Tubes (pipes)

SYE

6 - Descripteur Fichier et Tubes

Résumé du document

Ce document parle de l'utilisation des tubes (ou pipes) pour la communication entre processus, en particulier les tubes anonymes et nommés. Les tubes anonymes, créés avec pipe(), facilitent l'échange entre processus parent et enfant, tandis que les tubes nommés, créés avec mkfifo(), permettent à plusieurs processus de se connecter. L'appel dup2() est utilisé pour rediriger les flux standard, comme dans la commande ls | more.

Table des matières

1. Tubes

Les tubes ou plutôt pipes servent principalement à transferer rapidement des messages courts entre processus et permettent une lecture/écriture de type **FIFO**.

Un message déposé dans un pipe restera présent tant qu'aucun processus n'autra effectué la lecture. Celui-ci sera automatiquement retiré du tube aussitôt que le processus consommateur aura effectué la lecture.

1.1. Capcités

Les pipes ont une capacité limitée, ils peuvent donc se trouver dans un état vide ou plein. Dans les deux cas, leur comportement par défaut consistera:

- **suspendre** le processus qui tentera d'effectuer une elcture d'un **tube vide**
- mettre le processus en **waiting** lors-ce qu'un processus veut écrire dans un pipe plein

1.2. Structure des pipes

Un pipe possède deux descripteurs de fichier car il possède deux extrémités.

- un descripteur pour écrire dans le tube
- un autre descripteur pour lire le tube

```c
/* includes ... */
int main(int argc, char *argv[]) {
  int pipe_fd[2];     //Descripteurs d'entrée et sortie
  int ret;

  pipe(pipe_fd);

  ret = fork();
  ...
}
```

- La création d'un tube anonyme se fait via l'appel système `pipe()`, qui retourne un tableau de deux descripteurs (entiers).
- Selon POSIX, le descripteur `pipe_fd[0]` est réservé à la lecture, tandis que `pipe_fd[1]` est destiné à l'écriture.
- Par analogie, le descripteur 0 est l'entrée standard (stdin) et le descripteur 1 est la sortie standard (stdout).
- Lorsqu'un appel à `fork()` suit l'appel à `pipe()`, le processus fils hérite des descripteurs, permettant une communication entre le parent et l'enfant via le même tube.
- Dans chaque processus, l'extrémité du tube non utilisée doit être fermée pour éviter des fuites de ressources.

1.3. Tube nommé

- crée avec la fonction `mkfifo()` → utilise l'appel système `open()`
  - apparaît comme un fichier virtuel
- plusieurs processus peuvent se connecter au même tube
- persistent au niveau du système de fichier et doit être explicitement détruit

1.3.1. Connexion

- Le processus qui crée le tube utilise `open()` pour se connecter en écriture.
- Un autre processus peut se connecter en lecture.

Exemple de code pour un tube nommé

```
// Code du processus d'écriture
int fd_pipe = open("essai.fifo", O_WRONLY);
write(fd_pipe, "Bonjour", 8);
close(fd_pipe);

// Code du processus de lecture
int fd_pipe = open("essai.fifo", O_RDONLY);
char buffer[80];
read(fd_pipe, buffer, 80);
printf("%s\n", buffer);
close(fd_pipe);
```

1.4. Tube annonyme

- Les tubes anonymes facilitent la communication entre un processus parent et ses enfants
- Exemple typique : la commande `ls | more` (ou `dir | more` sous Windows) redirige la sortie de `ls` vers l'entrée de `more`.
- Le symbole `|` indique au shell de créer un tube anonyme entre `ls` et `more`.
- `dup2()` est utilisé pour :
  - connecter la sortie standard de `ls` à l'entrée du pipe ;
  - connecter l'extrémité du pipe à l'entrée standard de `more`.
- **Création avec `pipe()`** : Cette fonction retourne deux descripteurs (un pour la lecture et un pour l'écriture) pour gérer chaque extrémité du tube.

Exemple de code pour un tube anonyme

```
int pipe_fd[2];
pipe(pipe_fd);
int pid = fork();

if (pid == 0) {  // Processus enfant
    close(pipe_fd[0]); // Ferme l'extrémité de lecture
    write(pipe_fd[1], "Hello", 6);
    close(pipe_fd[1]);
} else {  // Processus parent
    close(pipe_fd[1]); // Ferme l'extrémité d'écriture
    char buffer[80];
    read(pipe_fd[0], buffer, 80);
    printf("Message du fils : %s\n", buffer);
    close(pipe_fd[0]);
}
```

1.4.1. Redirection de Flux avec `dup2()`

- **Principe** : `dup2(orig, copy)` redirige un descripteur (par exemple, `stdout`) vers un autre descripteur pour qu'ils pointent vers la même ressource.
- **Exemple avec le shell** :
  - La commande `ls | more` redirige la sortie de `ls` vers l'entrée de `more`, grâce au symbole `|` et à l'appel système `dup2()`.
- **Exemple de code** :

```c
int pipe_fd[2];
pipe(pipe_fd);
if (fork() == 0) {
    close(pipe_fd[0]);
    dup2(pipe_fd[1], STDOUT_FILENO); // Redirige stdout vers l'écriture du tube
    execlp("ls", "ls", NULL);
} else {
    close(pipe_fd[1]);
    dup2(pipe_fd[0], STDIN_FILENO); // Redirige stdin vers la lecture du tube
    execlp("more", "more", NULL);
}
```