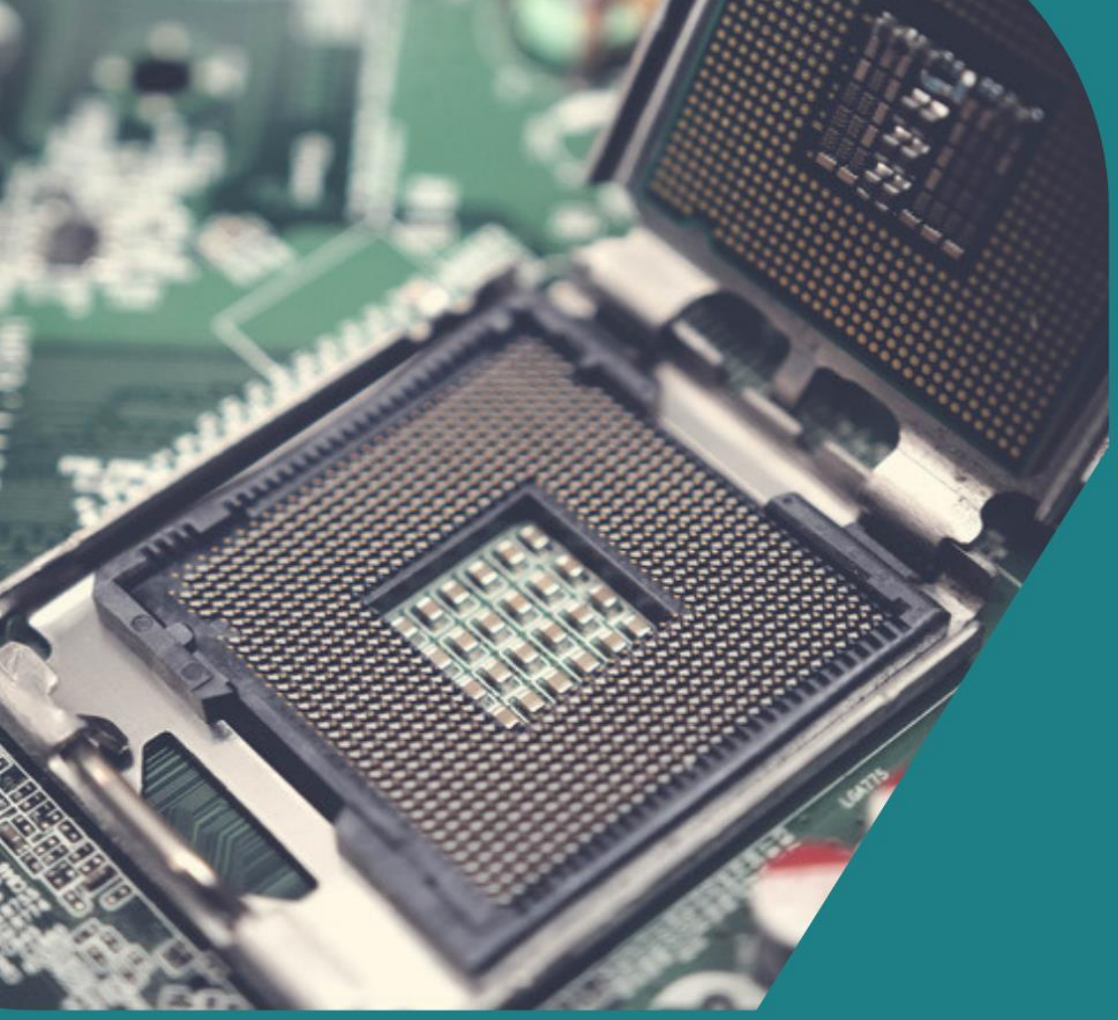


OPERATING SYSTEMS

ORDONNANCEUR DE PROCESSUS SOUS LINUX

Mini projet systèmes d'exploitation - 1 ING 2

PRÉSNTÉ À :

