in class to compare performance between different algorithms.

  These objectives guide the project's development and serve as a framework for assessing its successful completion.

4 Roles in Scrum

- **Product Owner**

  - The Product Owner is responsible for defining the product vision, maintaining the product backlog, and prioritizing features based on business value.
  - In our project, the Product Owner will be the key stakeholder or a representative who provides insights into the simulation tool's functionalities and requirements. They will guide the development team on what features are crucial for achieving project goals.

- **Scrum Master**

  - The Scrum Master is the facilitator of the Scrum process, ensuring that the Scrum team adheres to Scrum principles and values.
  - In our project, the Scrum Master will be a guide and mentor to the development team, helping them understand and implement Scrum practices. They will also be responsible for removing impediments and fostering a collaborative and productive environment.

- **Development Team**

  - The Development Team is a self-organizing, cross-functional group responsible for delivering a potentially shippable product at the end of each sprint.
  - In our project, the Development Team comprises individuals with expertise in software development, simulation, and algorithm implementation. They collaborate to implement the features defined in the product backlog during each sprint.

- **Collaboration**

  - Regular communication and collaboration among these roles are vital for the success of the project. The Product Owner's insights guide the team's priorities, the Scrum Master facilitates the Scrum process, and the Development Team brings the vision to life through code and implementation.

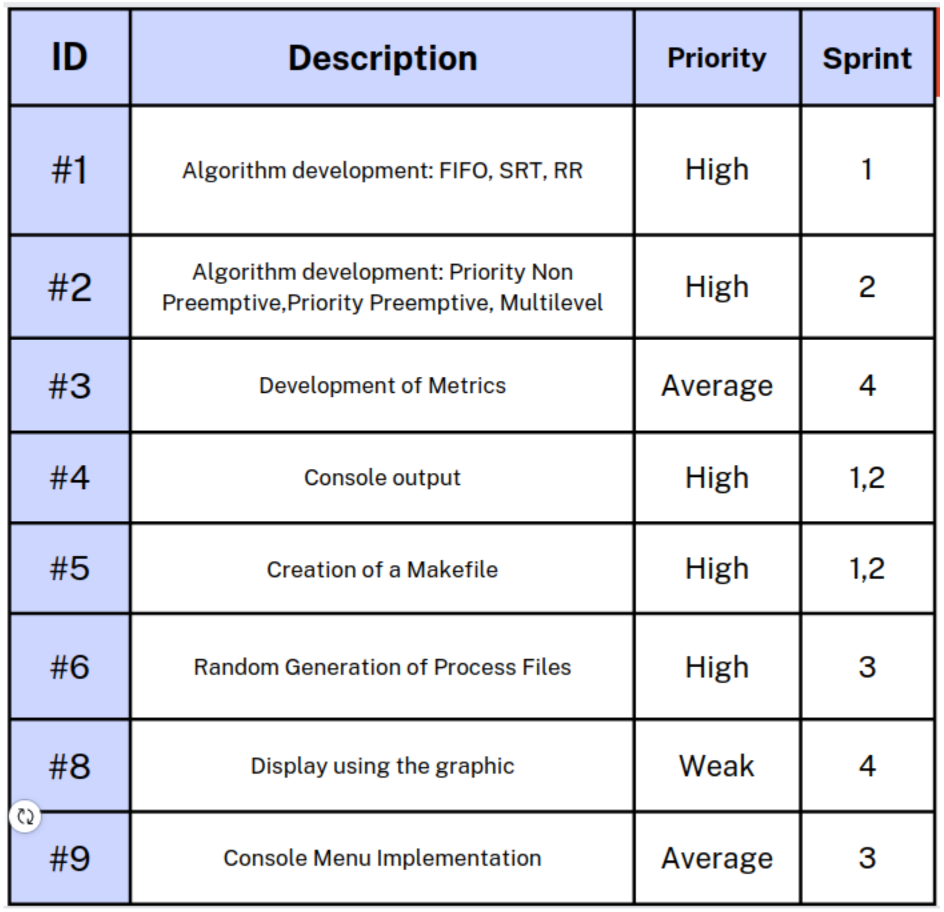
5 Project Planning with Scrum

5.1 Product Backlog

The product backlog represents the comprehensive list of features and tasks to be developed throughout the project. For the multitasking scheduling algorithm simulation system, the backlog could include the following items:

1. Algorithm development: FIFO, SRT, RR
2. Algorithm development: Priority Non Preemptive,Priority Preemptive, Multilevel
3. Creation of a Makefile
4. Development of Metrics
5. Console output
6. Random Generation of Process Files
7. Display using the graphic.

