Université de Rouen
I.U.P. 2
Module : Système

T.D. 5 : Création de processus

1 Création de processus

```
#include <unistd.h>
pid_t fork(void)
```

Permet la création dynamique d'un nouveau processus fils qui s'execute de façon concurrente au processus appelant (père). Les processus P et F sont en tout point semblables sauf en ce qui concerne l'identifiant des processus (pid). La fonction *fork()* retourne alors 2 valeurs différentes :

- 0 pour le fils
- pid du fils pour le père

Important : Le programme executé est identique.Pour effectuer des traitements différents, il est alors nécessaire de faire en fonction du code retour de la fonction *fork()* :

Exemple :

```
main()
{
if (fork() == 0)
/* Traitement associe au fils */
else
/* Traitement associe au pere */
}
```

Le processus fils va hérité de son père les tables de descripteurs, l'espace de données et les principales caractéristiques (uid,gid,...). Attention, c'est une recopie et non un partage des données.

2 Primitives de recouvrement

Recouvre le processus courant par le code passé en paramètre : execl,execv

(p) : La commande est recherché dans la variable PATH

(e) : L'environnement est modifiable

() : La commande doit être passée en aboslu et l'environnement n'est pas modifiable.

```
execl(char * ref,argv[0],argv[1],... ,NULL)
execv(char * ref,char *argv[]) (Le dernier element doit etre NULL)
```

3 Attente de la fin d'un processus

```
#include <sys/types.h>
#include <sys/wait.h>
pid_t wait(int *status)
pid_t waitpid(pid_t pid,int *status,int options)
```

Attente bloquante de la mort de l'un de ses fils. La variable *status* permettra d'identifier la façon dont le fils est mort (niveau d'exit dans le cas d'un exit , numéro du signal dans le cas d'un signal).

4 Exercices

1. Montrer que l'espace de données d'un processus père est recopié (et non partagé) pour le fils
2. Ecrire un programme qui lance x processus qui afficheront leur numéro d'ordre de lancement (Attendre la fin du processus n avant de lancer le processus $n+1$).
3. Ecrire un programme qui execute la commande *ls -l* et écrit ¡¡Merci¿¿ à la fin de la commande. Généraliser ce programme de façon à ce qu'il fonctionne pour toute commande shell.