Système d'exploitation II

TP 2 : Héritage et communication entre Processus

Solution

Partie I

Comme c'est expliqué dans le cours chaque processus est défini par un ensemble d'attributs (voire tableau). Pour visualiser les attributs d'un processus nous pouvons afficher la table des processus, chose qui n'est pas pratique. Ainsi nous pouvons afficher les attributs du processus en cours d'exécution via certaines fonctions disponibles dans les APIs système de linux.

attribut	Explication	Fonction
UID	User ID (int)	getuid()
GID	Group ID (int)	getgid()
PID	Process ID (int)	getpid()
PPID	Process ID du processus père (int)	getppid()
CWD	Répertoire personnel (chaine de caractère)	getcwd()

Pour l'utilisation de ces fonctions il suffit de faire un appel dans la partie du code du fils, c'est-à-dire à l'intérieur de la condition.

- *Ecrire un programme C « prog1.c » sous TP2, qui permet de créer un processus fils.*

```
#include <sys/times.h>
int main()
{ ptintf("je suis le processus pere\n") ;
  //creation du fils
   int p = fork() ;
  if(p==0)
{
// c'est la partie du fils car la valeur de p=0
 pringf(« je suis le fils ») ;
}
else
{
if(p>0)  // partir du père
          {printf(« suite du père ») ;}
else // p<0
          {printf (« erreur d'execution » ) };
```

```
}
return 0;}
```

Remarque : si vous vous faites copier/coller à partir du fichier Word vers linux vous aurez des caractères super flux qui poseront problème lors de la compilation. Vous devez réecrire le code dans un éditeur linux (vi, nano, gedit)

- *Modifier « prog1.c » afin d'afficher les attributs de chaque processus (père et fils), le **pid**, **ppid**, le **uid, guid** et le répertoire de travail. Pour la récupération du répertoire en utilise la syntaxe suivante : printf(" répertoire de travail : %s\n ",getcwd(buf,1024)), avec buf est une variable de type chaine de caractère.*

Pour afficher les attributs des processus il suffit d'appeler les fonctions mises dans le tableau, dans la partie de code du processus concerné.

```
#include <sys/times.h>
int main()
{ ptintf("je suis le processus pere\n") ;
// affichage des attributs du père
    int pid=getpid() ;
    int ppid=getppid() ;
    int gid=getgid() ;
    int uid=getuid();
   printf ("PID=%d   PPID=%d   UID=%d   GID=%d",pid,ppid,gid,uid);
   printf(" répertoire de travail : %s\n ",getcwd(buf,1024))
 //creation du fils
   int p = fork() ;
  if(p==0)
{
// c'est la partie du fils car la valeur de p=0
 pringf(« je suis le fils ») ;
// affichage des attributs du fils
     int pid=getpid() ;
   int ppid=getppid() ;
    int gid=getgid() ;
    int uid=getuid();
   printf ("PID=%d   PPID=%d   UID=%d   GID=%d",pid,ppid,gid,uid);
   printf(" répertoire de travail : %s\n ",getcwd(buf,1024))

}
else
{
if(p>0)  // partir du père
           {printf(« suite du père ») ;}
else // p<0
           {printf (« erreur d'execution » ) };
}
```

```
return 0;}
```

- *Quelles sont les attributs différents entre les deux processus. Expliquer.*

On remarque que tous les attributs sont identiques sauf le PID et le PPID. Ce qui s'explique par l'unicité du PID et que les deux processus n'ont pas le même père ainsi les deux PPID sont différents.

On peut conclure qu'un processus hérite tous ces attributs de son père sauf le PID et le PPID.

Partie II :

Après l'héritage on veut maintenant observer comment le passage des variables est effectué du père au fils.
Dans la littérature nous avons deux méthodes passage par adresse et passage par valeur. Le passage par adresse consiste à donner la valeur de la variable mais aussi l'adresse. Tandis que le passage par valeur consiste à donner juste la valeur de la variable.
Pour identifier la méthode il suffit de comparer les valeurs et les adresses des variables dans le fils et le père.

Exécuter le code :

```
#include <sys/times.h>
int n=1000;
        main(){int m=1000, pid;
        printf("Adresse de n dans le père: %p\n ", &n);
        printf("Adresse de m dans le père: %p\n ", &m);
        printf("1a valeur de m et n dans le père : %d %d\n ", m, n);
        switch(pid=fork()){
            case        -1:     perror("fork");exit(2);
            case        0 :                     /* on est dans le processus fils*/
                                printf("Adresse de n dans le fils: %p\n ", &n);
                                printf("Adresse de m dans le fils: %p\n ", &m);
                                printf("2 valeur de m et n dans le fils : %d %d\n ", m, n);
                                m*=2;n*=2;
                                printf("3 valeur de m et n dans le fils : %d %d\n ", m, n);
                                sleep(3);
                                exit(0);
            default:                            /*on est dans le processus père*/
                                sleep(2);
                                printf("4 valeur de m et n dans le père : %d %d\n ", m, n);
                                m*=3;n=*3;
                                printf("5 valeur de m et n dans le père : %d %d\n ", m, n);
                                sleep(2);
```

```
exit(0);}
}
```

- *Suivez l'évolution des valeurs des variables m, n, qu'est ce que vous remarquer ?*

Vous devez remarquez que les adresses des variables sont identiques dans le fils et le père. De même si vous changer une valeur d'une variable dans le fils, la valeur change dans le père.

- *Comment les variables sont communiqués entre le père et le fils. Justifier.*

Le passage des valeurs s'effectue par adresse.

Partie III

La partie III est traitée dans une vidéo (communication entre les processus) afin de mieux expliquer le concept de communication entre les processus.