8. Console Menu Implementation

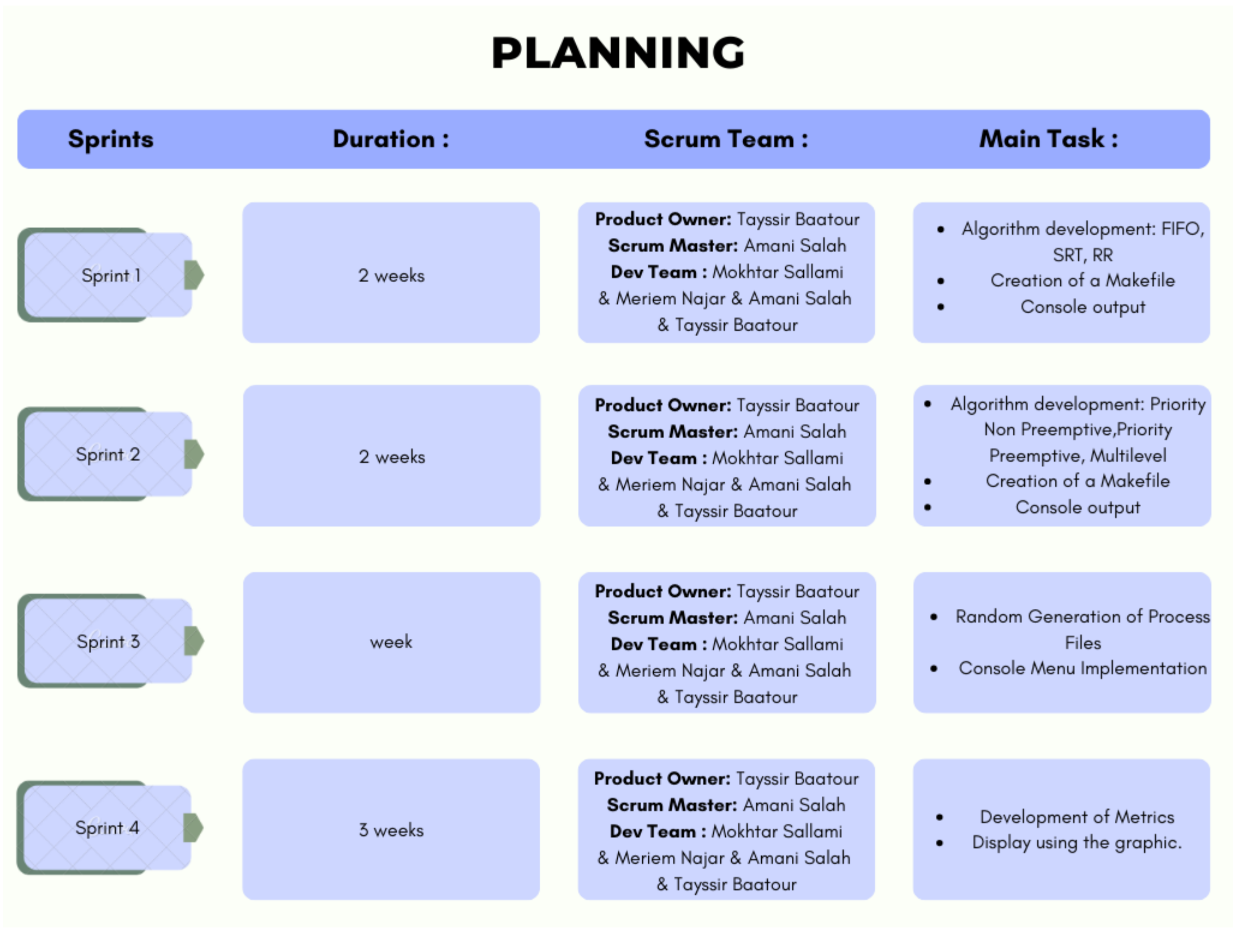


ID	Description	Priority	Sprint
#1	Algorithm development: FIFO, SRT, RR	High	1
#2	Algorithm development: Priority Non Preemptive,Priority Preemptive, Multilevel	High	2
#3	Development of Metrics	Average	4
#4	Console output	High	1,2
#5	Creation of a Makefile	High	1,2
#6	Random Generation of Process Files	High	3
#8	Display using the graphic	Weak	4
#9	Console Menu Implementation	Average	3

Figure 1: Product Backlog

5.2 Sprint Planning

Sprint planning involves breaking down the project into development iterations, known as sprints, each with a defined duration. For this project, an indicative sprint plan might look like this:

