

Ordonnancement des processus

1 Introduction

Pour la réalisation de ce projet, nous avons tout d'abord installé notre environnement de travail. Pour ce faire, nous avons installé Ubuntu et le VMWARE WORKSTATION pour pouvoir installer Linux, et pour la rédaction du rapport nous avons utilisé l'éditeur LaTeX en ligne Overleaf. Ensuite, nous avons choisi les algorithmes avec lesquels on va travailler (FiFo, LiFo, RoundRobin, SJF et Priorité).

2 Description du projet

Tout d'abord, nous avons réalisé un dossier intitulé "Projet-Se" qui englobe un autre dossier nommé "Politiques" et d'autres fichiers citons parmi lesquels :

- Main.c : contient le menu principal du programme de gestion des processus
- Ajouterprocessus.c : contient le code c de chargement des processus dans le fichier de configuration
- Configuration.txt : comprend les différents processus, leurs temps d'arrivée, leurs temps d'exécution ainsi que leurs priorités
- MenuAlgo.c : contient le code c qui permet de récupérer du dossier Politiques la liste dynamique des algorithmes à exécuter

La figure 1 montre l'interface de notre projet

```
-----
| PROGRAMME DE GESTION DES PROCESSUS |
-----

| MENU PRINCIPAL |
| 1 . Chargement des données |
| 2 . Choix de l'algorithme |
| 3 . Quitter |
|-----|

Votre Choix : 2

| ALGORITHMES D'ODONNANCEMENT |
|-----|
| 1 . FIFO |
| 2 . LIFO |
| 3 . Priority |
| 4 . RoundRobin |
| 5 . SJF |
| 6 . Quitter |
|-----|

Votre Choix : █
```

Figure 1: Menu Principale

3 Description du jeu de processus

L'utilisateur peut charger ses données et ceci en d'définissant le nom de chaque processus, en saisissant sa date d'arrivée, son temps d'exécution ainsi que sa priorité. Ce dernier peut aussi introduire des commentaires et laisser des lignes blanches.

```
-----
| PROGRAMME DE GESTION DES PROCESSUS |
-----

| MENU PRINCIPAL |
| 1 . Chargement des données |
| 2 . Choix de l'algorithme |
| 3 . Quitter |
|-----|

Votre Choix : 1

Nombre des processus : 3
Entrer le temps d'arriver du processus : 0
Entrer le temps d'execution du processus : 4
Entrer la priorite du precessus : 2
Entrer le temps d'arriver du processus : 3
Entrer le temps d'execution du processus : 2
Entrer la priorite du precessus : 1
Entrer le temps d'arriver du processus : 2
Entrer le temps d'execution du processus : 4
Entrer la priorite du precessus : 4

Voulez vous continuer O/N ?
```

Figure 2: Chargement des processus

4 Description du choix de l'algorithme

Pour le choix de l'algorithme l'utilisateur a la possibilité d'ordonner ces processus selon une diversité d'algorithmes possibles :

- FiFo
- LiFo
- RoundRobin
- SJF
- Priorité

4.1 FiFo

En exécutant cet algorithme, l'utilisateur sera muni d'un tableau ordonné par la politique d'ordonnancement FIFO qui se base sur le temps d'arrivée : c'est à dire que le premier arrivé sera le premier à s'exécuter. De plus, nous lui offrant la possibilité de visualiser l'ordonnancement de ces processus `à chaque instant` à l'aide du diagramme de GANTT. En outre, il peut aussi continuer son exécution et ceci en choisissant une autre politique d'ordonnancement.

