TP 3
Le système Linux : Gestion des processus

Introduction

Un processus est un programme en exécution. Plusieurs processus peuvent être exécutés en même temps. Ayant un seul processeur à sa disposition, le système d'exploitation met en œuvre une politique de partage de temps entre les processus. Ainsi un processus peut être dans l'un des trois états suivants :

- en exécution,
- en attente d'exécution : dans cet état le processus a accès à toutes les ressources qui lui sont nécessaires sauf le processeur occupé à exécuter un autre processus ,
- bloqué : dans cet état le processus attend une ressource non encore disponible comme par exemple le cas d'un programme qui demande à l'utilisateur de saisir une valeur.

Le système alloue à chaque processus une durée limité d'accès au processeur (appelé quantum). A l'expiration de chaque quantum, le système choisit un autre processus parmi ceux en état d'attente pour l'exécuter pendant un quantum. Différentes politiques de sélection de processus peuvent être appliquées (système de priorité par exemple). Ces politiques seront étudiées en cours des systèmes d'exploitation.

Exercice 1 Donner le diagramme d'états d'un processus

Au démarrage du système un processus nommé `init` est lancé. Il est responsable du lancement des autres processus du système. Un processus peut en lancer d'autres. Ainsi les processus sont groupés en une hiérarchie dont la racine est le processus `init`. Chaque processus est identifié par :

- un numéro unique (`PID` : process identifier),
- l'identité du propriétaire (par exemple l'utilisateur qui l'a exécuté)
- et par le numéro du processus père (`PPID`) celui à partir duquel il a été lancé.

Chaque processus possède son propre environnement d'exécution. Les processus peuvent communiquer entre eux par l'envoi de messages appelés signaux. L'envoi d'un signal se passe d'une manière asynchrone,autrement dit la réception d'un signal est un événement non prévisible. Par contre les signaux possibles sont d'un nombre limité. Les réactions aux signaux sont prédéfinies. Par exemple on pourra arrêter un processus en lui envoyant le signal `SIGINT` (signal numéro 2) généré depuis le clavier en appuyant simultanément sur les touches `CTRL+C`.

En bash, la commande `trap` permet d'inhiber ou activer la réaction associé un signal. Certains signaux, comme le signal de terminaison `SIGKILL` (signal numéro 9) ne peuvent pas être inhibés. La commande `trap -l` affiche la liste des signaux avec leur nom. La commande

```
trap ' ' numéro de signal
```

permet d'inhiber, si possible, la réaction associée au signal cité. Pour activer le réaction il suffit de taper la commande : `trap numéro du signal.`

Exercice 2

1. Donner des exemples de quelques signaux avec leur numéro
2. Soit le script `bash` suivant :

```
i= 0
while true
do
        i=`expr $i + 1`
        echo "Valeur: $i"
        sleep 2
done
```

   Éditer ce script dans un fichier nommée ex et exécuter le. Que fais ce script ?
3. Dans un shell bash, inhiber le signal 2 puis exécuter le programme ex. Essayer d'interrompre le programme par les touches CTRL+C. Quelle est la réponse du système ? Essayer les touches CTRL+Z. Quelle est la réponse du système ?
4. Activer la réaction associé au signal 2 et refaire la question 2.

Gestion des processus

Les principales commandes de gestion de processus sont les suivantes :

Commande	Description
`ps`	Affiche les informations sur les processus en cours d'exécution
`kill`	Envoi un signal à un processus
`nice`	Changer la priorité d'un processus
`nohup`	Lancer un processus qui ne sera pas terminé même après la déconnexion de l'utilisateur

Exercice 3

Consulter les pages de manuel pour les commandes citées ci-haut et répondre aux questions suivantes :

1. En utilisant la commande `ps`, tracer le chemin dans la hiérarchie de processus qui mène du processus de shell de connexion d'un utilisateur au processus racine init.
2. Dans un *shell bash* lancer le programme ex puis interrompre l'exécution en tapant CTRL+Z. Donner deux manières pour tuer le processus suspendu (celui correspondant au programme) en utilisant la commande `kill`.
3. Modifier le programme donné en exemple 3 pour que la valeur du compteur soit sauvegardée dans un fichier au lieu d'être affichée sur l'écran. Puis lancer le programme modifié de sorte qu'il survive à la déconnexion de l'utilisateur (en utilisant la commande

nohup). Vérifier le fonctionnement de cette commande. Quel est le processus père de celui du programme ?

Traitement différé

Deux principales commandes servent au lancement en différé de processus : la commande at et la commande `crontab`. Les utilisateurs autorisés à utiliser la commande `at` (respectivement la commande `crontab`) sont donnés dans le fichier /etc/at.allow (respectivement /etc/cron.allow). D'une manière similaire les fichiers `/etc/at.deny` et `/etc/cron.deny` donnent les utilisateurs non autorisés à utiliser ces commandes.

La commande `at` permet de spécifier une date (jour et heur) du lancement d'un processus.

La commande `crontab` permet de spécifier des tâches périodiques. La première est utile pour lancer par exemple des tâches qui nécessite une grande consommation de ressources à des périodes où le système est moins chargé (la nuit par exemple). La deuxième est plus utile pour les tâches administratives comme par exemple : nettoyer le répertoire `/tmp` toute les nuit, sauvegarder les données utilisateurs sur une bande tous les jours, etc.

Pour utiliser la commande `at`, l'utilisateur (autorisé) tape `at` suivi de la date du lancement voulu. Par exemple : at 09:00 01/02/06 pour dire qu'il a un processus à lancer à neuf heur le 1 février 2006.

Cette commande commence une dialogue interactive avec l'utilisateur pour spécifier le processus à lancer. La fin de saisie des informations se fait en tapant CTRL+D. Les processus planifiés par at seront placés dans le répertoire /var/spool/at. L'utilisateur peut consulter les processus planifiés par lui via la commande `atq` et peut effacer un processus planifié en utilisant la commande `atrm`.

La commande `crontab` a deux options principales :

-e édite le fichier de planification de l'utilisateur
-l Affiche le fichier de planification de l'utilisateur

Le fichier de planification est un fichier texte, où chaque ligne correspond à une tâche. Les lignes de commentaires commencent par #. Chaque champs est une valeur numérique, une liste de valeur ou un intervalle. Le caractère * remplace toutes les valeurs. Une liste est une suite de valeurs Séparées par des virgules et un intervalle est représenté par la première et la dernière valeur séparées par -.

Une ligne est composée de 6 champs qui sont dans l'ordre : 1) les minutes 2) les heures 3) le jour du mois 4) le moins 5) le jour de la semaine (dimanche=0 et lundi=1, ..) et 6) la tâche à lancer. Ainsi l'entrée suivante : `0,30 8-20 * * 0 check` précise que tous les dimanches de 8h à 20h toutes les heures pleines et les demi heures lancer le programme *check*.

Exercice 4

1. Utiliser la commande `at` pour lancer en différé une commande qui écrit l'heure dans un fichier texte. Décommander cette tâche.
2. Utiliser la commande `at` pour nettoyer `/tmp` dans 10 minutes
3. Planifier le nettoyage de `/tmp` tous les soirs à 3h le matin.