AI01
TP3

GAMONDELE Maxime
REDOUIN INNECCO Gabriel
23 novembre 2025

Table des matières

1 Modification du schéma de données

Une structure `FIFO` est introduite afin d'encapsuler une liste de processus et permettre des opérations d'enfilage et de défilage en O(1). La fonction `processus_free_recursive()` est ajoutée pour permettre la libération récursive d'une liste complète de processus. Une fonction d'affichage, `processus_print()`, est également définie afin de présenter les informations relatives à chaque processus de manière claire et structurée. La fonction `fifo_is_sorted()` est introduite pour vérifier que la FIFO est correctement triée selon un attribut donné. Enfin, la fonction `fifo_print()` fournit un affichage formaté du contenu de la FIFO. Comme la structure FIFO conserve désormais sa propre taille, il n'est plus nécessaire de fournir explicitement le nombre de processus lors de son utilisation.

2 Processus

- **t_processus *processus_init(int pid, int arrivee, int duree)**
  Crée et initialise un nouveau processus.
  Complexité : O(1) alloue une structure fixe.

- **void processus_free(t_processus **process)**
  Libère un processus unique.
  Complexité : O(1) libération d'un seul bloc mémoire.

- **void processus_free_recursive(t_processus *process)**
  Libère récursivement une liste de processus.
  Complexité : O(n) chaque nœud est libéré une fois.

- **t_processus *processus_load(char *nom_fichier, int *processes_loaded)**
  Lit un fichier et crée une liste chaînée de processus.
  Complexité : O(n) charge une ligne par processus.

- **void processus_print(t_processus *process)**
  Affiche récursivement la liste des processus.
  Complexité : O(n) parcours linéaire de la liste.

3 FIFO

- **FIFO *fifo_init()**
  Initialise une FIFO vide.
  Complexité : O(1) alloue une structure simple.

- **FIFO *fifo_init_from_process(t_processus *process)**
  Initialise une FIFO à partir d'une liste de processus.
  Complexité : O(n) copie chaque élément une fois.

- **FIFO *fifo_init_sorted_from_process(t_processus *process, PROCESS-FIELDS field)**
  Initialise une FIFO triée selon un champ donné.
  Complexité : $O(n^2)$ insertion triée répétée sur n éléments.

- **void fifo_free(FIFO **queue)**
  Libère une FIFO et tous ses processus.
  Complexité : O(n) libère chaque élément de la file.

- **bool fifo_is_empty(FIFO *queue)**
  Vérifie si la FIFO est vide.
  Complexité : O(1) test direct de la taille.

- **void fifo_add(FIFO *queue, t_processus *process)**
  Ajoute un processus à la fin de la FIFO.
  Complexité : O(1) insertion en queue direct car on possède un pointeur vers le dernier élément de la FIFO.

- **void fifo_add_sorted(FIFO *queue, t_processus *process, PROCESSFIELDS field, int ignore_first_n_index)**
  Ajoute un processus en maintenant l'ordre selon un champ.
  Complexité : O(n) recherche de la position d'insertion O(n) puis ajout O(1).

- **t_processus *fifo_unqueue(FIFO *queue)**
  Retire le premier processus de la FIFO.
  Complexité : O(1) mise à jour directe du pointeur de tête.

- **void fifo_print(FIFO *queue)**
  Affiche tous les processus dans la FIFO.
  Complexité : O(n) un affichage par élément O(1) n fois pour la taille de la liste donc O(n).

- **bool fifo_is_sorted(FIFO *queue, PROCESSFIELDS field)**
  Vérifie si la FIFO est triée selon un champ.
  Complexité : O(n) comparaison séquentielle de chaque éléments avec le suivant.

4 Ordonnancement

- **void simuler_fcfs(FIFO *tab)**
  Simule l'ordonnancement First-Come, First-Served (FCFS).
  Complexité : A VERIFIER

- **void simuler_sjf(FIFO *tab)**
  Simule l'ordonnancement Shortest Job First (SJF) non préemptif.
  Complexité : A VERIFIER