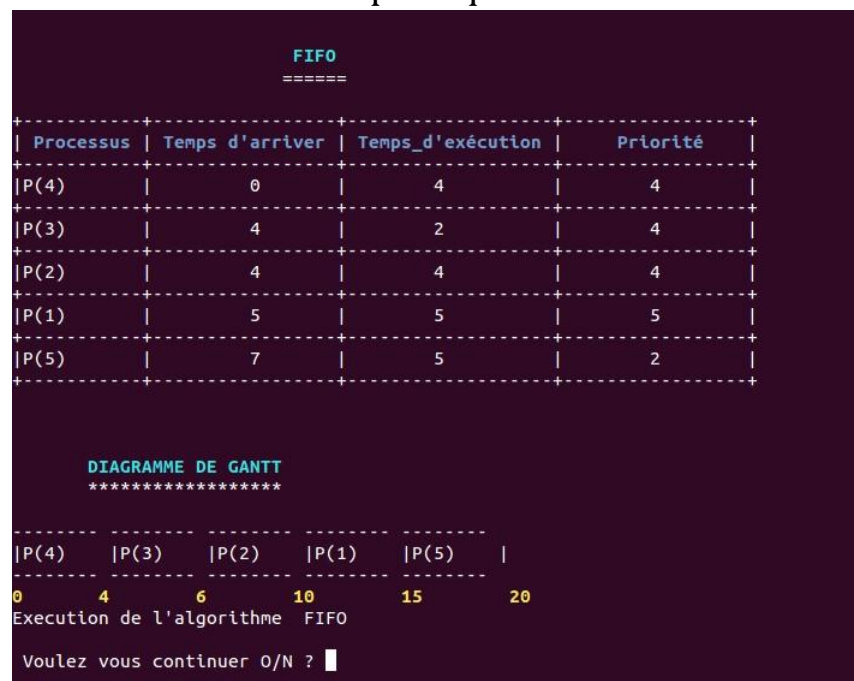


Figure 3: Exécution de l'algorithme FiFo

4.2 LiFo

En exécutant cet algorithme, l'utilisateur sera muni d'un tableau ordonné par la politique d'ordonnancement LIFO qui se base sur le temps d'arrivé : c'est à` dire que le dernier arrivé sera le premier à s'exécuter. De plus, nous lui offrant la possibilité de visualiser l'ordonnancement de ces processus a` chaque instant a` l'aide d'un diagramme de GANTT.

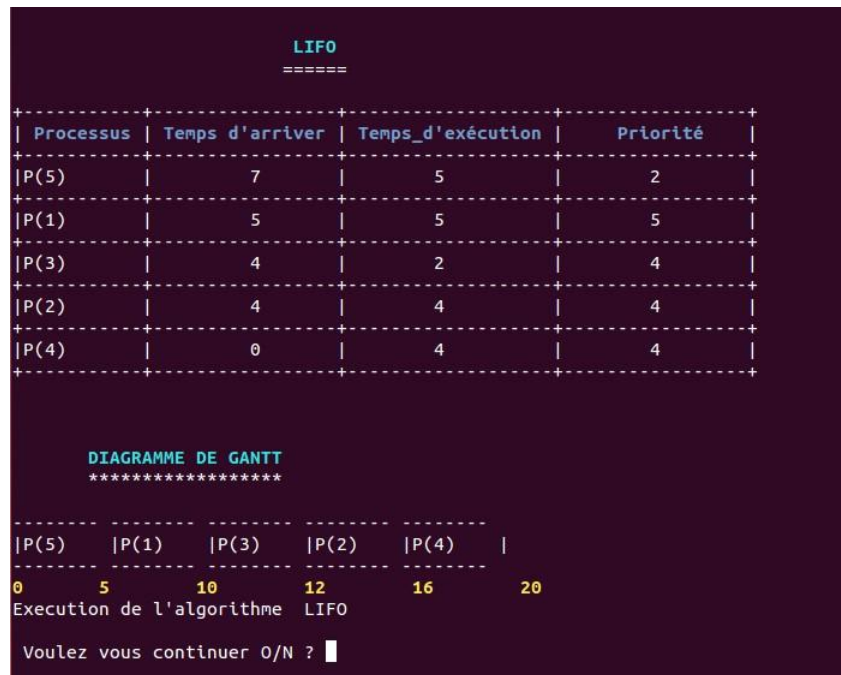


Figure 4: Exécution de l'algorithme LiFo

4.3 RoundRobin

En exécutant l'algorithme d'ordonnancement RoundRobin, l'utilisateur doit tout d'abord saisir le Quantum. Cet algorithme se base sur le temps d'arrivé : c'est à` dire que le premier arrivé sera le premier à s'exécuter et que le temps processeur sera divisé en intervalles de temps (Quantum).