M. Hajer Ouerghi

PRÉSENTÉ PAR

Nassim Ben Nsib

Doniez Touil

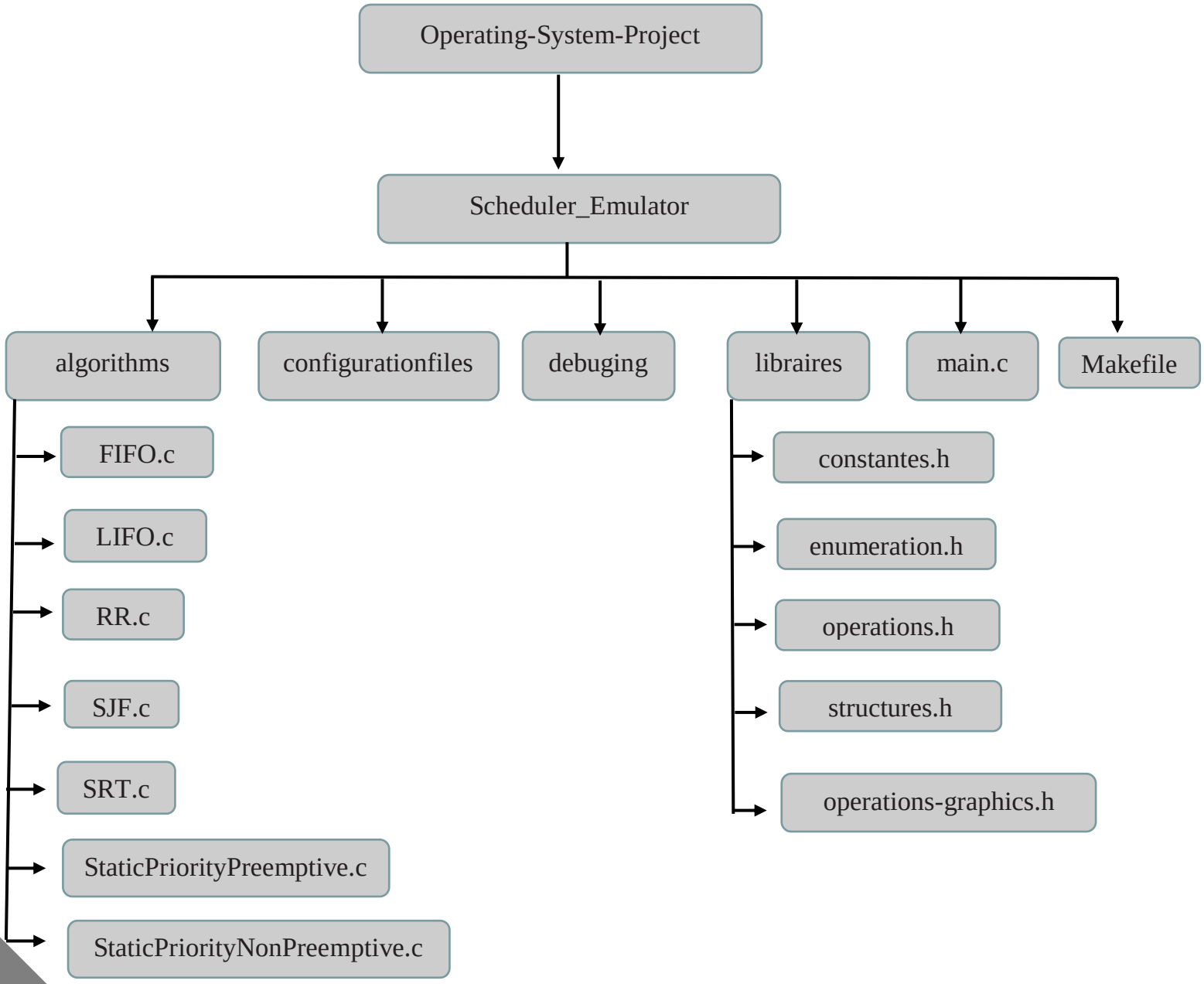
Introduction Générale

Ce document désigne le choix de structures de données et d'algorithmes faits pour réaliser un ordonnanceur de processus sous Linux.

