

Les tubes

LO41

P.Descamps

Principe des tubes

- **Les tubes Unix sont une implantation de la communication suivant le schéma producteur consommateur, avec un tampon de taille fixe, interne au système.**
 - Pour un processus, un tube se présente comme **deux flots**, l'un ouvert en écriture (pour le producteur), l'autre ouvert en lecture (pour le consommateur).
 1. Lors d'une écriture de n octets par le producteur, celui-ci sera mis en attente s'il n'y a pas assez de place dans le tube pour y mettre les n octets. Il sera réactivé lorsque les n octets auront pu être tous écrits dans le tube.
 2. Le système interrompra l'exécution normale du processus s'il n'y a plus de consommateur pour ce tube.
 3. Lors d'une lecture de p octets par le consommateur, celui-ci sera mis en attente s'il n'y a pas assez d'octets dans le tube pour satisfaire la demande.

Principe des tubes

- Fichiers spéciaux gérés selon une méthodologie FIFO
- Taille limitée, généralement 10 blocs
- Deux types de tubes :
 - **Tubes ordinaires**, non visibles dans l'arborescence, entre processus d'une même filiation
 - **Tubes nommés**, visibles dans l'arborescence, entre processus non forcément liés par filiation
- Création ou ouverture de tube retournent des descripteurs de fichiers ouverts gérés comme les autres fichiers

Tube ordinaire

- Accès par création ou héritage
- Les descripteurs créés sont ajoutés à la table des descripteurs de fichiers du processus appelant.
- Perte par un processus = perte d'accès définitive par ce processus
- Position courante entièrement déterminée par les lectures/écritures
 - Primitive **lseek** interdite
- Taille maximum déterminée par `PIPE_BUF` définie dans `<limits.h>`
- Opération de lecture par défaut bloquante
 - Bloquant jusqu'à lecture de la taille demandée ou disparition de tous les écrivains
 - Pour non bloquante : utiliser **fcntl** avec le drapeau `O_NONBLOCK`
- Opération d'écriture atomique (tout est écrit) si opération bloquante
 - Signal `SIGPIPE` si aucun lecteur
- Primitive correspondante : **int pipe(int descfich[2])**
 - 0 pour la lecture et 1 pour l'écriture
 - retourne -1 en cas d'erreur
 - Se mettre d'accord pour l'utilisation (celui qui écrit, ceux qui lisent)

Tube ordinaire

- **Le processus appelant crée un tube accessible par lui-même et par tous les processus de sa descendance qui seront créés ultérieurement (fils, petitsfils...).**
- Le vecteur tube est un vecteur de descripteurs de fichiers où :
 - tube[0] est ouvert en lecture,
 - tube[1] ouvert en écriture ;
- les données écrites dans tube[1] (voir write) sont lues dans tube[0] (voir read) dans l'ordre First-In, First-Out (FIFO),
- Exemple :
 - write(tube[1], "bonjour", 8) :
 - envoi de caractères dans le tube, on indique le nombre de caractères à déposer (ici 8 pour écrire aussi le '\0' de fin de chaîne).
 - read(tube[0], tabcar, i) :
 - lecture de i caractères dans le tube, les caractères sont copiés dans le tableau tabcar (qui doit pouvoir contenir au moins i caractères).

Manipulation Tube ordinaire:

Lecture

- **int read** (int fildes, char buf[], unsigned nbyte)
 - read tente de lire nbyte octets dans le fichier associé au descripteur fildes (les octets lus sont copiés dans buf).
 - renvoie le nombre d'octets effectivement lus

– Algorithme de lecture :

- si le tube n'est pas vide :

- extrait au plus taille caractères du tube ;

- si le tube est vide :

- s'il y a des descripteurs ouverts en écriture : attend une écriture dans le tube (la lecture est bloquante par défaut);

- s'il n'y a plus de descripteurs ouverts en écriture : retourne 0.

– Conseil : toujours fermer les descripteurs dont on a pas besoin

Manipulation Tube ordinaire

:Ecriture

`int write (int fildes, char buf[], unsigned nbyte)`

- write tente d'écrire nbyte octets dans le fichier associé au descripteur fildes (les octets sont lus dans buf).
- renvoie le nombre d'octets effectivement écrits ($\leq nbyte$). Par défaut, un write sur un tube réussit toujours mais peut éventuellement être bloquant si le buffer alloué au tube est plein.
- Algorithme d'écriture :

– s'il n'y a pas de descripteur en lecture d'ouvert :

» *le signal SIGPIPE (broken pipe) est envoyé au processus écrivain (qui se termine par défaut) ;*

– s'il y a des descripteurs en lecture d'ouverts :

» l'écriture est bloquante par défaut : attend une lecture sur le pipe ;

Manipulation Tube ordinaire

- Fermeture

int **close** (int fildes)

- Ferme le fichier associé au descripteur fildes (on ferme un tube en lecture avec `close(tube[0])`).

