IUT de Blagnac

Iulian Ober
Patricia Stolf

TP 5 (4,5h) : Communication entre processus

1. Gestion des processus par envoi de signaux depuis le shell

En mode ligne de commande, la commande **kill -n PID** permet d'envoyer le signal numéro n au processus de numéro PID. La commande kill -l permet d'afficher la liste des signaux disponibles et leurs numéros. Des explications sur les types de signaux sont données dans la page de manuel de **signal**.

Pour les exercices qui suivent, utiliser le programme suivant :

```
int main(){
  write(2,"Attention : ce programme boucle indéfiniement!\n\n",49);
  while (1){
    write(1,".",1);
    sleep(1) ;
  }
  return 0; /* jamais atteint */
}
```

a) Faites quelques tests pour utiliser :
   - la séquence de touches CTRL+C qui permet d'envoyer le signal SIGINT à un processus en avant plan.
   - La séquence de touches CTRL+Z qui permet d'envoyer le signal SIGSTOP à un processus en avant plan.
   - Les commandes **fg** et **bg** permettent d'envoyer le signal SIGCONT à un processus suspendu.

b) A partir d'un autre terminal, à l'aide de la commande **kill**, vérifier que la plupart des signaux envoyés au processus provoquent sa terminaison (relancer le processus à chaque fois).

c) A l'aide de la commande **kill**, envoyer au processus le signal SIGSTOP, contrôler son état (**ps**), puis lui envoyer le signal SIGCONT.

2. Réception et gestion des signaux par les processus

En faisant appel à la fonction **signal** (voir man) écrire un programme C dont le comportement à la réception du signal SIGINT (CTRL+C) soit le suivant :

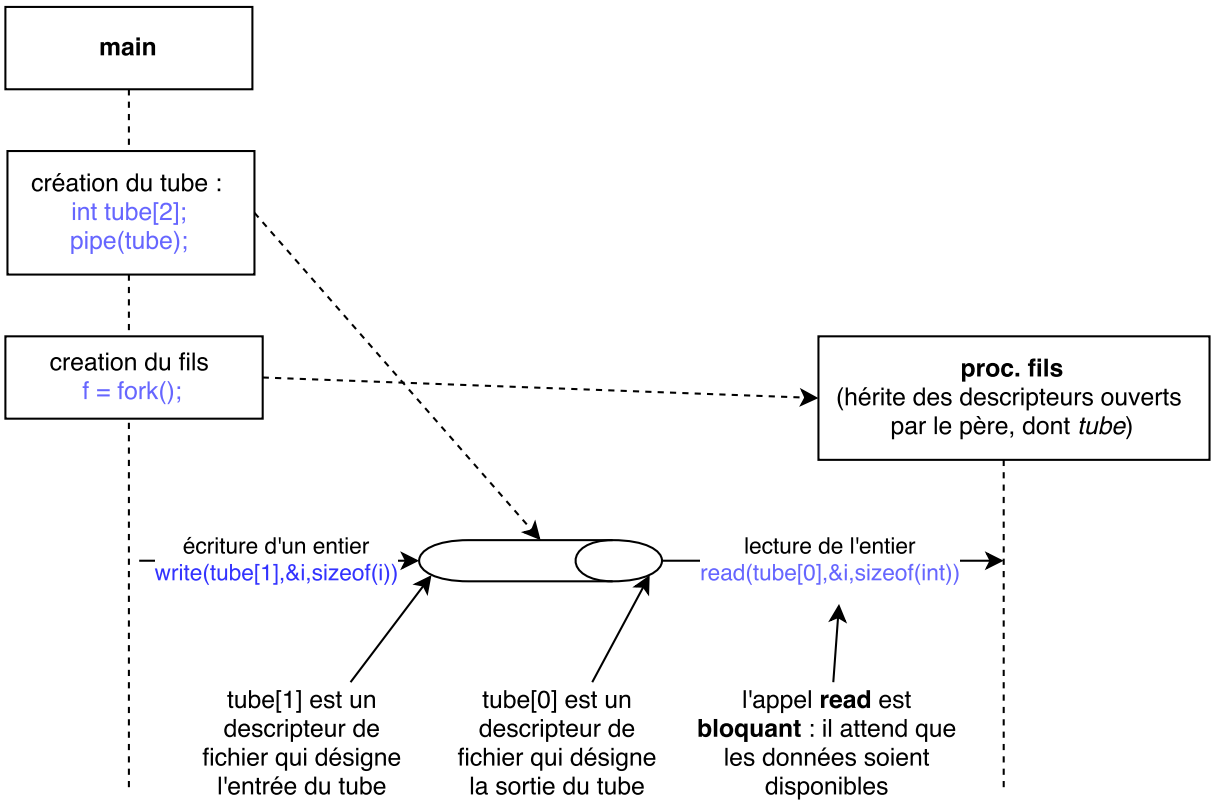
1. A la première réception de 'CTRL+C' : on affichera "interruption numéro 1".
2. A la seconde réception de 'CTRL+C' : on affichera "interruption numéro 2".
3. A la troisième réception de 'CTRL+C' : on affichera "interruption numéro 3".
4. A la quatrième : rien.
5. A la cinquième : rien.
6. Enfin à la sixième : on terminera le processus.

3. Communication par tube anonyme

Écrire un programme C qui crée un tube de communication (avec **pipe**) puis crée un processus fils et lui communique un entier (choisi aléatoirement[1]) à travers le tube. Le processus fils lit l'entier depuis le tube et l'écrit à la sortie standard.

