TD Programmation système et réseaux
No 6

Mai 2003

Rappel

- création d'un pipe anonyme : int pipe (int filedes[2]);
  - crée une paire de descripteurs de fichiers, pointant sur un i-noeud de tube, et les place dans un tableau `filedes`.
  - `filedes[0]` est utilisé pour la lecture, et `filedes[1]` pour l'écriture.
    - ATTENTION : les lectures et écritures sont bloquantes par défaut
  - rend -1 en cas d'erreur.
- création d'un pipe nommé int mkfifo (const char *pathname, mode_t mode);
  - crée un fichier spécial FIFO (tube nommé) à l'emplacement `pathname`. `mode` indique les permissions d'accès. Ces permissions sont modifiées par la valeur d'umask du processus : les permissions d'accès effectivement adoptées sont (mode & ~umask).
  - il faut ouvrir les deux extrémités simultanément avant de pouvoir effectuer une opération d'écriture ou de lecture.
  - rend -1 en cas d'échec
- positionnement des droits *a priori* lors de la création de fichiers int umask(int mask);
  - fixe le masque de création de fichiers à la valeur `mask` & 0666 et `mask` & 0777 pour les répertoires
    - ATTENTION : umask "enlève" des droits
  - la commande n'échoue jamais
  - renvoie la valeur précédente du masque
- dupplication de descripteurs de fichiers int dup2(int oldfd, int newfd);
  - met une copie du descripteur de fichier `oldfd` à la place de `newfd`
  - renvoie le nouveau descripteur
  - `newfd` est fermé auparavant si besoin est.
  - rend -1 en cas d'échec

Exercices

1 - Soit le programme

```
void J_ecris(int out_pipe){
  char Buff[]="Salut, c'est ton fils qui vient demander son argent de poche";
  write(out_pipe, Buff, strlen(Buff)+1);
  close(out_pipe);
}
void Je_lis(int in_pipe){
  char Buff;
  while (read(in_pipe,&Buff, 1) > 0)printf("%c", Buff);
  printf("\n");
}
main(){
  pid_t Pid;
  int fd[2];
  pipe(fd);
  Pid = fork();
  if ( Pid == 0) {
    close(fd[0]);
    J_ecris(fd[1]);
  }
  else {
    close(fd[1]);
    Je_lis(fd[0]);
  }
}
```

1.1 - Expliquer ce que fait ce programme

1.2 - Ecrire un programme qui permette au père de répondre "Salut, c'est ton père qui répond NON". Expliquer pourquoi il est difficile d'utiliser un seul pipe.

1.3 - Utiliser un ou plusieurs pipes nommés pour réaliser le même dialogue mais sans que les deux processus soient père/fils.

2 - Dans la cas de l'exercice n°3 "Fork et fichier" de la feuille N°5, proposer une solution dans laquelle les processus ne font pas d'attente active.

Rappel de l'énoncé : Ecrire un programme qui crée le fichier `Res.txt` puis crée deux fils. Ces deux fils doivent écrire 5 '`A`' et 5 '`B`' dans le fichier `Res.txt` avec la contrainte que le premier fils ne peut qu'écrire des '`A`' dans ce fichier et le deuxième fils que des '`B`'. Enfin, le processus père doit afficher le fichier `Res.txt`.

Le fichier doit contenir exactement `"ABABABABAB"`

3 - Ecrire un programme C qui simule l'exécution de `ls -il | sort` en utilisant les commandes système `ls` et `sort` sans `|`.

Vous devez utiliser `pipe` et `dup2`.