IUT Le Havre, Programmation Système, Info2#, TP2,

Thème : Recouvrement de processus

Rappel des différents formes de exec : Il y a 6 fonctions commençant par exec, permettent à un processus de charger et d'exécuter un nouveau programme binaire en lui transmettant les paramètres nécessaires.

```
#include <unistd.h>
int execl (const char *path, const char *arg, ...);
int execlp (const char *file, const char *arg, ...);
int execle (const char *path, const char *arg , ..., char * const envp[]);
int execv (const char *path, char *const argv[]);
int execvp (const char *file, char *const argv[]);
int execve (const char *path, char *const argv[], char * const envp[]);
```

Exercice 0 *(extrait de polycopies du cours magistral)* Ecrire un programme qui appelle le compilateur cc pour compiler un fichier (prog.c) et génère un fichier exécutable (prog.exe). Les noms du fichier à compiler et de l'exécutable sont passés en paramètres au programme. Donner :

1. **Une version avec execl**

```
int main(int argc, char *argv[])
{ /* vérification du nombre de paramètres */
if (argc != 3)
{ printf ( "Usage : %s fichier.c fichier.exe \n", argv[0]);
exit(1);
}
   /* appel de execl avec le chemin absolu de la commande cc */
execl("/usr/bin/cc", "cc", argv[1],"-o", argv[2], NULL);
perror("execl");
exit (2);
}
```

2. **Une version avec execlp**

```
int main(int argc, char *argv[])
{ /* vérification du nombre de paramètres */
if (argc != 3)
{ fprintf (stderr, "Usage : %s fichier.c fichier.exe \n",
argv[0]);
exit(1);
}
      /* appel de execlp avec le nom de la commande cc */
execl("cc", "cc", argv[1],"-o", argv[2], NULL);
perror("execlp");
exit (2);}
```

3. Une version avec execv

```
int main(int argc, char *argv[])
{ char * args[5];    /* vérification du nombre de paramètres */
if (argc != 3)
{ fprintf (stderr, "Usage : %s fichier.c fichier.exe \n",
argv[0]);
exit(1);   }
args[0]="cc"; args[1]=argv[1]; args[2]="-o";
args[3]=argv[2]; args[4]=NULL;
execv("cc",agrs);
perror("execv");
exit (2);
}
```

Ces 3 programmes sont enregistrer sous **corton sous info2# sous tp2**. Les télécharges, les analyser, puis exécuter ces programmes, et expliquer le résultat.

Exercice 1

Ecrire un programme qui affiche le contenu du répertoire racine, donner :

1. une version avec la fonction *execl*,
2. une version avec la fonction *execlp*,
3. une version avec la fonction *execv*.

Exercice 2

Ecrire un programme *execmde*.c qui lit et exécute une ligne de commande shell avec ou sans arguments.

Exercice 3

Ecrire un programme qui affiche les caractéristiques suivantes du processus :

- pid
- uid et nom de l'utilisateur correspondant (utiliser les fonctions *getuid* et *getpwuid*)

et demande à l'utilisateur un programme exécutable(avec ou sans paramètres) à exécuter par recouvrement. Dans le cas d'une erreur, le programme affiche le numéro de l'erreur système (avec la variable ***extern int errno***)et le message d'erreur correspond (avec la fonction ***strerror(errno)***) et demande de nouveau le nom de l'exécutable avec ses paramètres. Tester différents exécutables. Faites le test avec le programme affichant les caractéristiques du processus indiquées ci-dessus.

Indication : le programme exécutable et ses paramètres sont séparés par un nombre d'espaces quelconque. Pour récupérer ces différents éléments (tokens) on utilise la fonction **strtok**.

---

fin du tp2

```c
/*  correction tp2 ex2.c   */

#include <stdio.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <pwd.h>

#define Length 100
int main(){
    char com[Length];
    printf("Donner commande à executer : ");
    if(fgets(com,Length,stdin) != NULL){
    com[strlen(com)-1]='\0';
    execlp("sh", "sh", com, NULL);
                }
}
```

```
/*  correction tp2 ex3.c   */
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <pwd.h>
#define Length 100
extern int errno;      /*extern char *sys_errlist[]; dans stdio.h */
int main(){
   char *s;
   char *args[Length];
   char com[Length];
   char rep ;
   int j,i;
   printf("Processus = %d\n", getpid());
   printf("Utilisateur = %d\n", getuid());
   printf("Nom utilisateur : %s\n", getpwuid(getuid())->pw_name);
   while(1){
      printf("Programme executable(e)/commande shell(s) ? : ");
      rep=getchar();
     /* pour vider le tampon */
      if(rep != '\n')
         while ( getchar() != '\n') {} ;
      switch(rep){
         case 'e' : printf("executable avec les paramètres : ");
         if(fgets(com,Length,stdin) != NULL){
               com[strlen(com)-1]='\0'; /* pour supprimer le caractere ' \n' */
               for(j=0, s=com; (s=strtok(s," ")) != NULL; j++, s=NULL)
                     args[j]=s;
               args[j]=NULL;
               execvp(args[0], args);
         }
         break ;
         case 's' : printf("Commande à executer : ");
            if(fgets(com,Length,stdin) != NULL){
               com[strlen(com)-1]='\0';
               execlp("sh","sh",com,NULL);
            }
         break;
         default : exit(1);
      }
      printf("\n Erreur numero = %d\n", errno);
      printf("\n Erreur %s\n", sys_errlist[errno]);   }  }
```