

Guide développeur

(Logiciel de simulation)

Auteurs

- Ayoub S

Hossein S

Foudil N

Objectifs

Le but de travail est d'implémenter un logiciel de simulation d'ordonnancement des processus dans un système d'exploitation.

Ce guide permet aux développeurs de savoir comment la conception a été faite afin d'assurer une meilleure maintenabilité.

Choix techniques

- Langage utilisé : C
- Environnement de travail : Visual Studio Code

Analyse des besoins

Pour améliorer la rentabilité des machines, il est nécessaire de pouvoir exécuter des activités parallèles, comme exécuter un transfert d'E/S en même temps qu'un calcul interne au processeur. Dans ce cas le système doit exécuter plusieurs processus dans les meilleurs délais possibles en partageant le temps processeur.

Le processeur est la ressource la plus importante d'une machine. Cette ressource doit être allouée à un seul processus sélectionné parmi un ensemble de processus éligibles. En effet, le S.E dispose d'un module qui s'occupe de l'allocation du processeur en l'occurrence le Dispatcheur. Ce module est exécuté chaque fois qu'un processus se termine ou se bloque dans le but de réquisitionner le processeur pour un autre processus. En plus de Dispatcheur, le S.E dispose d'un autre module permettant ainsi la mise en ordre des processus qui demandent le processeur à savoir l'ordonnanceur (scheduler).

Les indicateurs de performances :

- Temps d'attente : c'est le temps passé à attendre dans la file d'attente des processus prêts.
- Temps de réponse : c'est le temps passé dans la file d'attente des processus prêts avant la première exécution.
- Temps de restitution : C'est le temps s'écoulant entre la soumission du travail et sa terminaison.
- Taux d'occupation : indique le montant d'une ressource utilisé pour un processus en particulier

Réalisation du travail

Nous avons commencé par définir l'ensemble des structures utiles à conception de notre logiciel, notamment :

```
6  typedef struct  Processus Processus;
7  typedef struct  Result Result;
8  struct Processus{
9      int dateArrive,dateFin;
10     int tempsExec;
11     int tempsRest;
12     int tempsAtt;
13     int tempsRep;
14     int dureeES;
15 };
16 struct Result{
17     int tempsAttMoy;
18     int tempsRestMoy;
19     int tempsRepMoy;
20     float tauxOccCPU;
21 };
```

Pour réaliser le processus détaillé ci-haut, nous avons l'architecture globale de notre programme définit comme suit :

```
int menuPrincipal();
```

```
void initData(Processus p[TAILLE_MAX], int n);
```

```
int lireFichier(FILE* file,Processus p[TAILLE_MAX]);
```

```
void trierTempsArrive(Processus process[TAILLE_MAX], int n);
```

```
void trierTempsRestant(Processus process[TAILLE_MAX], int n);
```

```
int exist(Processus process[TAILLE_MAX], int n);
```

```
int notin(int index[TAILLE_MAX],int t,int i);
```

```
Result calculResultat(Processus process[TAILLE_MAX], int n,int temps);
```

```
void enregistrer(Result resultat,Processus process[TAILLE_MAX], int n);
```

La tâche de chaque fonction est décrit en commentaire dans le projet.

Exécution du programme

L'exécution se fait en ligne de commande, il faut préalablement disposer du compilateur du langage et se rendre se trouvant le fichier main.c.

Avant d'exécuter ce programme, il faut préalablement compiler le fichier par la commande:

>> make

Et ensuite pour lancer ce programme, il faut taper l'une de deux commandes suivantes en fonction des données à fournir :

Soit :

>> ./main

Pour une insertion de données à travers la console.

Ou soit :