De ce fait, chaque algorithme s'exécutera pendant son Quantum.

De plus, nous lui offrant la possibilité de visualiser l'ordonnancement de ces processus à chaque instant a` l'aide du diagramme de GANTT. En outre, il peut ainsi visualiser son Temps d'attente moyen ainsi que son Temps de rotation moyen.

```

                                ROUND ROBIN
                                =====
Entrez le Quantum : 2

+-----+-----+-----+-----+
| Processus | Temps d'arriver | Temps_d'exécution | Priorité |
+-----+-----+-----+-----+
| P(1)      | 0                | 4                  | 2        |
+-----+-----+-----+-----+
| P(3)      | 2                | 4                  | 4        |
+-----+-----+-----+-----+
| P(2)      | 3                | 2                  | 1        |
+-----+-----+-----+-----+

DIAGRAMME DE GANTT
*****
0 -> [P(1)] <- 2 -> [P(3)] <- 4 -> [P(2)] <- 6 -> [P(1)] <- 8 -> [P(3)] <- 10

Temps d'attente moyen : 3.00
Temps de rotation moyen : 6.33
Execution de l'algorithme RoundRobin

Voulez vous continuer O/N ? █

```

Figure 5: Exécution de l'algorithme RoundRobin

4.4 SJF

En exécutant cet algorithme, l'utilisateur sera muni d'un tableau ordonné par la politique d'ordonnancement SJF qui se base sur le temps d'exécution : c'est à dire que le processus qui demande le moins de temps d'exécution sera élu.

De plus, il peut aussi continuer son exécution et ceci en choisissant une autre politique d'ordonnancement.

```

                                SHORTEST JOB FIRST
                                =====
+-----+-----+-----+-----+
| Processus | Temps d'arriver | Temps_d'exécution | Priorité |
+-----+-----+-----+-----+
| P(1)      | 0                | 5                  | 4        |
+-----+-----+-----+-----+
| P(3)      | 1                | 2                  | 2        |
+-----+-----+-----+-----+
| P(4)      | 3                | 4                  | 2        |
+-----+-----+-----+-----+
| P(2)      | 2                | 6                  | 4        |
+-----+-----+-----+-----+

DIAGRAMME DE GANTT
*****
|p(1) |p(3) |p(4) |p(2) |
0      5      7      11     17
Execution de l'algorithme SJF

Voulez vous continuer O/N ? █

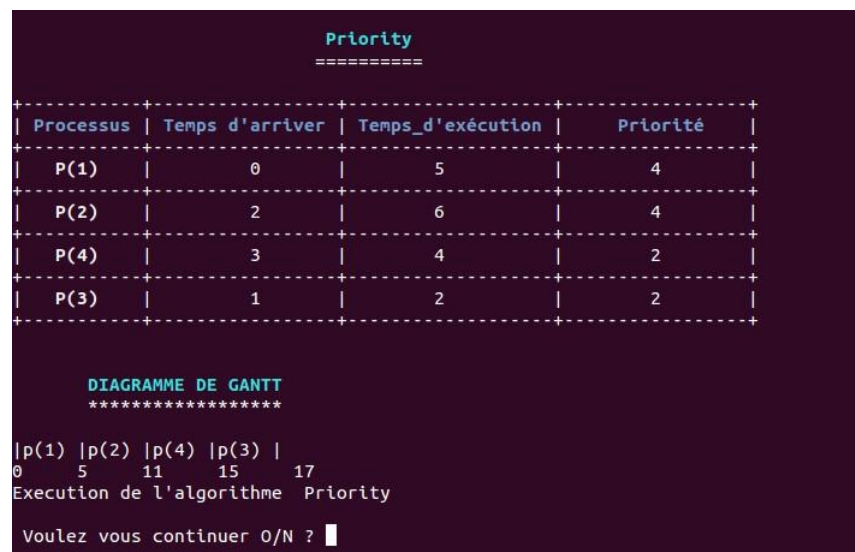
```

