TD1 : Création de processus

Exercice 1 : Arcorescence d'un processus

Objectif : Identifier l'arborescence de processus sous Linux

Un processus peut créer un ou plusieurs processus à l'aide d'une commande système ("fork" sous les systèmes de type Unix). Imaginons un processus A qui crée un processus B. On dira que A est le père de B et que B est le fils de A. B peut, à son tour créé un processus C (B sera le père de C et C le fils de B). On peut modéliser ces relations père/fils par une structure arborescente.
Sous un système d'exploitation comme Linux, au moment du démarrage de l'ordinateur un tout premier processus (appelé processus 0 ou encore Swapper) est créé à partir de "rien" (processus à la racine de l'arborescence). Ensuite, ce processus 0 crée un processus souvent appelé "init" ("init" est donc le fils du processus 0). À partir de "init", les processus nécessaires au bon fonctionnement du système sont créés (par exemple les processus "crond", "inetd", "getty",...).

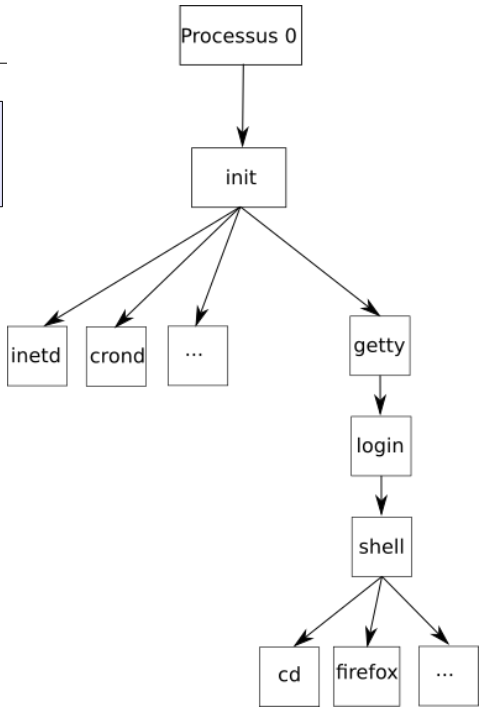

Figure 1: Arborescences des processus sous Linux

PID et PPID

Chaque processus possède un identifiant appelé PID (Process Identification), ce PID est un nombre. Le premier processus créé au démarrage du système à pour PID 0, le second 1, le troisième 2... Le système d'exploitation utilise un compteur qui est incrémenté de 1 à chaque création de processus, le système utilise ce compteur pour attribuer les PID aux processus.
Chaque processus possède aussi un PPID (Parent Process Identification). Ce PPID permet de connaitre le processus parent d'un processus (par exemple le processus "init" vu ci-dessus à un PID de 1 et un PPID de 0). À noter que le processus 0 ne possède pas de PPID (c'est le seul dans cette situation).

1- En se basant sur la figure 1, **donner** le PID (en partant du principe qu'il est créé juste après init) et le PPID du processus "**getty**".

2- **Ouvrir** un terminal sous linux (machine online disponible sur *https://cocalc.com/*) taper la commande suivante :

```
>>> ps -aef
```

3- **Recopier** le résultat obtenu par cette commande et **dessiner** l'arborescence des processus exécutés.

La commande ps ne permet pas de suivre en temps réel les processus (affichage figé). Pour avoir un suivi en temps réel, vous pouvez utiliser la commande `top`.

4- **Tester** cette commande sur un terminal. Taper q pour stopper la commande.

Exercice 2 : Mise en concurrence de plusieurs processus

Objectif : Illustrer le fonctionnement de tâche concurrentes dans un système d'exploitation.

On dispose de deux fichiers dont le but est d'écrire en continue un chiffre sur la console. Les scripts bash sont les suivants :

Script 1	Script 2

`#!/bin/bash`
`while true`
`do`
`echo 111111111111111111111111`
`done`
	`#!/bin/bash`
	`while true`
	`do`
	`echo 2222222222`
	`done`

1- **Décrire** le travail réalisé par chaque script

2- Dans une console linux (https://cocalc.com/projects/73da63a9-6f93-4c92-81d5-91a1cfe406bb/files/Welcome%20to%20CoCalc.term?session=default), **exécuter** les deux fichiers simultanément avec la commande :

```
>>> ./test1.sh & ./test2.sh
```

3- **Copier** le l'affichage obtenu sur la console et **analyser** le résultat. **Indiquer** notamment l'ordre d'exécution des tâches.

4- Dans une autre console, **vérifier** l'existence de ces tâches. **Relever** leur PID et leur père commun. **Arrêter** ces tâches avec la commande :

```
>>> kill -9 PID
```

Exercices 3 : Partage de ressources par plusieurs tâches

Le fichier python suivant écrit dans un fichier les nombres compris entre 0 et 999 précédés du PID du processus qui exécute ce programme.

```
from os import getpid
pid = str(getpid())
with open ('test.txt', 'w') as file :
    for i in range(100000) :
        file.write('['+pid+'] : ' + str(i) + '\n')
        file.flush()
```

1- **Exécuter** ce programme et **relever** quelques lignes du fichier qui a été créé.

2- **Exécuter** ce même fichier par trois tâches différentes avec la commande :

```
>>> python3 ecritfichier.py & python3 ecritfichier.py &python3 ecritfichier.py
```

3- **Relever** quelques lignes du fichier créé et **interpréter** ce résultat