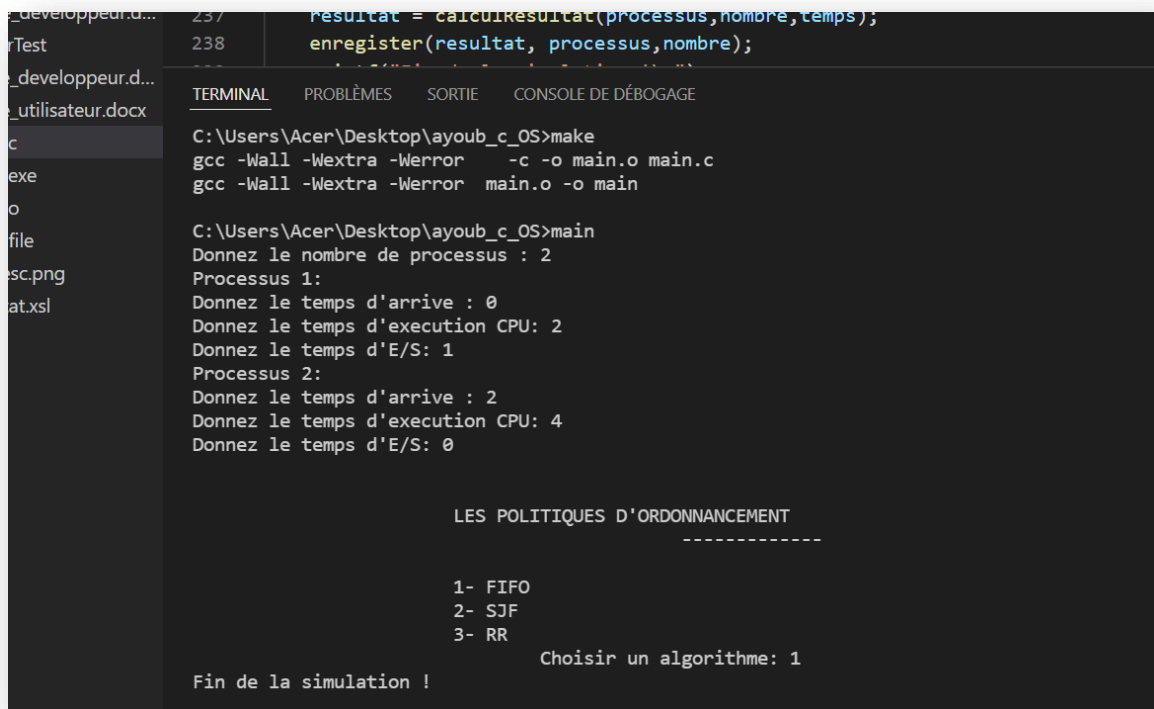
>> ./main 'nomfichier'

Si les données sont renseignées dans le fichier 'nomfichier.txt' et disponible dans le même répertoire.

Tests de validation

Voici le test effectué et le résultat fourni :

- Exécution et lancement du logiciel avec un exemple de processus et l'algorithme FIFO :



```
_developpeur.d... 237 resultat = calculer(resultat, processus, nombre, temps);
Test 238 enregister(resultat, processus, nombre);
_developpeur.d...
_utilisateur.docx
c
exe
o
file
sc.png
atxsl

C:\Users\Acer\Desktop\ayoub_c_OS>make
gcc -Wall -Wextra -Werror -c -o main.o main.c
gcc -Wall -Wextra -Werror main.o -o main

C:\Users\Acer\Desktop\ayoub_c_OS>main
Donnez le nombre de processus : 2
Processus 1:
Donnez le temps d'arrive : 0
Donnez le temps d'execution CPU: 2
Donnez le temps d'E/S: 1
Processus 2:
Donnez le temps d'arrive : 2
Donnez le temps d'execution CPU: 4
Donnez le temps d'E/S: 0

LES POLITIQUES D'ORDONNANCEMENT
-----

1- FIFO
2- SJF
3- RR

Choisir un algorithme: 1

Fin de la simulation !
```

- Résultat dans un fichier .xsl comme suit :

CPU						
	A	B	C	D	E	F
1	CPU	E/S	Attente	Reponse	Restitution	
2	2	1	1	0	3	
3	4	0	1	3	5	
4						
5	TAM	TRepM	TResM	Taux		
6	1.00	1.50	4.00	85.71		
7						
8						

Conclusion

Notre travail consiste à concevoir un logiciel de simulation d'ordonnancement des processus en langage C.

Pour atteindre notre objectif, nous avons entamer la modélisation grâce au langage de modélisation de nos différentes structures. Puis, nous sommes passés à la conception en écrivant le code au fonctionnement approprié.

Au niveau de la répartition des tâches, la contribution de chacun était équivalente, étant donné que nous avons presque tout réalisé ensemble à la bibliothèque de notre ville.

Le but principal du projet a été atteint, mais tout de même des perspectives d'amélioration de notre projet restent toujours indispensables.