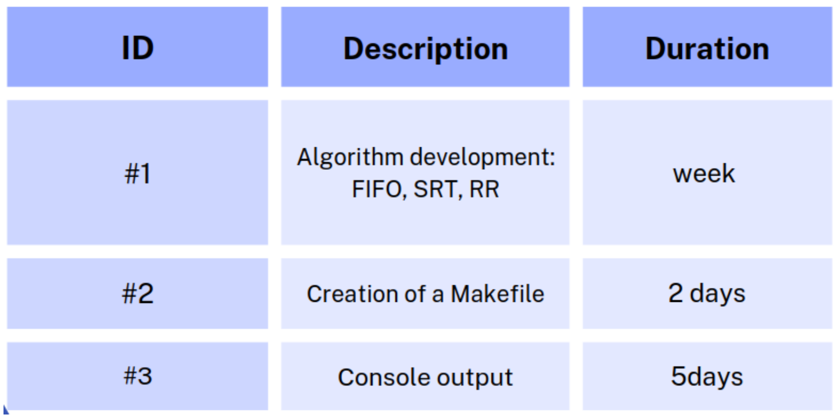


PLANNING

Sprints	Duration :	Scrum Team :	Main Task :
Sprint 1	2 weeks	**Product Owner:** Tayssir Baatour **Scrum Master:** Amani Salah **Dev Team :** Mokhtar Sallami & Meriem Najar & Amani Salah & Tayssir Baatour	• Algorithm development: FIFO, SRT, RR • Creation of a Makefile • Console output
Sprint 2	2 weeks	**Product Owner:** Tayssir Baatour **Scrum Master:** Amani Salah **Dev Team :** Mokhtar Sallami & Meriem Najar & Amani Salah & Tayssir Baatour	• Algorithm development: Priority Non Preemptive,Priority Preemptive, Multilevel • Creation of a Makefile • Console output
Sprint 3	week	**Product Owner:** Tayssir Baatour **Scrum Master:** Amani Salah **Dev Team :** Mokhtar Sallami & Meriem Najar & Amani Salah & Tayssir Baatour	• Random Generation of Process Files • Console Menu Implementation
Sprint 4	3 weeks	**Product Owner:** Tayssir Baatour **Scrum Master:** Amani Salah **Dev Team :** Mokhtar Sallami & Meriem Najar & Amani Salah & Tayssir Baatour	• Development of Metrics • Display using the graphic.

Figure 2: Sprint planning

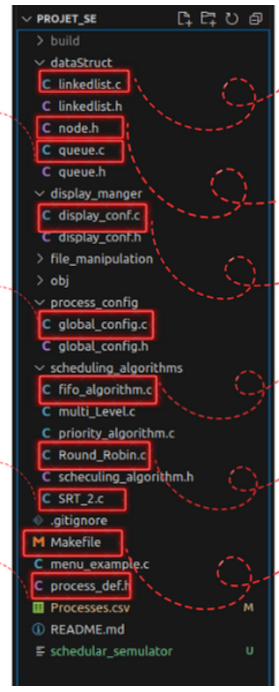
5.2.1 Sprint 1 :



ID	Description	Duration
#1	Algorithm development: FIFO, SRT, RR	week
#2	Creation of a Makefile	2 days
#3	Console output	5days

Figure 3: Sprint 1

– **Sprint Review :**



The code is designed to manage a queue of processes in scheduling or simulation scenarios.
- It includes functions for initializing, creating, enqueuing, and dequeuing a queue, along with additional sorting and utility functions.

The linkedlist.c file provides management of linked lists in the project:
- It creates and initializes a linked list of processes.
- It performs operations such as inserting a node at the head of the list and printing the elements of the list.
- It offers sorting functions (linkedlist_bubbleSort and linkedlist_bubbleSortpriority) to sort the list based on the waiting time ("ta") and the priority of processes, respectively.

The node.h file is used to define the structure of a node in a linked list.

le fichier display_conf.c est essentiel pour gérer et présenter les aspects visuels des processus, leur progression et les détails associés dans un contexte de simulation ou de planification.

The global_conf.c file houses functions that collectively contribute to the generation of random process attributes, including execution time, arrival time, priority, and color. Additionally, it facilitates the calculation of the total simulation time based on the linked list of processes.

The fifo_algorithm.c file contains the implementation of the First-In-First-Out (FIFO) scheduling algorithm.

The srt_algorithm.c file serves the purpose of implementing the Shortest Remaining Time (SRT) scheduling algorithm.

