1 Problématique

Un processeur ne peut exécuter qu'une seule instruction à la fois. Pourtant sur un ordinateur, il est possible d'écouter de la musique tout en surfant sur le web.

Comment réaliser plusieurs activités en même temps sur une machine ?

2 Les processus

2.1 Définition

Nous travaillerons sur un système de type *Unix*. La page ci-après

https://tinyurl.com/y839kd4f

propose un simulateur d'un *terminal* Linux.
Un *programme* est un fichier en mémoire qui ne fait rien. Un *processus* est l'exécution d'un programme.
Il est possible de visualiser les processus en cours exécution en utilisant la commande **top** (figure 1).
Nous pouvons comparer cette commande au *gestionnaire de tâches* de Windows.

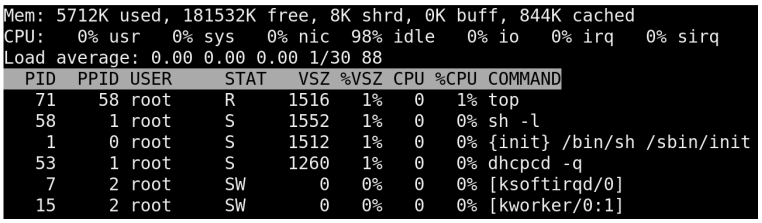

FIGURE 1 – Processus en cours

2.2 Création d'un processus

Chaque processus possède un identifiant unique, *le PID*. Au démarrage de la machine un premier processus spécial (*init*) est lancé. Ce processus crée d'autres *processus fils*. Ainsi chaque processus possède un (seul) parent, *le PPID*.
La commande

```
1 ps sort=pid
```

liste la totalité des processus.
Il est possible de tuer un processus en appelant l'instruction :

```
1 kill numéro_PID
```

3 Ordonnancement

3.1 Le chef d'orchestre

Dans le système plusieurs processus sont en cours simultanément, mais le processeur ne peut exécuter qu'une seule instruction à la fois. Le processeur travaille donc *en temps partagé*. Il bascule constamment d'un processus à l'autre.
L'ordonnanceur (scheduleur) sélectionne le prochain processus prêt (*Ready*) qui sera exécuté par le processeur. L'objectif est d'obtenir un *temps de traitement moyen* le plus court possible.

Christophe Viroulaud

3.2 Le scheduling

Les algorithmes d'ordonnancement peuvent être classés en deux catégories :

— **Non pré emptif :** Sélectionne un processus, puis le laisse s'exécuter jusqu'à ce qu'il bloque (soit sur une E/S, soit en attente d'un autre processus) où qu'il libère volontairement le processeur.

— **Pré emptif :** Sélectionne un processus et le laisse s'exécuter pendant un délai déterminé.

3.3 Quelques algorithmes d'ordonnancement

3.3.1 First Come First Served

Une fois que la CPU a été allouée à un processus, celui-ci garde la CPU jusqu'à ce qu'il la libère. Cet algorithme est particulièrement incommode pour le temps partagé où il est important que chaque utilisateur obtienne la CPU à des intervalles réguliers.

3.3.2 Shortest Job First

Quand la CPU est disponible, elle est assignée au processus qui possède le prochain cycle le plus petit. La difficulté est pouvoir connaître la longueur de la prochaine requête de la CPU.

3.3.3 Round Robin

Chaque processus a une petite unité de temps appelée *quantum* (en général de 10 à 100 ms). L'ordonnanceur parcourt la file d'attente des processus prêts et alloue la CPU à chaque processus pendant un quantum. La performance du *tourniquet* dépend fortement du choix du quantum de base.

Christophe Viroulaud