TD UNIX
PIPE
INFO - 2éme année

Notions étudiées : La communication par pipes et la synchronisation par signaux .

Les "PIPES"

Le "pipe" est le moyen classique offert par UNIX pour permettre à deux processus de communiquer entre eux.

Description

Un pipe est un fichier particulier. Il ne possède pas de nom (il n'appartient à aucune directory). Il est connu par deux descripteurs: un pour la lecture, un pour l'écriture.

Ses pointeurs courants de lecture et d'écriture ne sont pas accessibles à l'utilisateur.

Ce fichier est géré par le noyau comme une file FIFO (First Input, First Output). Chaque demande de lecture provoque un positionnement sur le premier caractère non lu, chaque demande d'écriture provoque un positionnement en fin de fichier.

Pour faire communiquer deux processus à l'aide d'un pipe, on effectue les actions suivantes:

1/ Le processus père ouvre un pipe (à l'aide de la fonction "pipe").

2/ Il crée un processus fils (à l'aide de "fork").

3/ A cet instant, les descripteurs du pipe sont utilisables par chacun des deux processus car les fichiers ouverts par le processus père sont transmis au processus fils.

4/ Pour envoyer un message au fils, le père écrit dans le pipe, comme il écrirait dans un fichier quelconque (avec "write").

5/ Pour lire le message du père, le fils lit dans le pipe (à l'aide d'un "read").

Utilisation

Les pipes se manipulent avec les primitives de bas niveau d'accès aux fichiers. Avec malgré tout, les restrictions suivantes:

- L'ouverture d'un pipe se fait à l'aide de la fonction "pipe" et non "open".

- La fonction de positionnement dans un fichier ("lseek") ne peut pas être utilisée.

- Le pipe a une taille limitée. Un processus essayant d'écrire dans un pipe plein sera bloqué (cad qu'il ne sortira pas de la fonction d'écriture) tant qu'une partie de ce pipe n'aura pas été lue pour refaire de la place.

- Un processus demandant à lire dans un pipe ne contenant pas le nombre suffisant d'octets sera également bloqué tant que le nombre d'octets souhaités ne sera pas présent dans le pipe.

Cette dernière restriction est très importante car elle peut mener à un blocage. Dans ce cas seul un signal peut faire changer d'état le processus.

Remarque

Un pipe est ouvert par un processus et est utilisable par tous les fils de ce processus. N'importe lequel de ces processus peut écrire des données dans le pipe et n'importe lequel de ces processus peut en lire. Une donnée étant écrite, c'est le premier processus qui lit le pipe qui va la récupérer. Ceci signifie que si un processus écrit dans le pipe puis immédiatement se met en

lecture, il va lire les données qu'il a écrites. Aussi pour éviter toute confusion, il est conseillé de fermer les descripteurs de pipe que l'on n'utilise pas.

Fonction "pipe"

Cette fonction permet de créer un pipe. Elle retourne les descripteurs du pipe.

int pipe(fildes)

int fildes[2]

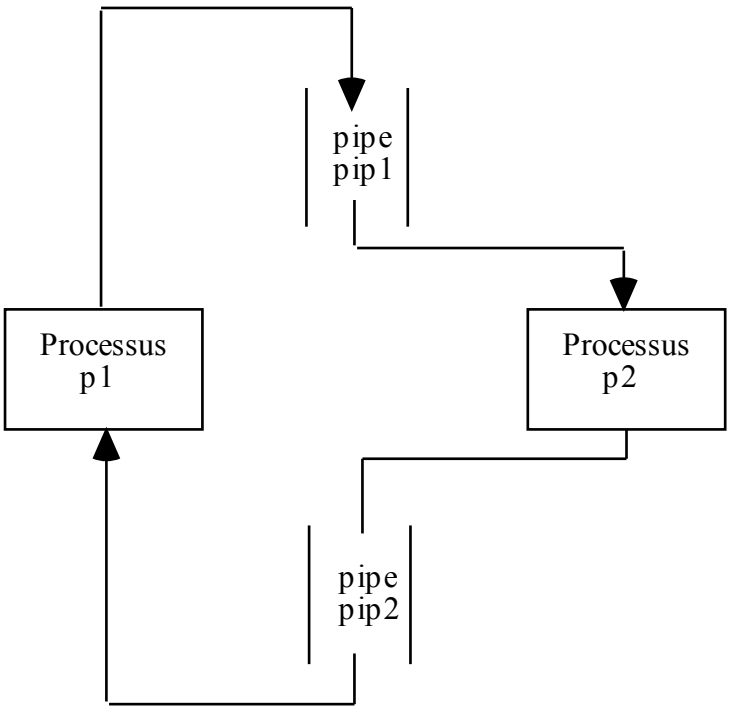
Un pipe est créé et on récupère dans un tableau de deux entiers, les descripteurs qui lui sont associés. Le descripteur fildes[0] est celui qui doit être utilisé pour les lectures. Le descripteur fildes[1] est celui qui doit être utilisé pour les écritures.

La valeur en retour de la fonction est zéro (état normal) ou -1 si le pipe n'a pu être créé (par exemple si trop de fichiers sont ouvert par le processus).

Exercice 1

On désire faire communiquer deux processus "p1" et "p2" (par un système de question/reponse) à travers deux "pipes".

Le schéma désiré est le suivant:



Il faudra donc ouvrir un pipe "pip1" pour les liaisons de "p1" vers "p2" ("p1" devra donc fermer le descripteur de lecture de "pip1" et "p2" le descripteur d'écriture de ce pipe) puis un pipe "pip2" de "p2" vers "p1" ("p1" devra donc fermer le descripteur d'écriture de "pip2" et "p1" ferme le descripteur de lecture de "pip2"). Ecrivez un (et un seul) programme permettant de recréer ce mécanisme formé de 2 processus échangeant des messages au travers de 2 pipes. Chaque processus désirant envoyer un message à l'autre dépose son message dans le pipe. Les messages sont dans des tableaux à deux dimensions (la correspondance question/réponse est réalisée à l'aide de l'indice de ligne).

Exercice 2

Reprenez l'exercice précédent et rajouter la fonctionnalité suivante: chaque processus producteur dépose son message dans le pipe et prévient le consommateur par l'envoi d'un signal, ce n'est qu'à la réception du signal que le consommateur lit le message.

Exemple de programme

```
/*                                                      */
/*          Utilisation de PIPE inter processus         */
/*                                                      */
#include <stdio.h>
int pip[2];         /*          descripteur de pipe     */

main()
{
if(pipe(pip))       /*          Ouverture d'un pipe     */
{ perror("Erreur de pipe"); exit(1);}
printf("Dernier message avant fork\n");
switch(fork())      /*          Creation d'un processus */
{
  case -1 : perror("Erreur de fork"); exit(1);
  case 0 : fils();break;
  default :pere();
}
printf("\nFin programme");
}
/*                                                  */
/*          Le fils lit dans le pipe                */
/*                                                  */
fils()
{
char buffer[15];
int i;
for(i=0;i<10;i++)
{
  if( read(pip[0],buffer,15) != 15)
        { perror("Erreur read");
        exit(1);
        }
printf("\nlu: %s", buffer);
}
return;
}
/*                                                  */
/*          Le pere ecrit dans le pipe              */
/*                                                  */
pere()
{
int i;
char buf[15];
strcpy(buf,"Hello World  ")
for(i=0;i<10;i++)
{
  buf[13]=i+48;
  buf[14]=0;
  if( write(pip[1],buf,15) != 15)
        { perror("Erreur write");
        exit(1);
        }
}
return;

}
```