Systèmes d'exploitation

TP1 : Processus

Objectifs du TP

Implémenter des simulations de gestion de processus

Consigne

Lisez bien l'énoncé, il contient beaucoup d'information et chaque mot est utile!

N'hésitez pas à chercher des informations sur les fonctions dans le cours et sur Internet, un bon informaticien doit savoir chercher des informations en ligne.

Environnement de travail

Le cours étant surtout basé sur des exemples Linux, nous allons travailler dans un environnement Linux. Vous pourrez éditer vos fichiers C avec le programme `Gedit`.

Compilation et exécution de programmes C

Pour compiler un programme C, il faut utiliser `gcc` (GNU Compiler Collection). Si vous voulez compiler un programme qui s'appellerait `monProgramme.c`, il faudra dans un terminal vous mettre dans le répertoire de ce fichier (utilisez la commande `cd`) et taper la commande `gcc monProgramme.c`. Cette commande compilera votre code C et créera un fichier exécutable qui s'appellera `a.out`. Pour exécuter ce fichier `a.out`, il vous suffira de taper la commande `./a.out`.

⚠Si vous voulez donner un nom à votre programme (et éviter d'avoir un fichier nommé `a.out`), vous pouvez préciser un nom cible en utilisant l'option `-o`, par exemple `gcc monProgramme.c -o monProgramme` créera un fichier exécutable nommé `monProgramme`.

---

1 Étude des processus

1.1 Quelques commandes utiles

1- A l'aide de la commande **`ps`**, affichez la liste des processus en cours d'exécution. Que remarquez-vous?

2- Trouvez l'option qui permet d'afficher tous les processus tournant sur votre machine.

3- Sous UNIX, chaque processus (excepté le premier) est créé par un autre processus, son processus père. Le processus père d'un processus est identifié par son PPID (Parent PID). Trouvez une option de la commande ps permettant d'afficher le PPID d'un processus.

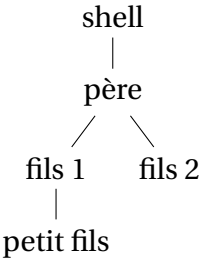
4- Essayez la commande **`top`**. Quelle est la différence?

1.2 Création de processus en C

En C, la bibliothèque **unistd.h** contient les primitives permettant de gérer les processus. N'oubliez pas de l'inclure dans vos programme (en plus de **stdio.h** pour les éventuels **printf**).

1- Écrivez un programme en C qui affiche le numéro d'identification du processus (**getpid**) associé et celui de son père.

2- La creation d'un processus se fait via la primitive **fork**. Le processus père est alors copié en un processus fils et les deux processus s'exécutent en parallèle. Ecrivez un programme ayant la structure suivante et qui affiche dans chaque processus son **pid** et le **pid** de son père :



3- La famille de primitives **exec** permet le lancement d'un programme. Il n'y a pas création d'un nouveau processus, mais simplement une mutation du processus. Écrivez un programme qui exécute la commande **date**. Ensuite, écrivez un programme qui crée un processus, qui lance la commande **date** dans le père et affiche "Bonjour" dans le fils. Vous pouvez utiliser la primitive de la famille **exec** de votre choix.

4- Il est également possible d'envoyer des signaux particuliers à des processus. Par exemple, un signal **kill** peut être envoyé pour "tuer" un processus. La primitive **kill(pid, SIGKILL**), définie dans la bibliothèque **signal.h**, envoie le signal **SIGKILL** au processus **pid** pour le "tuer". Ecrivez un programme qui crée un processus fils puis le "tue". Pour vérifier que tout se passe comme prévu :

— Dans le processus fils, faites une boucle infinie qui affiche du texte à l'écran

— Dans le processus père, mettez en pause **n** secondes le processus avec la primitive **sleep(n)** puis "tuez" le processus fils.

5- Reprenez le programme de la question 1 et "tuez" le processus père. Que se passe-t-il? Attention, l'utilisation de la commande kill se fait avec précautions!