# EXERCICES DIRIGES 3
# Primitives de recouvrement Exec
# CORRECTION

*Préambule : rappel du format des primitives de la famille exec*

## Exercice 1

```
/*************************************************************/
/*   EXERCICE 1                                              */
/*************************************************************/
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

main()
{
pid_t pid;

pid = fork();
if (pid == -1)
      printf ("erreur creation de processus");
else
      if (pid == 0)
      {
        printf (" je suis le fils, mon pid est %d\n", getpid());
        printf(" Le  pid de mon pere est %d\n", getppid());
        execlp("ls", "ls", "-l", "/", NULL);
        }
      else
      {
        printf (" je suis le pere, mon pid est %d\n", getpid());
        printf(" Le  pid de mon fils est %d\n", pid);
        wait ();
        }
```

## Exercice 2

```
/*************************************************/
/*         programme essai_fich.c : processus pere   */
/*************************************************/

#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int i;
main()
{
int pid, fp, j;
char ch[4], conv[11];

fp = open("fichier", O_RDWR);
printf("valeur fp a l'ouverture, %d\n", fp);
```

```
pid = fork();
if (pid==0)
      execlp("lire_fichier","lire_fichier",gcvt((double)fp,10,conv),NULL);

else
      {
      for(j=0;j<3;j++)
      {
      read(fp,ch,4);
      printf("hello pere; %s\n ", ch);
      }
      wait();
      close(fp);
      }

}


/*****************************************/
/* programme lire_fichier.c : processus fils */
/*****************************************/
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

main(argc, argv)
char **argv;
{
int fp, i;
char ch[4];

fp = atoi(argv[1]);
printf("valeur fp fils, %d\n", fp);

      for(i=0;i<4;i++)
      {
      read(fp,ch,4);
      printf("hello fils; %s\n ", ch);
      }
      close(fp);
}
```

**Exercice 3**

Processus shell :

```
for(;;)
{
        lire une ligne de commande sur l'entrée standard;
        analyser la commande lue pour la décomposer en commande, arg1, …, argn;
        if (la ligne de commande contient &)
                ARRIERE_PLAN = VRAI;
        else
                ARRIERE_PLAN = FAUX;
        pid = fork();   /* le shell crée un fils pour exécuter la commande */
        if (pid == 0)
                execlp ("commande", "arg1", .."argn");
```

```
        else
                if (ARRIERE_PLAN == FAUX)
                        wait();    /* le pere attend son fil si celui-ci n'est pas en arriere plan */
}
```