Redirection des E/S standards

- La fonction : `int dup(desc);`
 - permet d'obtenir une copie d'un descripteur ;
 - le descripteur est le plus petit nombre dans la table des descripteurs.
 - Exemple de la re-direction de la sortie standard vers un tube :

```
int tube[ 2];
```

```
if( pipe( tube ) == -1 ){  
    perror( "tube" );  
    exit( 1 );  
}
```

```
close( 1 ); /* ferme la sortie standard */  
close( tube[0] );  
dup( tube[1] );
```

Exemple Tube ordinaire

- 1. Le processus père crée un tube en utilisant pipe()
- 2. Le processus père crée un ou plusieurs fils en utilisant l'appel système fork() ;
- 3. Le processus écrivain ferme le fichier, non utilisé, de lecture du tube ;
- 4. De même, le processus lecteur ferme le fichier, non utilisé, d'écriture du tube ;
- 5. Les processus communiquent
- 6. Chaque processus ferme son fichier lorsqu'il veut mettre fin à la communication

```
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/wait.h>

void parent(int tube, char *message) {
    sprintf(message, "Message du parent (%d)\n", getpid());
    write(tube, message, strlen(message + 1));
    close(tube);
}

void enfant(int tube) {
    char boite;
    printf("Message reçu par %d : \\", getpid());
    while (read(tube, &boite, 1) == 1)
        putchar(boite);
    printf("\\n\\n");
}

int main() {
    int pid, descfich[2], code_retour;
    char tampon[80];
    pipe(descfich);
    pid = fork();
    switch (pid) {
        case -1 :
            fprintf(stderr, "echec de fork\\n");
            exit(1);
        case 0 : /* enfant */
            close(descfich[1]); /* ... n'ecrit pas dans le tube mais lit */
            enfant(descfich[0]);
            break;
        default : /* parent */
            close(descfich[0]); /* ... ne lit pas dans le tube mais ecrit */
            parent(descfich[1], tampon);
            wait(&code_retour);
            break;
    }
    return 0;
}
```

Tubes nommés

- Tube qui possède un nom dans le SGF.
- Tous processus est autorisé à l'ouvrir à condition de connaître son nom et de posséder les droits d'accès.
- Un processus ayant ouvert un tube en écriture est suspendu tant qu'il n'y a pas de processus lecteur.
- Les données du tube ne sont pas gardés d'une session à l'autre.

Tubes nommés

- Création par la primitive **mknod** qui permet de créer les fichiers spéciaux ou par **mkfifo**
 - **mknod** était préalablement réservée au super-utilisateur
- Ouverture possible (Open) par les processus connaissant le nom du tube
 - Par défaut, ouverture bloquante : lecteur et écrivain s'attendent → synchronisation
- Suppression du tube lorsque explicitement demandé et pas d'ouverture en cours
 - Si suppression alors qu'il existe des lecteurs et écrivain, fonctionnement comme un fichier ordinaire
- Primitives correspondantes :
 - **int mknod(const char *nom_fich, mode_t mode, dev_t n_periph)**

```
#include <sys/types.h>
#include <sys/stat.h>

int mkfifo (const char *ref, mode_t mode) ;
```

○ Le paramètre **ref** définit le chemin d'accès au tube et le paramètre **mode** les droits d'accès des différents utilisateurs à cet objet,

Tubes nommés : Exemple

```
/* Processus ecrivain */
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
main()
{mode_t mode;
int tub;
mode = S_IRUST | S_IWUSR;
mkfifo ("fictub",mode)           /* création fichier FIFO */
tub = open("fictub",O_WRONLY)    /* ouverture fichier */
write (tub,"0123456789",10);     /* écriture dans fichier */
close (tub);
exit(0);}

```

Tubes nommés : Exemple

```
/* Processus lecteur */
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
main()
{int tub;
char buf[11];

tub = open("fictub",O_RDONLY)      /* ouverture fichier */
read (tub,buf,10);                /* lecture du fichier */
buf[11]=0;
printf("J'ai lu %s\n", buf);
close (tub);
exit(0); }
```

