TD #6 – Ordonnancement des processus

Préambule

Objectifs :

- Implémenter quelques uns des algorithmes d'ordonnancement vus en cours
- Calculer des métriques simples de performance

Fichiers additionnels :

- scheduler.py – script à utiliser pour l'ensemble du TD, où seules les fonctions `xxx_routine()` sont à implémenter
- input_file_X.csv – exemples de fichiers d'entrée à utiliser avec le programme, le premier sans priorité et le second avec

Implémentation de l'ordonnanceur

À la queue leu leu

Votre ordonnanceur doit appliquer une règle simple : le premier processus souhaitant s'exécuter, s'exécute sur le processeur jusqu'à sa terminaison.

La tombola

Votre ordonnanceur choisit, parmi les processus attendant leur exécution, lequel s'exécutera.

Fast Pass

Votre ordonnanceur ajoute à l'algorithme FIFO, la notion de priorité : plus la valeur de priorité d'un processus est faible, plus il est prioritaire.

Petit à petit

Votre ordonnanceur utilise cette fois-ci un modèle préemptif. Chaque processus est sélectionné pour une durée de 2 secondes, puis replacé dans file d'attente. La file d'attente est traitée en FIFO sans priorité.

Performance

Modifiez les corps de vos routines pour récupérer les statistiques suivantes pour chaque processus :

- latence : durée totale d'exécution de l'arrivée du processus dans la file d'attente à sa terminaison
- temps d'attente : durée totale d'attente depuis l'arrivée du processus ie. temps durant lequel il n'est pas exécuté par le processeur

Une fois ceci fait, déterminez quel algorithme est le plus efficace ie. génère le moins de temps d'attente.

Supra-optimal

Votre ordonnanceur peut prévoir l'avenir : il connait la durée d'exécution de chaque processus. Ainsi, il va toujours chercher à exécuter le processus dont la durée d'exécution **restante** est la plus faible, quitte à mettre en pause l'exécution d'autres processus.

La description de la fonction est fournie dans `scheduler.py`. Attention, contrairement aux autres fonctions qui retournent le temps total d'exécution, vous devez ici retourner une chaîne de caractères indiquant à chaque quantum de temps, le nom du processus s'exécutant.

Par exemple, si `A` s'exécute durant 3 quantum, puis `B` durant 4 et enfin `A` durant 2, on aura : `AAABBBBAA`.