STI 1ère année – Programmation Système

TD 6: processus

J. Briffaut

1 Processus

L'objectif de ce TD est d'éudier :
- la création de processus
- la gestion de l'accès à un fichier par différents processus
- la terminaison de processus

1.1 Création de processus

Exercice 1 Ecrire un programme qui crée un nouveau processus via la primitive `fork()` et affiche les différentes informations relatives suivantes :
- PID du processus père et fils
- ordre d'apparition des processus père et fils
- ordre de disparition de ces processus

Listing 1 – Solution de l'exercice 1

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <assert.h>


int main() {
   pid_t status;
   printf("[%d] Je vais engendrer\n", getpid());
   status = fork();
   switch (status) {
      case -1 :
         perror("Creation processus");
         exit(EXIT_FAILURE);
      case 0 :
         printf("[%d] Je viens de naitre\n", getpid());
         printf("[%d] Mon pere est %d\n", getpid(), getppid
              ());
         break;
      default:
         printf("[%d] J'ai engendre\n", getpid());
         printf("[%d] Mon fils est %d\n", getpid(), status);
   }
   printf("[%d] Je termine\n", getpid());
   exit(EXIT_SUCCESS);
}
```

1.2 Processus et accès concurrent à un fichier

Exercice 2 Écrire un programme qui ouvre un fichier en lecture/écrire puis crée un nouveau processus fils. Le processus fils réalisera alors :
- Une écriture de la chaine "foo" à l'offset 3 dans ce fichier
- une pause de 2 secondes via `sleep`
- puis relira et affichera 3 caractères à l'offset 3

Le processus père réalisera alors :
- Une écriture de la chaine "bar" à l'offset 3 dans ce fichier
- une pause de 1 seconde via `sleep`
- puis relira et affichera 3 caractères à l'offset 3

Que ce passe t'il ?

Listing 2 – Solution de l'exercice 2

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <assert.h>

int main(int argc, char *argv[])
{
   int fd;
   char buf[10];
   fd = open(argv[1], O_RDWR);
   assert(fd != -1);
   read(fd, buf, 2);
   switch (fork()) {
      case -1 :
         perror("Creation processus");
         exit(EXIT_FAILURE);
      case 0 :
         write(fd, "foo", 3);
         sleep(2);
         read(fd, buf, 3);
         buf[3]='\0';
         printf("[%d] fils a lu '%s'\n", getpid(), buf);
               break;
      default:
         write(fd, "bar", 3);
         sleep(1);
         read(fd, buf, 3); buf[3]='\0';
         printf("[%d] pere a lu '%s'\n", getpid(), buf);
   }
   close(fd);
   exit(EXIT_SUCCESS);
}
```

Exercice 3 Proposez une solution pour palier ce problème.

1.3 Terminaison de processus

Exercice 4 Écrire un programme qui crée un nouveau processus. Le processus fils attendra 2 secondes avant de quitter. Le processus père attendra que le processus fils est fini son traitement avant d'afficher le code de retour du processus fils.

Listing 3 – Solution de l'exercice 4

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <assert.h>


int main(int argc, char *argv[])
{
   int status;
   pid_t pid, pidz;
   switch (pid = fork()) {
      case -1 :
         perror("Creation processus");
         exit(EXIT_FAILURE);
      case 0 :
         printf("[%d] fils eclair\n", getpid());
         sleep(2);
         exit(EXIT_SUCCESS);
         break;
      default:
         printf("[%d] pere a cree %d\n", getpid(), pid);
         pidz = wait(&status);
         if (WIFEXITED(status))
            printf("[%d] mon fils %d a termine normalement,\
                  n"
                        "[%d] code de retour: %d\n",
                        getpid(), pidz,
                        getpid(), WEXITSTATUS(status));
         else
            printf("[%d] mon fils a termine anormlement\n",
                  getpid());
   }
   exit(EXIT_SUCCESS);
}
```