Description du projet

Ce travail s'agit de réaliser un projet qui simule l'ordonnancement de processus, en fonction d'une politique d'ordonnancement et d'un ensemble de processus sélectionnés sous un système d'exploitation Linux

Description de l'architecture du projet



La figure ci-dessus illustre l'architecture générale de notre projet.

En effet, le fichier **algorithms** contient les algorithmes choisit lors de la réalisation de cet ordonnanceur.

Les algorithmes :

- FIFO.c : Contient le script d'algorithme "First In First Out".
- LIFO.c : Contient le script d'algorithme "Last In First Out".
- RR.c : Contient le script d'algorithme "Round Robin ou Tourniquet".
- SJF.c : Contient le script d'algorithme "Shortest Job First".
- SRT.c : Contient le script d'algorithme "Shortest Remaining Time".
- StaticPriorityPreemptive.c : Contient le script d'algorithme "Static priority preemptif".
- StaticPriorityNonPreemptive.c : Contient le script d'algorithme "Static priority not preemptif".

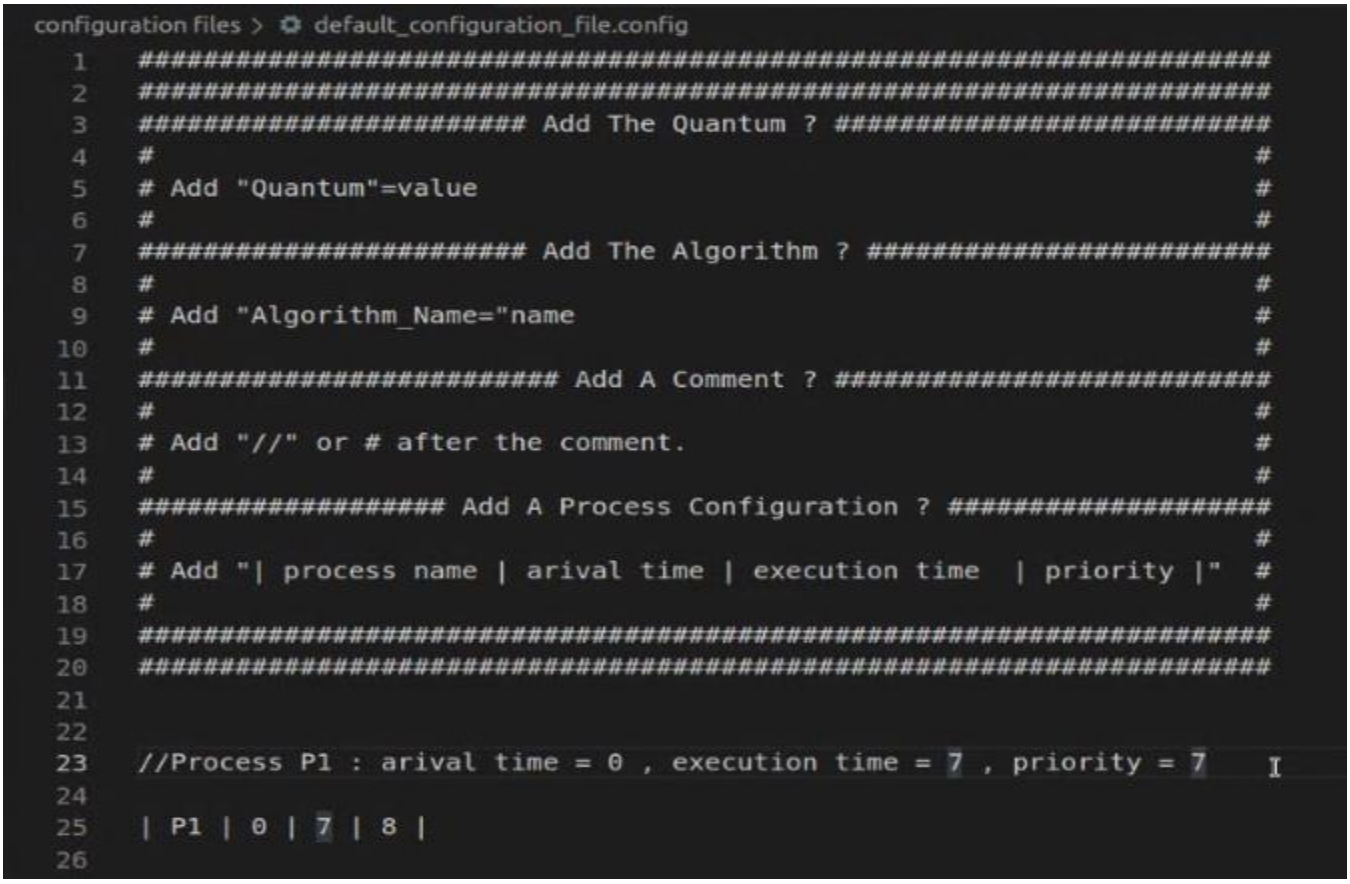
Le fichier Structures.h :

