Système d'Exploitation Unix

TP

Les appels système : communication par tubes

Les exercices suivants ont pour but de vous familiariser avec les appels système de base relatifs à la communication par tube.

Il vous est recommandé de consulter les pages man de ces primitives pour de plus amples informations sur leur syntaxe, leur sémantique et les éventuelles options qu'elles offrent.

1) Communication par tubes ordinaires

Les tubes ordinaires permettent la communication entre processus de même filiation.

a) Ecrire un programme qui permet à un processus de communiquer des données, entrées par l'utilisateur, à un processus fils via des tubes ordinaires. Le fils affiche les données transmises par son père.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define MAX 25

int p[2], P1,lu,ecrit;
char buff[MAX];

int main(int argc, char *argv[]) {
 if (argc !=2) {
    fprintf (stderr, "%s: parametre incorrect\n", argv[0]);
    exit (EXIT_FAILURE);
 }
 pipe(p);

 P1=fork();
 if (P1== 0) { // Fils
    close (p[1]);

    lu=read (p[0], buff, MAX);
    if (lu<0){
       perror ("erreur read");
       exit (EXIT_FAILURE);
    }
    buff[lu]='\0';
    printf ("fils: %s\n", buff);
    close (p[0]);
    exit(EXIT_SUCCESS);
 }
// Pere
 close (p[0]);
 ecrit=write(p[1], argv[1], strlen(argv[1]));
 if (ecrit<0){
    perror ("erreur write");
    exit (EXIT_FAILURE);
 }
 close (p[1]);
 exit(EXIT_SUCCESS);
}
```

b) Modifier le programme précédent de façon que les deux processus communiquent dans les deux sens. Utiliser deux tubes et chaque processus doit garder ouvert l'ensemble des descripteurs de tubes.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define MAX 25

int t1[2], t2[2], P1,lu,ecrit;
char buff[MAX];

int main(int argc, char *argv[]) {
 if (argc !=2) {
    fprintf (stderr, "%s: parametre incorrect\n", argv[0]);
    exit (EXIT_FAILURE);
 }
 pipe(t1);
 pipe(t2);
 P1=fork();
 if (P1== 0) { // Fils
    close (t1[1]);
    close (t2[0]);

    lu=read (t1[0], buff, MAX);
    if (lu<0){
       perror ("erreur read");
       exit (EXIT_FAILURE);
    }
    buff[lu]='\0';
    printf ("fils, je reçois: %s\n", buff);
    ecrit=write(t2[1], argv[1], strlen(argv[1]));
    if (ecrit<0){
          perror ("erreur write");
          exit (EXIT_FAILURE);
    }
    close (t1[0]);
    exit(EXIT_SUCCESS);
```

```
 }
// Pere
 close (t1[0]);
 close (t2[1]);
 ecrit=write(t1[1], argv[1], strlen(argv[1]));
 if (ecrit<0){
    perror ("erreur write");
    exit (EXIT_FAILURE);
 }
lu=read (t2[0], buff, MAX);
if (lu<0){
    perror ("erreur read");
    exit (EXIT_FAILURE);
}
buff[lu]='\0';
printf ("père, je recois: %s\n", buff);
close (t1[1]);
close (t2[0]);

 exit(EXIT_SUCCESS);
}
```

c) Quel comportement obtenez-vous de l'exécution du programme précédent si les deux processus commencent chacun par une lecture ?

Quelles solutions à ce problème ?

a. Eviter cette situation !!

b. Utiliser le E/S en mode non bloquant

d) Envoyer un signal au processus fils. Que constatez-vous ? Expliquer ce qui se passe.

Le père se termine car il reçoit le signal SIGPIPE lorsqu'il écrit dans le tube vers son fils

2) Communication par tubes nommés

Les tubes nommés permettent la communication entre processus, même sans lien de parenté.

a) Ecrire un programme qui permet à deux processus sans lien de parenté d'échanger des données, entrées par l'utilisateur, via un tube nommé. Chaque processus affiche les données reçues.

Le tube utilisé sera créé au préalable avec la commande `mkfifo`

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



```
int main(int argc, char *argv[]){
    int h1;

    if (argc !=3) {
        fprintf (stderr, "%s: tube mot\n", argv[0]);
        exit (EXIT_FAILURE);
    }

    h1=open(argv[1], O_WRONLY);
    printf("%d: j'ecris \"%s\" dans le tube\n", getpid(), argv[2]);
    write(h1, argv[2],strlen(argv[2]));
    close(h1);
    exit(0);
}

#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <stdlib.h>
#include <stdio.h>
#define TAILLE 16

int main(int argc, char *argv[]){
    int h1, nread;
    char msg[TAILLE];

    if (argc !=2) {
        fprintf (stderr, "%s: tube\n", argv[0]);
        exit (EXIT_FAILURE);
    }
    h1=open(argv[1], O_RDONLY);
    nread=read(h1,&msg,TAILLE);
    msg[nread]='\0';
    printf("%d: je lis \"%s\" depuis le tube\n", getpid(), msg);

    close(h1);
    exit(0);
}
```