IUT de Montpellier, Dép. Informatique
M2101 - Archi 2 V. Poupet

# TD n° 5 - Manipulations de processus

**Exercice 1.** *Copie du processus courant*

On considère le programme suivant :

```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

int main(int argc, char **argv) {
  int i;
  printf("PID: %d (avant fork)\n", getpid());
  i = fork();
  if (i != 0) {
    printf("père PID: %d, résultat du fork: %d\n",getpid(),i);
  } else {
    printf("fils PID: %d, résultat du fork: %d \n",getpid(),i);
  }
  printf("PID: %d (après fork)\n", getpid());
}
```

1. Recopiez, compilez et exécutez le programme. Expliquez les résultats observés.

2. En utilisant la procédure `unsigned int sleep(unsigned int seconds)` de la bibliothèque `unistd.h`, faites en sorte que les lignes d'affichage du processus fils s'exécutent 5 secondes après celles du processus père.
Que se passe-t-il lorsque l'on exécute le programme ?

3. Exécutez le programme de la question précédente puis lancez la commande « `ps -l` » avant que le fils ait fini de s'exécuter (vous avez 5 secondes). Que remarquez-vous ?
**Indication :** Combien de processus sont en cours, quels sont leurs parents ?

On considère maintenant le programme suivant :

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

int main(int argc, char **argv) {
  int i, j, s, tabpid[3];
  printf("[avant fork] PID: %d\n", getpid());
  for(j = 0; j < 3; j++) {
    tabpid[j] = fork();
    if (tabpid[j] != 0 ) {
```

```
      printf("[père] PID: %d, retour fork: %d \n",getpid(),tabpid[j]);
    } else {
      printf("[fils] PID: %d \n",getpid());
      exit(j);
    }
  }
  for(j = 0; j < 3; j++) {
    i= waitpid(tabpid[j], &s, 0);
    if (i > 0) {
      printf("terminé PID: %d\n", i);
    }
    sleep(1);
  }
}
```

**4** Que fait ce programme ?

**5** Que récupère la variable `s` ? Modifiez le programme pour qu'il affiche le code de retour de chacun des processus fils.

**Indication :** Il faut chercher comment trouver le code de retour d'un processus dans la documentation de `waitpid`.

**6** Modifiez le programme pour que chacun des fils affiche les valeurs de `tabpid[0]`, `tabpid[1]` et `tabpid[2]`. Observez ce qui se passe lors de l'exécution. Comment l'expliquez-vous ?

**7** Reprenez le programme `negatif` écrit au TD 4. Rajouttez un appel `fork` juste après avoir recopié l'en-tête du fichier, puis faites en sorte que le processus père génère le négatif de l'image tandis que le fils recopie l'image sans la modifier. Observez le fichier obtenu. Comment expliquez-vous cela ?

**8** Exécutez le même programme avec une image en couleur. Expliquez le résultat.

**Exercice 2.** *Exécution*

**1** En utilisant la fonction `execlp`, écrivez un programme qui exécute la commande « `ps -l` ».

**Indication :** Il faut appeler la commande `execlp` pour remplacer le code du processus courant par celui de la commande `ps`, tout en lui donnant les bons arguments pour lui passer l'option `-l`.

**2** Écrivez un programme `lancer.c` qui se clone à l'aide de l'appel `fork` et dont le processus fils exécute l'exécutable passé en paramètres avec ses arguments. Par exemple l'appel « `./lancer ls -l *` » doit exécuter la commande « `ls -l *` »

**Indication :** Il faut utiliser l'appel `fork` puis utiliser une des commandes de la famille `exec` pour remplacer le code du processus fils par celui du programme passé en argument, en lui transmettant également tous les autres arguments.