Figure 6: Exécution de l'algorithme SJF

4.5 Priorité

Suite à l'exécution de cet algorithme, l'utilisateur sera doté d'un tableau contenant les différents processus saisis précédemment, ordonné par la politique d'ordonnancement priorité qui se base sur la priorité : c'est à dire que le processus le plus prioritaire (qui a la priorité la plus élevée) sera élu.

De plus, ce dernier peut aussi continuer l'exécution des autres algorithmes et ceci en choisissant une autre politique d'ordonnancement.



```
Priority
=====
```

Processus	Temps d'arriver	Temps_d'exécution	Priorité
P(1)	0	5	4
P(2)	2	6	4
P(4)	3	4	2
P(3)	1	2	2

```
DIAGRAMME DE GANTT
*****
```

|p(1)|p(2)|p(4)|p(3)|

0 5 11 15 17

Execution de l'algorithme Priority

Voulez vous continuer O/N ? █

Figure 7 : Exécution de l'algorithme priorité

5 Développement graphique

Dans cette partie, nous avons implémenter une interface graphique en utilisant le module GTK et nous avons tout d'abord créer une fenêtre contenant l'écran d'accueil de notre application.

Suite au clic sur le bouton "Passer au programme", un menu principal s'affichera contenant 3 boutons :

- **Ajouter un processus** : permet d'ajouter un processus dans un fichier de configuration.
- **choix de l'algorithme** : permet de choisir un algorithme pour l'exécuter parmi ceux présenté.
- **Quitter** : permet de quitter le programme.

Suite au clic sur le bouton "choix de l'algorithme", une fenêtre apparaît dans laquelle vous pourrez sélectionner un fichier de configuration. Suite

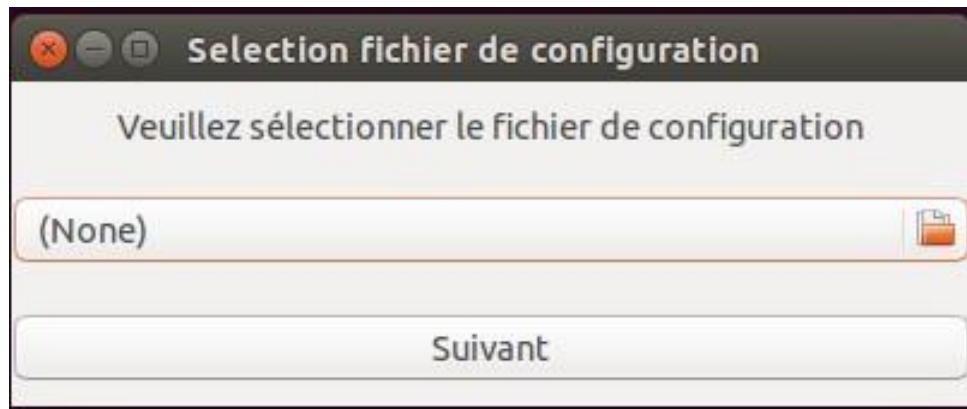


Figure 8 : S'élection du fichier de configuration

Au clic sur le bouton "suivant" on aura l'affichage du résultat dans une nouvelle fenêtre selon l'algorithme choisit.