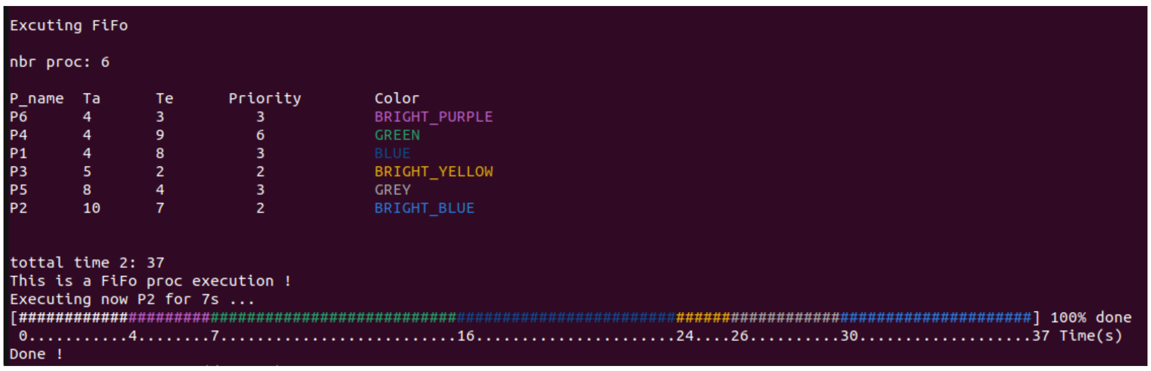
The round_robin.c file serves the purpose of implementing the Round Robin scheduling algorithm.

The process_def.c file is used to define and configure parameters and characteristics related to processes for a simulation or scheduling program.

The makefile provides a set of rules and dependencies for compiling and linking various source files to create multiple scheduling algorithm executables. The targets include the main scheduler simulator, individual algorithms (FIFO, Round Robin, SRT, Priority, Multilevel, and Priority Preemptive), and additional test targets (tfifo, tRR, tSRT, tPRIORITY, tMulti, and tPP).

– **Console Interface :**

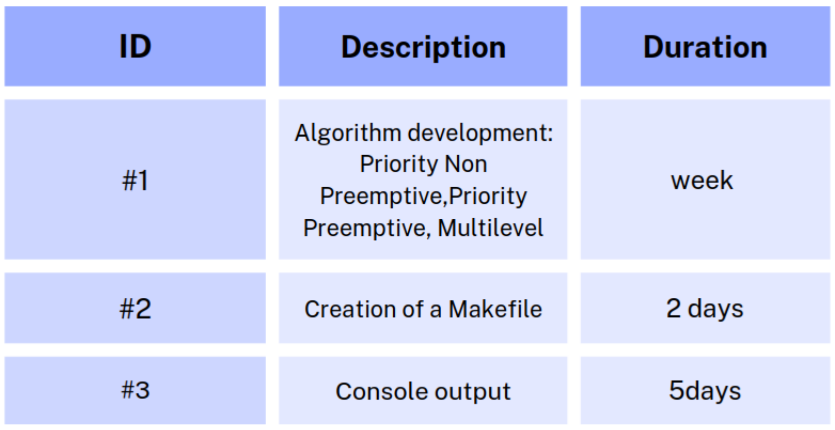
We have developed three algorithms(FIFO, SRT, RR) with their console interfaces, and here is an example of the execution of the FIFO algorithm.



```
Excuting FiFo

nbr proc: 6

P_name  Ta      Te      Priority        Color
P6      4       3          3            BRIGHT_PURPLE
P4      4       9          6            GREEN
P1      4       8          3            BLUE
P3      5       2          2            BRIGHT_YELLOW
P5      8       4          3            GREY
P2      10      7          2            BRIGHT_BLUE


tottal time 2: 37
This is a FiFo proc execution !
Executing now P2 for 7s ...
[###############################################################################################################] 100% done
 0...........4........7..........................16......................24....26..........30...................37 Time(s)
Done !
```

Figure 4: Execution of the FIFO algorithm in the console

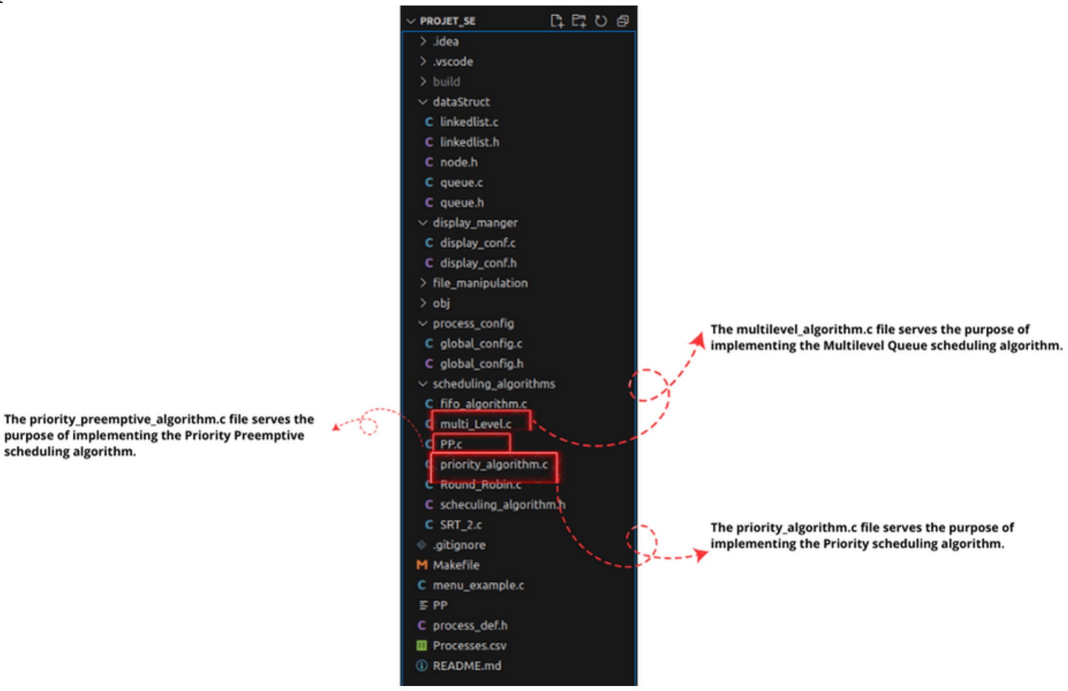
5.2.2 Sprint 2 :



