Systèmes d'exploitation

TP - Processus

ING1 Informatique - Mathématique apliquée

Année 2022–2023

La fonction `fork()` permet de dupliquer un processus existant afin d'accéder à la programmation parallèle. Il existe tout un tas d'autres techniques pour faire ceci, mais dans le cadre de ce cours nous nous intéresserons uniquement à la fonction `fork` et ses utilisations.

Premier contact

1 Écrire un programme qui affiche les information suivantes associées à un processus :

- Le numéro du processus (`pid`)
- le numéro du père du processus (`ppid`)
- l'UID réel du processus (`uid`)
- l'UID effectif du processus (`euid`)
- le GID réel du processus (`gid`)
- le GID effectif du processus (`egid`)

Un exemple d'exécution est :

```
./a.out
Je suis le processus de pid      : 20011
Mon père est le processus de pid :  5411
Mon uid                          :   322
Mon euid                         :   322
Mon gid                          :   100
Mon egid                         :   100
```

□

(2) Écrire un programme qui crée un processus fils et qui affiche les informations `pid` et `ppid` de chaque processus créé ainsi que la valeur retournée par `fork()`. Un exemple d'exécution est :

```
./a.out
Valeur de fork = 22723
Je suis le processus père : pid=22722, ppid=5411,pid fils = 22723
Valeur de fork = 0
Je suis le processus fils : pid=22723, ppid 22722
```

— Que constatez vous concernant la valeur du `fork()` ?

— Est-il possible d'avoir deux morceaux de codes exécutés qu'une seule fois ?

□

(3) Reprendre l'exercice 1 et affichez les informations relatives aux processus père et fils comme suit :

```
/a.out

Valeur fork = 0
Je suis le processus de pid      : 22851
Mon père est le processus de pid : 22850
Mon uid                          : 322
Mon euid                         : 322
Mon gid                          : 100
Mon egid                         : 100

Valeur fork = 22851
Je suis le processus de pid      : 22850
Mon père est le processus de pid : 5411
Mon uid                          : 322
Mon euid                         : 322
Mon gid                          : 100
Mon egid                         : 100
```

□

Tel père, tel fils

4 Dans un programme qui utilise la fonction `fork`, vous créerez des variables de types primitifs (`int`, `float`, etc.). Vous les initialisez avant l'utilisation de `fork`, puis vous les affichez dans le code du père et dans le code du fils. Pour chaque variable, vous afficherez aussi l'adresse de la variable ainsi que sa valeur. Que constatez vous par rapport aux valeurs des variables ? □

5 Écrivez un programme C qui crée un fils. Chaque processus doit afficher son PID à l'écran. Ensuite, le père doit attendre la terminaison du fils. Lorsque le fils termine, il enverra un code de retour. Le père devra récupérer cette code et l'affichage à l'écran, ainsi que le pid du fils qui vient de se terminer. □

6 Dans un programme qui utilise la fonction `fork`, vous créerez quatre fils à partir du même père. Mettez le père en attente (avec `wait` ou `waitpid`) jusqu'à la terminaison des fils et, ensuite, continuez l'exécution du père.

— Quelle est la différence entre `wait` et `waitpid` ?

□