Aix*Marseille université

Domaine Sciences et Technologies
DIU "Enseigner l'informatique au lycée"

Bloc 3 : Systèmes d'exploitation
TD/TP 2

Année 2018-19

Processus et Ordonnancement

1 Travaux Dirigés

Exercice 1 Dans cet exercice, nous allons aborder d'autres algorithmes d'ordonnancement. La plupart se base sur des données qui sont en général de nature statistique associées aux processus.

Voici ces différents types d'ordonnancement :

— **Ordonnancement FIFO (First In First Out)** : ordonnancement sans préemption pour lequel les tâches sont exécutées (complètement) dans l'ordre de leur arrivée.

— **Ordonnancement SJF (Shortest Job First)** : ordonnancement sans préemption pour lequel les tâches sont exécutées (complètement) dans l'ordre de leur durée, les plus courtes étant exécutées en premier.

— **Ordonnancement SRT (Shortest Remaining Time)** : peut être vu comme la version préemptive de l'ordonnancement SJF. Dès que le processeur est réquisitionné, le processus dont le temps nécessaire à sa terminaison est le plus court est élu.

Pour mesurer la qualité d'un ordonnancement, il existe notamment deux critères :

— le **temps de réponse** : c'est le temps mis entre la date de création d'un processus et celui de la date où il devient actif pour la première fois.

— le **temps d'attente** : c'est le temps mis entre la date où il devient actif pour la première fois et la date où il se termine

Il convient de noter que dans le cas d'un processus unique pour un processus alors le temps de réponse est nul et le temps d'attente est la durée du processus.

Question 1.1 Soient cinq processus décrits par la table ci-dessous qui seront soit dans l'état "prêt", soit dans l'état "actif". Déterminez l'ordre d'exécution de ces cinq processus pour chacun des algorithmes d'ordonnancement suivants : FIFO (premier arrivé premier servi), SJF (le plus court d'abord), SRT (le temps restant le plus court) et RR (tourniquet avec un quanta fixé à 10 unités de temps). Pour chaque exécution et chaque processus, calculez le temps de réponse et le temps d'attente pour chacun des processus afin de comparer les stratégies d'ordonnancement.

Processus	date d'arrivée	durée
P1	0	10
P2	5	29
P3	35	3
P4	29	7
P5	24	12

Question 1.2 Nous avons évoqué l'équité comme une propriété souhaitable pour les algorithmes d'ordonnancement. Une propriété associée est l'absence de famine qui garantit qu'un processus continuellement dans l'état "prêt" finira par devenir "actif".

Supposons que nous soyons comme dans le cas de la question précédente en présence de processus "prêt" ou "actif" qui se créent régulièrement. Pour chaque algorithme d'ordonnancement, y a t-il un risque de famine ? Si oui, donner un tel scénario.

2 Travaux Pratiques

Exercice 2 Vous allez observer les processus qui s'exécutent sur votre machine à l'aide de la commande `ps` de votre shell `bash`.

La commande `ps` permet de lister les processus s'exécutant sur votre machine avec un grand nombre de paramètres. Vous pourrez connaitre l'ensemble des options disponibles en tapant la commande `man ps`.

Question 2.1 Tapez `ps` dans votre fenêtre shell. Lancez en tâche de fond votre application favorite depuis le shell `mon_appli &` Retapez `ps`. Que constatez-vous ?

Dans les informations affichées, identifiez votre idendifiant utilisateur (UID) et l'identifiant du processus de l'application lancée. "Tuez" avec la commande `kill -9` ce processus et constatez sa disparition avec `ps`.

Etendre votre recherche à l'ensemble de vos processus par `ps -u` *votre nom d'utilisateur*. Faire de même pour tous les processus système d'utilisateur `root`.

Question 2.2 Lancer au travers de l'interface graphique de votre SE une application (qui ne l'est pas actuellement). Le résultat de commande `ps` a-t-il changer ? Comment justifier cela ?

Retrouver l'application lancée en utilisant la bonne option de `ps` et en utilisant la commande `grep` pour vous aider à mieux voir.

Question 2.3 Relancez en tâche de fond votre application favorite depuis le shell `mon_appli &`. En utlisant la commande `ps -f` déterminer le père du processus de votre application.

Retrouvez cela en utilisant l'option d'affichage en arborescence `f`.

Afin d'étudier en temps réel, les processus s'exécutant sur votre machine, testez la commande `top`.

Question 2.4 Créer un script `infinite_loop.sh` qui ne fait rien d'autre que d'être une boucle infinie. Créer un autre script `launch_loop.sh` composé d'une boucle sans fin qui à chaque itération lance un `infinite_loop.sh`. Observez à l'aide la commande `top` le comportement de ces processus.