ID	Description	Duration
#1	Algorithm development: Priority Non Preemptive,Priority Preemptive, Multilevel	week
#2	Creation of a Makefile	2 days
#3	Console output	5days

Figure 5: Sprint 2

– **Sprint Review :**



```
PROJET_SE
  > .idea
  > .vscode
  > build
  v dataStruct
    C linkedlist.c
    C linkedlist.h
    C node.h
    C queue.c
    C queue.h
  v display_manger
    C display_conf.c
    C display_conf.h
  > file_manipulation
  > obj
  v process_config
    C global_config.c
    C global_config.h
  v scheduling_algorithms
    C fifo_algorithm.c
    C multi_Level.c
    C PP.c
    C priority_algorithm.c
    C Round_Robin.c
    C scheculing_algorithm.h
    C SRT_2.c
  .gitignore
  M Makefile
  C menu_example.c
  PP
  C process_def.h
  Processes.csv
  README.md
```

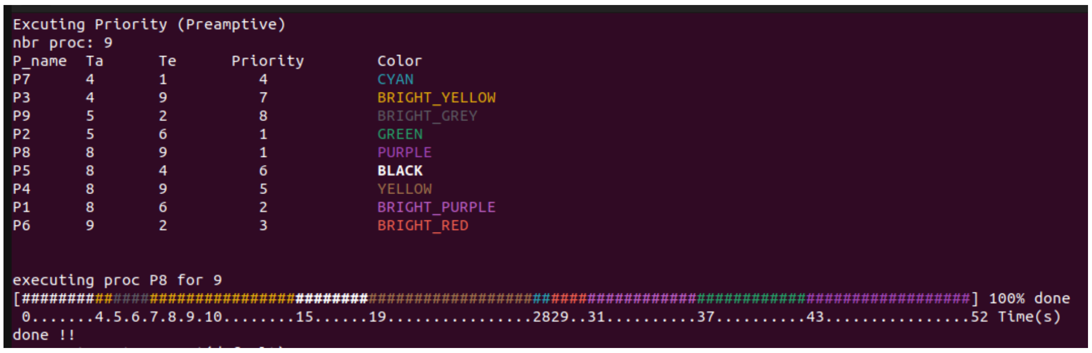
The multilevel_algorithm.c file serves the purpose of implementing the Multilevel Queue scheduling algorithm.

The priority_preemptive_algorithm.c file serves the purpose of implementing the Priority Preemptive scheduling algorithm.

The priority_algorithm.c file serves the purpose of implementing the Priority scheduling algorithm.

– **Console Interface :**

We have developed three algorithms(Priority Non Preemptive,Priority Preemptive, Multilevel) with their console interfaces, and here is an example of the execution of the Priority Preemptive algorithm.



```
Excuting Priority (Preamptive)
nbr proc: 9
P_name  Ta      Te      Priority        Color
P7      4       1          4            CYAN
P3      4       9          7            BRIGHT_YELLOW
P9      5       2          8            BRIGHT_GREY
P2      5       6          1            GREEN
P8      8       9          1            PURPLE
P5      8       4          6            BLACK
P4      8       9          5            YELLOW
P1      8       6          2            BRIGHT_PURPLE
P6      9       2          3            BRIGHT_RED


executing proc P8 for 9
[#############################################################################################################] 100% done
 0.......4.5.6.7.8.9.10........15......19................2829..31..........37..........43................52 Time(s)
done !!
```

