TD-TP de Système n° 11

Exercice 1 : tubes nommés et situations d'interblocage

Expliquer ce qui se passe lors de l'exécution des programmes ci-après. Que faut-il faire pour débloquer la situation ?

1. ```
   #include <sys/types.h>
   #include <sys/stat.h>
   #include <fcntl.h>
   #include <stdio.h>
   #include <unistd.h>
   #include <stdlib.h>

   int main(void){
     int tube[2], lus;
     const char s[] = "je communique";
     char str[20];

     if(mkfifo("mon_tube",0777) == -1)
       perror("open(mon_tube)");

     if((tube[0] = open("mon_tube", O_RDONLY)) == -1)
       abort();
     if((tube[1] = open("mon_tube", O_WRONLY)) == -1)
       abort();

     lus = write(tube[1], s, sizeof(s));
     if(lus < 0)
       abort();
     read(tube[0], str, sizeof(s));
     printf("%s\n", str);

     close(tube[0]);
     close(tube[1]);
     return 0;
   }
   ```
   ```

2.
```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>

int main(void){
  int fdLect, fdEcri;

  if(mkfifo("fifo1",0777) == -1)
    perror("open(fifo1)");

  if(mkfifo("fifo2",0777) == -1)
    perror("open(fifo2)");

  if(fork() == 0){
    printf("fils lance\n");
    if( (fdEcri = open("fifo1", O_WRONLY)) == -1)
      abort();
    printf("fils apres 1er open\n");
    if( (fdLect = open("fifo2", O_RDONLY)) == -1)
      abort();
    printf("fils apres 2eme open\n");
  }
  else{
    printf("pere lance\n");
    if( (fdEcri = open("fifo2", O_WRONLY)) == -1)
      abort();
    printf("pere apres 1er open\n");
    if( (fdLect = open("fifo1", O_RDONLY)) == -1)
      abort();
    printf("pere apres 2eme open\n");
  }
  return 0;
}
```

Exercice 2 :

1. Écrire un programme qui crée deux tubes nommés `question` et `reponse` ; chaque fois qu'il recoit un mot sur le tube `question`, il inverse ses lettres et les renvoie sur le tube `answer`. On considère que le programme ne s'arrête jamais.
2. Écrire un programme qui prend une phrase en argument, envoie tous ses mots au serveur écrit ci-dessus et affiche les mots renvoyés par le serveur.

Exercice 3 : tubes nommés

Écrire un programme qui prend une option `mode` et un argument `fichier` et qui agit selon la valeur de `mode` :

- `c` : crée un tube nommé `fichier` avec des droits en lecture et écriture pour tout le monde.
- `r` : ouvre en lecture le fichier `fichier` et affiche son contenu sur la sortie standard.
- `w` : ouvre en écriture le fichier `fichier` et y copie l'entrée standard.

Écrire le programme en version bloquante (synchronisée), puis en version non bloquante. Dans chacun des cas, quel est l'ordre dans lequel il faut lancer les différents modes ? Quel est l'effet de l'interruption d'un processus en mode lecture (resp. en mode écriture) ?

Exercice 4 :

Écrire deux programmes `eleve` et `prof` qui permettent à des élèves de communiquer avec leur enseignant pour lui poser des questions de cours et recevoir des réponses. Pour envoyer leur questions, les élèves utiliseront le tube `/tmp/<nom>_lect.tmp`, par lequel l'enseignant pourra les lire. Symétriquement, pour lire les réponses, les élèves utiliseront le tube `/tmp/<nom>_ecr.tmp`, dans lequel l'enseignant les aura écrites. La chaîne `<nom>` identifiant les deux tubes sera donnée en argument aux deux programmes. L'enseignant est le seul qui pourra activer ces tubes, qui seront désactivés dès que l'étudiant rentre "bye" comme question.

Que se passe-t-il si plusieurs élèves sont en train d'utiliser les deux mêmes tubes et si l'un d'entre eux signale la fin de ses questions par "bye" ?

Que se passe-t-il si plusieurs élèves sont en train d'utiliser les deux mêmes tubes et si l'un d'entre eux soumet une question avant que l'enseignant ait eu le temps de répondre à une question posée précédemment ?

Ces problèmes peuvent être resolus si tous les élèves utilisent le même tube pour poser les questions (appelé par exemple `/tmp/<nom>_public.tmp`, où la chaîne `<nom>` sera donnée en argument), mais chacun dispose d'un tube dédié pour recevoir ses réponses.

Proposer un protocole de communication entre l'enseignant et les élèves adapté à ce modèle et écrire le code des programmes `eleve_2` et `prof2` qui permettent un telle communication.

Pour simplicité, on choisira de limiter à 10 le nombre maximal d'élèves qui peuvent poser des questions simultanément. La session de questions/réponses se terminera automatiquement dès qu'il n'y a plus d'élèves connectés.