Figure 9: Ecran d'accueil avec GTK

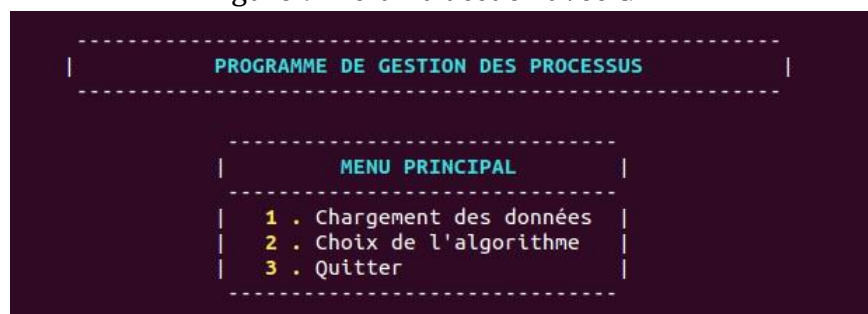


Figure 10 : Menu principal

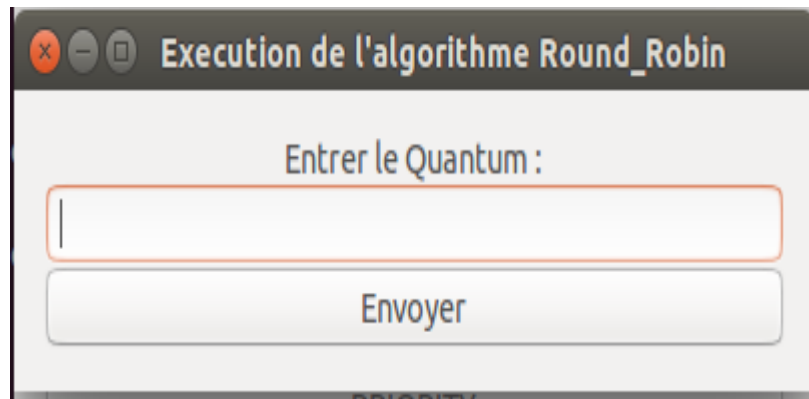
Execution de l'algorithme FIFO				
Processus	Temps d'arriver	Temps_d'exécution	Priorité	
P(1)	0	5	4	
P(3)	1	2	2	
P(2)	2	6	4	
P(4)	3	4	2	

P(1)	start : 0		finish : 5	
P(3)	start : 5		finish : 7	
P(2)	start : 7		finish : 13	
P(4)	start : 13		finish : 17	

Figure 11 : Exécution de l'algorithme FIFO avec GTK

Execution de l'algorithme SJF				
Processus	Temps d'arriver	Temps_d'exécution	Priorité	
P(1)	0	5	4	
P(3)	1	2	2	
P(4)	3	4	2	
P(2)	2	6	4	
P(1)	start : 0		finish : 6	
P(3)	start : 6		finish : 8	
P(4)	start : 8		finish : 12	
P(2)	start : 12		finish : 18	

Figure 12 : Exécution de l'algorithme SJF avec GTK



Processus	Temps d'arriver	Temps_d'exécution	Priorité
[P(1)]	start 0	Finish : 3	
[P(3)]	start 3	Finish : 5	
[P(2)]	start 5	Finish : 8	
[P(4)]	start 8	Finish : 11	
[P(1)]	start 11	Finish : 13	
[P(2)]	start 13	Finish : 16	
[P(4)]	start 16	Finish : 17	

Figure 13 : Exécution de l'algorithme RoundRobin avec GTK

Execution de l'algorithme SRT

Fichier de configuration

Processus	Temps d'arriver	Temps_d'exécution	Priorité
P(1)	0	5	4
P(2)	2	6	4
P(3)	1	2	2
P(4)	3	4	2