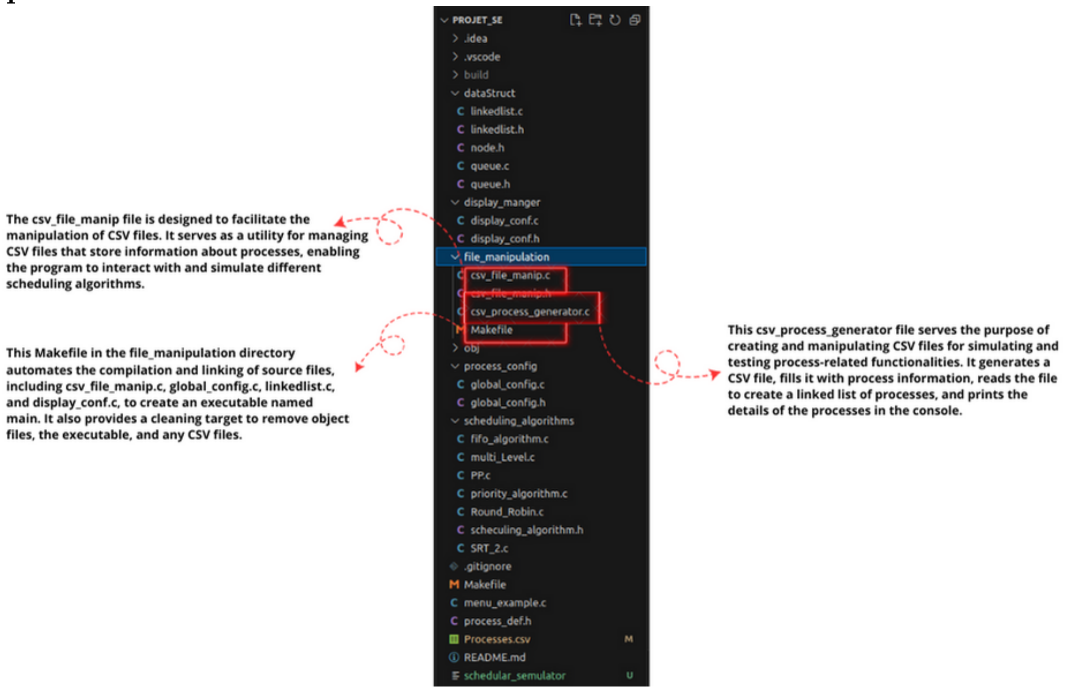
Figure 6: Execution of the Priority Preemptive algorithm in the console

5.2.3 Sprint 3 :

ID	Description	Duration
#1	Random Generation of Process Files	2 days
#2	Console Menu Implementation	5 days

Figure 7: Sprint 3

– **Sprint Review :**



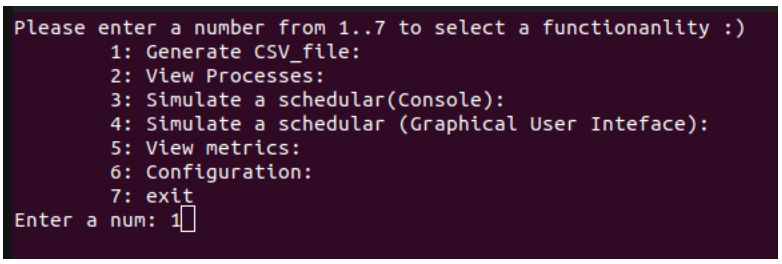
The csv_file_manip file is designed to facilitate the manipulation of CSV files. It serves as a utility for managing CSV files that store information about processes, enabling the program to interact with and simulate different scheduling algorithms.

This Makefile in the file_manipulation directory automates the compilation and linking of source files, including csv_file_manip.c, global_config.c, linkedlist.c, and display_conf.c, to create an executable named main. It also provides a cleaning target to remove object files, the executable, and any CSV files.

```
PROJET_SE
  > .idea
  > .vscode
  > build
  v dataStruct
    C linkedlist.c
    C linkedlist.h
    C node.h
    C queue.c
    C queue.h
  v display_manger
    C display_conf.c
    C display_conf.h
  v file_manipulation
    C csv_file_manip.c
    C csv_file_manip.h
    C csv_process_generator.c
    M Makefile
  > obj
  v process_config
    C global_config.c
    C global_config.h
  v scheduling_algorithms
    C fifo_algorithm.c
    C multi_Level.c
    C PP.c
    C priority_algorithm.c
    C Round_Robin.c
    C scheculing_algorithm.h
    C SRT_2.c
  .gitignore
  M Makefile
  C menu_example.c
  C process_def.h
  Processes.csv                M
  README.md
  schedular_semulator          U
```

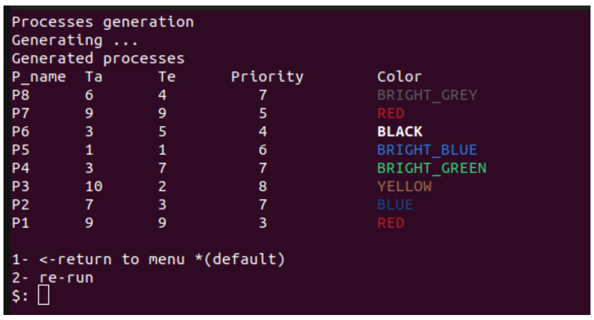
This csv_process_generator file serves the purpose of creating and manipulating CSV files for simulating and testing process-related functionalities. It generates a CSV file, fills it with process information, reads the file to create a linked list of processes, and prints the details of the processes in the console.

