TD 4

Communications entre Processus

Tubes classiques

Objectifs du TD :

L'objectif principal de ce TD est de comprendre et maitriser l'utilisation des fonctions ***pipe(), fork()***, ***wait()***, ***exit()***, etc pour pouvoir faire communiquer des processus entre eux. Dans ce TD, les processus font partie de la même famille.

Pour rappel :

- ***pipe()*** crée un tube de communication entre un processus père et ses fils.
- ***fork()*** crée un nouveau processus, fils du processus en exécution.
- ***wait()*** avec ***exit()*** permettent de synchroniser les processus père et fils. Ces fonctions permettent au programmeur de mettre en attente un processus père jusqu'à la terminaison d'un processus fils.

Dans l'Annexe ci-joint (pages 3-4), on considère un extrait d'un programme C. Ce programme est constitué des étapes suivantes :

a) Création de tubes
b) Création de fils
c) Lecture d'une phrase (ou une chaine de caractères) sur le clavier
d) Transmission de cette phrase à un groupe de processus (***Groupe1***), en utilisant un premier tube
e) Transmission de cette phrase des processus du groupe Groupe1 vers d'autres processus qui appartiennent au groupe « ***Groupe2*** », en utilisant un deuxième tube
f) Assemblage, par les processus du groupe Groupe2, des caractères et affichage sur l'écran.

Répondre aux questions suivantes :

1) Combien de processus fils y a-t-il dans le programme ?
2) Quel est la tâche (ou le rôle) du processus père ?
3) Quelle est la tâche de chaque processus fils ?
4) Est-ce que tous les processus fils sont identiques ? Justifiez votre réponse.

5) Quelle valeur prend etat[ordre], pour ordre =0, 1, 2, 3 ?
6) Compléter le tableau suivant tout en tenant compte de toutes les étapes (**a-f**) mentionnées ci-dessus.
7) Comment peut-on modifier le programme dans l'annexe pour qu'on puisse réaliser les tâches des fils en exécutant (chargeant à partir du disque) d'autres programmes ?

1		**2**		**3**	
4		**5**		**6**	
7		**8**		**9**	
10		**11**		**12**	
13		**14**		**15**	
16		**17**		**18**	

Annexe A

```
#include <sys/types.h>
#include <stdio.h>
#include <errno.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>


#define       NbOfChild  2
#define       MOD        5


int main (int nbarg, char *tbarg[]) {
      int idpro, tubeA[2],tubeB[2], nbcarlus, attente, etat[2*NbOfChild], ordre,
cptcar=0;
      char phrase[256],carlu;
      ssize_t taille;
      size_t tmax=255;

      srand(getpid());

      pipe(-1-);
      pipe(-2-);

      for (ordre=1; ordre<=NbOfChild; ordre++) {
            if ((idpro=fork ()) < 0) {
                  perror("creation de processus");
                  exit(errno);
            }

            if (idpro == 0) {//Groupe1
                  cptcar=0;
                  close(-3-);
                  close(-4-);
                  nbcarlus= read(-5-,&carlu,1);
                  while (nbcarlus > 0) {
                        int nbecr;
                        cptcar ++;
                        attente = rand()%MOD;
                        sleep(attente);
                        nbecr=write(-6-,&carlu, 1);
                        nbcarlus= read(-7-,&carlu,1);
                  }
                  close(-8-);
                  close(-9-);
                  printf ("\t Je suis le fils %d, nbcarlus : %d et je me termine
\n", ordre, cptcar);
                  exit(0);
            }
      }
```

```
for (ordre=1; ordre<=NbOfChild; ordre++) {

            if ((idpro=fork ()) < 0) {
                  perror("creation de processus");
                  exit(errno);
            }

            if (idpro == 0) {//Groupe2
                  char chaine[256];
                  cptcar=0;
                  close(-10-);
                  close(-11-);
                  close(-12-);
                  nbcarlus=read(-13-,&carlu,1);
                  chaine[cptcar]=carlu;
                  while(nbcarlus > 0) {
                        cptcar++;
                        nbcarlus= read(-14-,&carlu,1);
                        chaine[cptcar]= carlu;
                  }
                  chaine[cptcar] = 0;
                  printf ("\n Phrase obtenue par le fils %d est : %s , nb car :
%d\n", ordre, chaine, cptcar);
                  close(-15-);
                  exit(1);
            }
      }


    close(-16-);
    close(-17-);
    close(-18-);
    strcpy(phrase, tbarg[1]);
    taille=strlen(phrase);

    while (cptcar < taille) {
          nbcarlus= write(tubeA[1] ,&(phrase[cptcar]),1);
         cptcar++;
    }
    close(tubeA[1]);

    for (ordre=1; ordre <= 2*NbOfChild; ordre++)
            idpro=wait(&etat[ordre]);

      printf("programme termine\n");

  /*Fin du programme*/
}
```