P(1)	start : 0
P(3)	start : 1
P(1)	start : 3
P(4)	start : 7
P(2)	start : 11

P(1)	finish : 7
P(3)	finish : 3
P(2)	finish : 17
P(4)	finish : 11

Figure 14 : Exécution de l'algorithme SRT avec GTK

Execution de l'algorithme Priority				
Processus	Temps d'arriver	Temps_d'exécution	Priorité	
P(1)	0	5	4	
P(2)	2	6	4	
P(4)	3	4	2	
P(3)	1	2	2	
P(1)	start : 0		finish : 6	
P(2)	start : 6		finish : 12	
P(4)	start : 12		finish : 16	
P(3)	start : 16		finish : 18	

Figure 15 : Exécution de l'algorithme priorité avec GTK

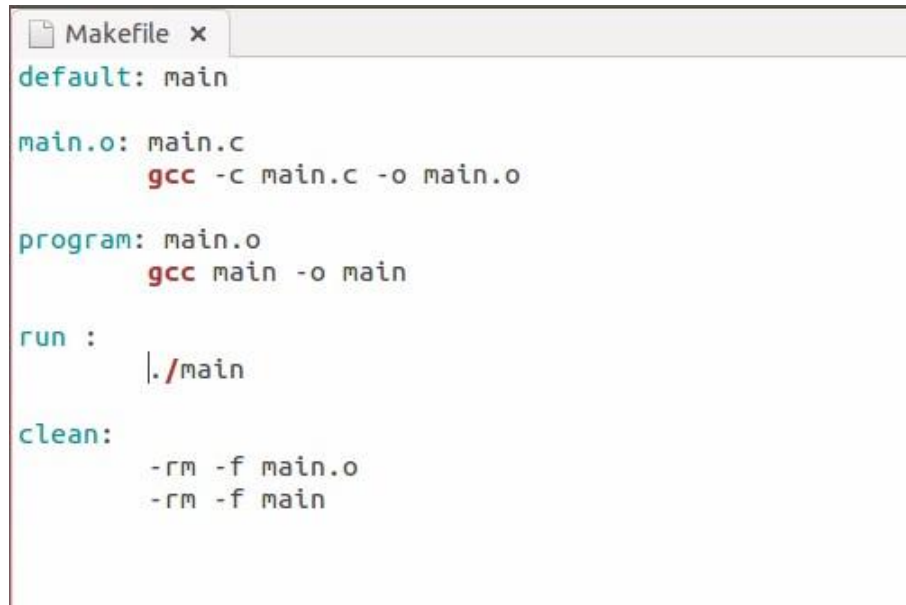
6 Création du fichier Makefile

Pour faciliter l'utilisation de notre programme, nous avons créé le fichier Makefile qui contient 3 commandes personnalisées :

main : génère des fichiers exécutables.

run: pour exécuter le programme que ce soit avec ou sans paramètres.

clean: pour supprimer le fichier exécutable.

A screenshot of a text editor window titled 'Makefile x'. The window contains the following text:

```
default: main

main.o: main.c
    gcc -c main.c -o main.o

program: main.o
    gcc main -o main

run :
    |./main

clean:
    -rm -f main.o
    -rm -f main
```

Figure 16 : Code du fichier Makefile

```
linux@ubuntu:~/Desktop/Project/Projet_SE$ make main
make: `main' is up to date.
linux@ubuntu:~/Desktop/Project/Projet_SE$ make run
./main
```

```
-----
| PROGRAMME DE GESTION DES PROCESSUS |
-----

-----
| MENU PRINCIPAL |
-----
| 1 . Chargement des données |
| 2 . Choix de l'algorithme |
| 3 . Quitter |
-----

Votre Choix :
```

Figure 17 : Exécution de la commande main et run

7 Conclusion

Afin de réussir notre projet nous avons élaboré la stratégie qui suit les étapes suivantes : D'abord, nous avons réalisé une étude complète pour comprendre le métier ensuite nous avons choisi les algorithmes adéquats.

Pour conclure, ce projet nous a permis d'apprendre de nouveaux acquis. En guise de perspectives, nous envisageons d'améliorer notre projet en ajoutant d'autres algorithmes d'ordonnancement.