- **Console Interface :**
  The figure below shows the menu interface and the random generation of CSV files.



```
Please enter a number from 1..7 to select a functionanlity :)
        1: Generate CSV_file:
        2: View Processes:
        3: Simulate a schedular(Console):
        4: Simulate a schedular (Graphical User Inteface):
        5: View metrics:
        6: Configuration:
        7: exit
Enter a num: 1
```

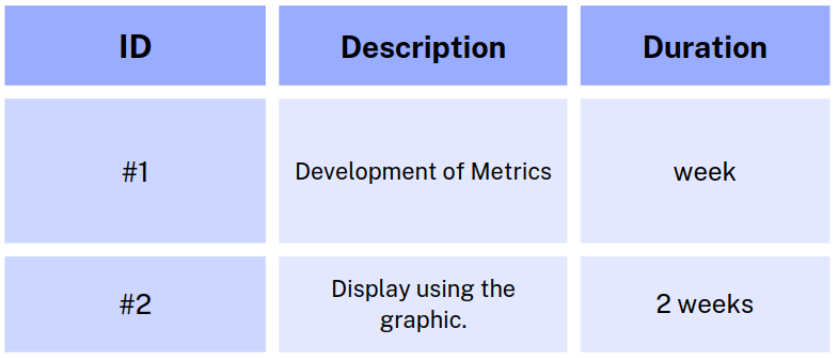
Figure 8: Menu intrface



```
Processes generation
Generating ...
Generated processes
P_name  Ta      Te      Priority        Color
P8      6       4          7            BRIGHT_GREY
P7      9       9          5            RED
P6      3       5          4            BLACK
P5      1       1          6            BRIGHT_BLUE
P4      3       7          7            BRIGHT_GREEN
P3      10      2          8            YELLOW
P2      7       3          7            BLUE
P1      9       9          3            RED

1- <-return to menu *(default)
2- re-run
$: 
```

Figure 9: Random generation of process Files

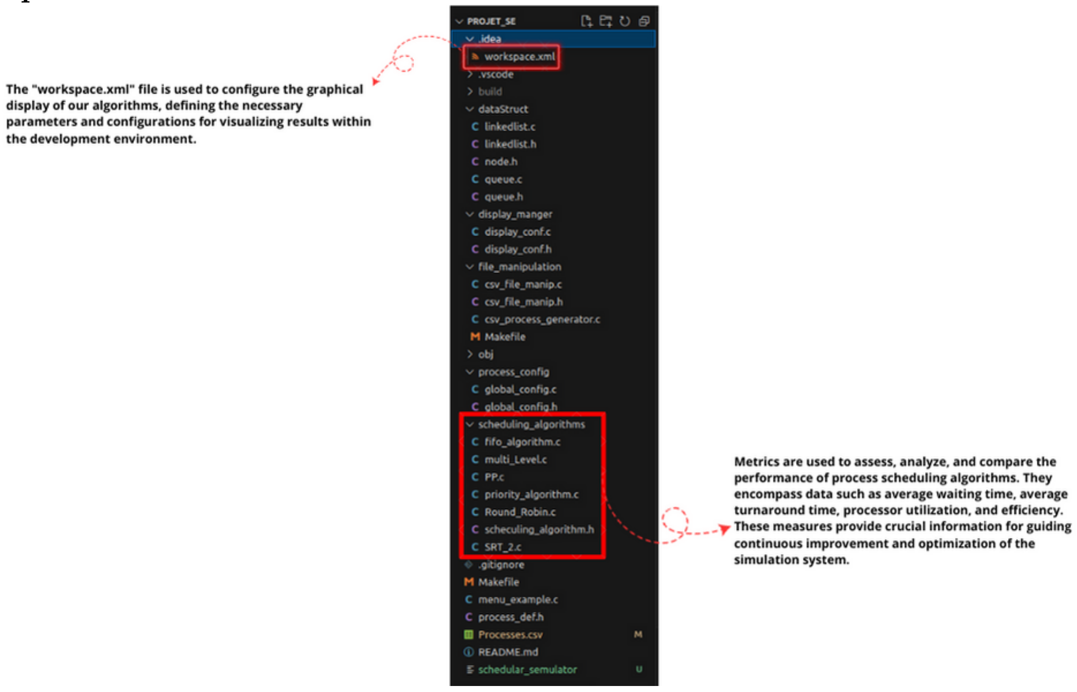
5.2.4 Sprint 4 :