*A noter : La fonction **pipe** crée un tube permettant de communiquer des données entre processus. Le tube est "anonyme" dans le sens où seul le processus qui le crée et ses descendants (créés avec **fork** après l'appel à **pipe**) peuvent y accéder.*

Le schéma suivant décrit la solution et les principaux pas exécutés par les deux processus :



[1] Pour générer des nombres aléatoires, vous pouvez utiliser les fonctions **srand** et **rand**, ou bien ouvrir (**open**) le fichier **/dev/urandom** et lire (**read**) des nombres dedans.

IUT de Blagnac

Iulian Ober
Patricia Stolf

4. Substitution de descripteurs de fichiers

Écrire un programme C qui

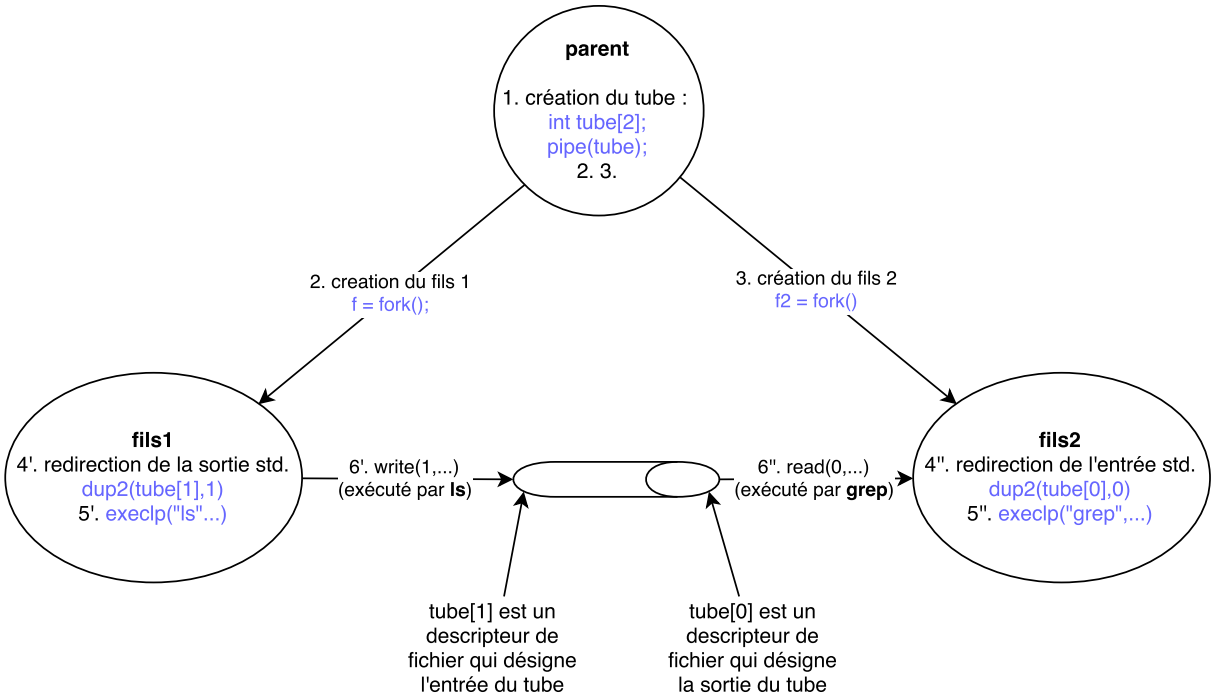
- Ouvre un fichier « *f* » en écriture
- Substitue le descripteur de « *f* » au descripteur de sortie standard (voir fonctions **dup** et/ou **dup2**)
- Lance avec **exec** la commande **ls –l**

Exécuter le programme, examiner le contenu de « f » et expliquer le résultat.

5. Combiner pipe et dup pour créer un tube entre deux commandes

Écrire un programme C qui crée un tube de communication (avec **pipe**) puis crée 2 processus fils, le premier exécutant la commande **ls -l**, le second exécutant la commande **grep \.c**. Le tube doit préalablement être raccordé (avec **dup2**) de manière à reproduire l'équivalent à la commande shell **ls -l | grep \.c**.

Le schéma suivant décrit la solution et les principaux pas exécutés par les trois processus. Ce schéma n'est cependant pas complet, notamment il faudra fermer les extrémités du tube inutilisées dans chacun des trois processus pour que tout fonctionne bien…



6. Communication par tube nommé

Essayer la séquence de commandes shell suivante:

```
mkfifo tmp
ls -l tmp
cat tmp
```

Dans un autre terminal, exécuter la commande "`cat > tmp`" et saisir quelques lignes de texte. Regarder le premier terminal et expliquer le résultat (en faisant appel à "**man mkfifo**").

Écrire deux programmes en C qui communiquent par le tube "tmp" crée précédemment. Le premier programme doit écrire dans le tube des nombres aléatoires entre 32 et 99 (sous forme de texte en décimal sur 2 caractères). Le deuxième programme doit récupérer les nombres et écrire sur la sortie standard les caractères ASCII correspondants aux codes numériques reçus.

Remarque : vous pouvez utiliser la fonction **sscanf** pour la conversion de la chaine de caractères en nombre. Aussi, pour rappel, un *char* c'est essentiellement un *entier* sur 1 octet et la question « comment convertir un code ASCII en caractère ? » n'a pas de sens.