Contient les différentes structures utilisé pour avoir un bien déroulement des fonctions et optimiser le résonnement des algorithmes choisit.

Configuration files :

Contient un fichier .txt et ce dernier contient la liste des processus en question et leurs informations supplémentaires tels que ; leurs noms, temps d'arrivé, temps d'exécution et la valeur prioritaire séparée par le délimiteur **« | »** :

Nom | temps d'arrivé | temps d'exécution | priorité



Makefile :

Permet de construire le programme à partir des codes sources et des fonctions localisé dans le répertoire contenant les politiques d'ordonnancement.
La compilation se déroule dans le répertoire **« debuging ».**

Description des structures de données implémenté dans le projet

L'implémentation des structures de données est nécessaire lors de la réalisation de ce travail puisque nous sommes entrain de définir une variable caractérisée essentiellement par son nom, son temps d'arrivée, son temps d'exécution et sa valeur de priorité.
Pour l'optimisation du code nous avons défini cinq structures différentes chacune a une certaine utilité.

struct process :

```
//Structure Of Process
typedef struct process {
  char name[10];
  int priority;
  int current_priority;
  int arrival_time;
  int execution_time;
  int remaining_execution_time;
  int start_execution_time;
  int end_execution_time;
  struct process * next;
  struct process * previous;
}
process;
```

- ⇨ Cette structure est définie pour décrire la structure générale du processus.
- ⇨ C'est une structure générale qui est facile à utilisé pour tous types d'algorithmes d'ordonnancement quelque soit préemptif ou pas.

struct list_process :

```
//Structure Of List Process
typedef struct list_process {

  struct process * head;
  struct process * tail;
}
list_process;
```

⇨ Cette structure est définie pour réserver la liste des processus.

struct execution_history :

```
//Structure Of Process Execution History
typedef struct execution_history {
  int priority;
  int is_finished;
  int executed_time;
  int start_execution_time;
  struct process * process;
  struct execution_history *next;
  struct execution_history *previous;
}execution_history;
```

⇨ Cette structure est conserver pour stocker l'historique d'exécution de processus.

struct list_execution_history :

```
//Structure Of List Execution History
typedef struct list_execution_history{
  struct execution_history *head;
  struct execution_history *tail;
}list_execution_history;
```

⇨ Cette structure est définie pour réserver la liste d'historique d'exécution pour chaque processus.

struct emulator :

```
//Structure Of Emulator
typedef struct emulator {
  char configuration_file_name[100];
  char algorithm_name[20];
  int quantum;
  struct list_process list_incomming_process;
  struct list_process list_waiting_process;
  struct list_process list_finished_process;
  struct list_execution_history list_execution_history;
}
emulator;
```

⇨ Cette structure définie la configuration de l'émulateur utilisé pour faire la simulation des processus.