ID	Description	Duration
#1	Development of Metrics	week
#2	Display using the graphic.	2 weeks

Figure 10: Sprint 2

– **Sprint Review :**



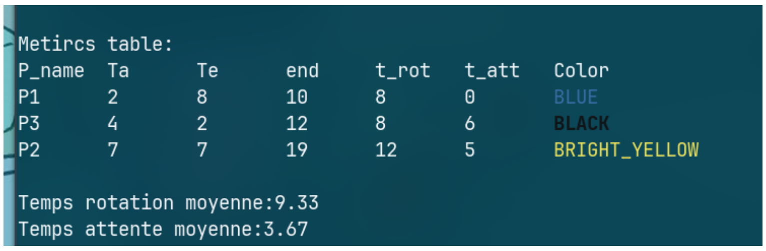
The "workspace.xml" file is used to configure the graphical display of our algorithms, defining the necessary parameters and configurations for visualizing results within the development environment.

```
PROJET_SE
  .idea
    workspace.xml
  .vscode
  build
  dataStruct
    linkedlist.c
    linkedlist.h
    node.h
    queue.c
    queue.h
  display_manger
    display_conf.c
    display_conf.h
  file_manipulation
    csv_file_manip.c
    csv_file_manip.h
    csv_process_generator.c
    Makefile
  obj
  process_config
    global_config.c
    global_config.h
  scheduling_algorithms
    fifo_algorithm.c
    multi_Level.c
    PP.c
    priority_algorithm.c
    Round_Robin.c
    scheculing_algorithm.h
    SRT_2.c
  .gitignore
  Makefile
  menu_example.c
  process_def.h
  Processes.csv                 M
  README.md
  schedular_semulator           U
```

Metrics are used to assess, analyze, and compare the performance of process scheduling algorithms. They encompass data such as average waiting time, average turnaround time, processor utilization, and efficiency. These measures provide crucial information for guiding continuous improvement and optimization of the simulation system.

– **Console Interface :**

In the figures below, we have the chart of an example of metrics and an example of the execution of an algorithm.



```
Metircs table:
P_name  Ta      Te      end     t_rot   t_att   Color
P1      2       8       10      8       0       BLUE
P3      4       2       12      8       6       BLACK
P2      7       7       19      12      5       BRIGHT_YELLOW

Temps rotation moyenne:9.33
Temps attente moyenne:3.67
```

Figure 11: Metric of the priority preemptive algorithm

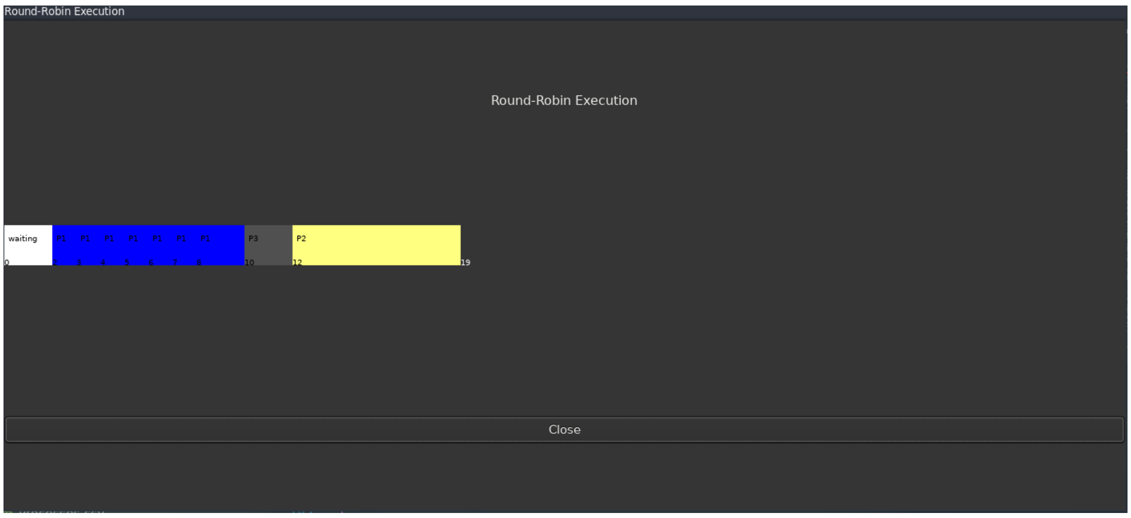


Figure 12: Graphic simulation of the priority preemptive algorithm

6 conclusion

In conclusion, our project has successfully achieved its primary objective of developing a multitasking algorithm simulation system using the C programming language. By providing a flexible and extensible environment, we have created a valuable platform for in-depth study and comparison of multitasking scheduling algorithms under the Linux operating system.