Système partie 1 - cours 3

Pastille 6 : les tubes

N. de Rugy-Altherre - Vincent Colotte

Communication par un tube

Idée

- Échange d'information entre 2 processus à l'aide d'un fichier
- Principe simple à mettre en oeuvre.
- Linux : propose de faciliter ce mode de communication
- Utilisation dans le Shell avec | (pipe)

Communication par un tube

Idée

- Échange d'information entre 2 processus à l'aide d'un fichier
- Principe simple à mettre en oeuvre.
- Linux : propose de faciliter ce mode de communication
- Utilisation dans le Shell avec | (pipe)

Exemple `ls /home/user/toto/ | wc -l`

- `ls repertoire` :
- `wc -l fichier` :
- `|` :

Communication par un tube

Idée

- Échange d'information entre 2 processus à l'aide d'un fichier
- Principe simple à mettre en oeuvre.
- Linux : propose de faciliter ce mode de communication
- Utilisation dans le Shell avec | (pipe)

Exemple `ls /home/user/toto/ | wc -l`

- `ls repertoire : liste les fichiers/dossiers contenu dans` */home/user/toto/*
- `wc -l fichier : compte le nombre de ligne d'un fichier`
- `| :`

Communication par un tube

Idée

- Échange d'information entre 2 processus à l'aide d'un fichier
- Principe simple à mettre en oeuvre.
- Linux : propose de faciliter ce mode de communication
- Utilisation dans le Shell avec | (pipe)

Exemple `ls /home/user/toto/ | wc -l`

- `ls repertoire : liste les fichiers/dossiers contenu dans` */home/user/toto/*
- `wc -l fichier : compte le nombre de ligne d'un fichier`
- `| : redirection par un tube/pipe de la sortie de` *ls* `vers l'entrée de` *wc*
  ⇒ `la commande compte le nombre de fichiers et dossiers contenu dans le répertoire` */home/user/toto/*

Communication par un tube

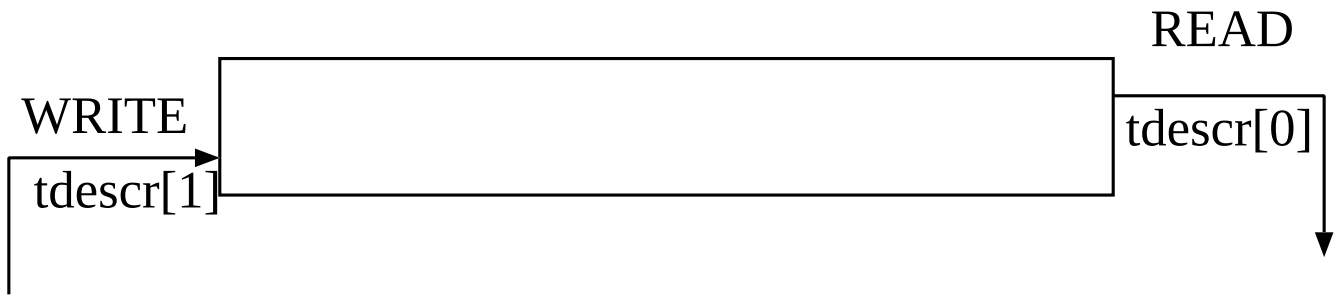
Un tube

- fait partie du système de fichier.
- lecture et écriture avec *read*() et *write*()
- unidirectionnel : une extrémité pour l'écriture et une pour la lecture.
- mode *stream* : lecture/écriture par octet (par opposition au mode par structure).
- la lecture est destructrice.
- un tube a une capacité : un tube peut donc être plein.

Communication par un tube

Un tube

- fait partie du système de fichier.
- lecture et écriture avec *read*() et *write*()
- unidirectionnel : une extrémité pour l'écriture et une pour la lecture.
- mode *stream* : lecture/écriture par octet (par opposition au mode par structure).
- la lecture est destructrice.
- un tube a une capacité : un tube peut donc être plein.



Communication par un tube

Propriétés :

nbre de lecteurs : nbre de descripteurs associés à l'entrée en lecture.
$\rightarrow$ si nbre de lecteur = 0, alors pas d'écriture (le signal SIGPIPE est envoyé).

Communication par un tube

Propriétés :

nbre de lecteurs : nbre de descripteurs associés à l'entrée en lecture.
$\rightarrow$ si nbre de lecteur = 0, alors pas d'écriture (le signal SIGPIPE est envoyé).

nbre d'écrivains : nbre de descripteurs associés à l'entrée en écriture.
$\rightarrow$ si nbre d'écrivain = 0 et le tube est vide alors c'est équivalent à fin de fichier.
(s'il ne peut devenir nul, il y a interblocage)

Tubes «anonymes»

*Ces tubes sont des fichiers **sans référence**, ils ne peuvent être utilisés que par des processus de **même filiation** et n'être manipulés que par des descripteurs.*

Tubes «anonymes»

*Ces tubes sont des fichiers **sans référence**, ils ne peuvent être utilisés que par des processus de **même filiation** et n'être manipulés que par des descripteurs.*

Intérêt

- Un processus (père) crée un tube puis crée un processus fils
- Les descripteurs sont hérités $\Rightarrow$ le fils est directement connecté au tube
- Reste à transmettre le numéro du descripteur pour la lecture ou l'écriture (à la création du fils)
- La destruction est gérée par le système (après la fermeture (*close*) du dernier descripteur en mémoire).

Tubes «anonymes»

```
int pipe (int *tdesc) ;
```

crée une nouveau tube :

- alloue un (i-)noeud,
- 2 entrées dans la table des fichiers ouverts,
- 2 descripteurs : *tdesc*[0] pour la lecture et *tdesc*[1] pour l'écriture.

Tubes «anonymes»

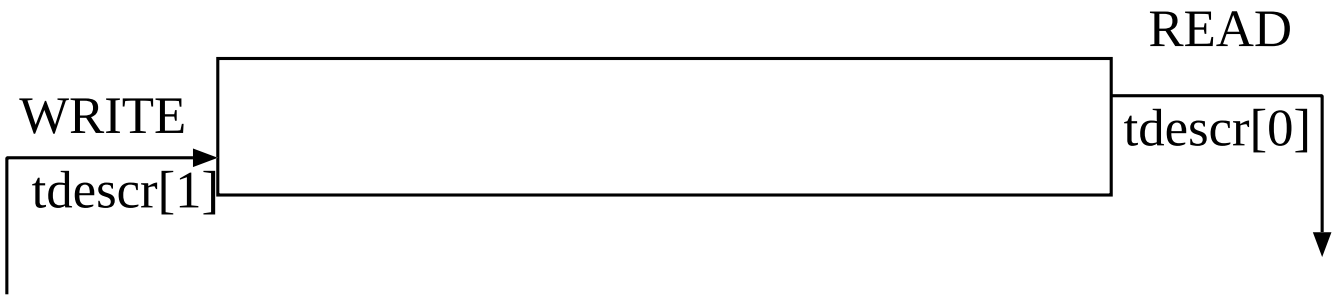
```
int pipe (int *tdesc) ;
```

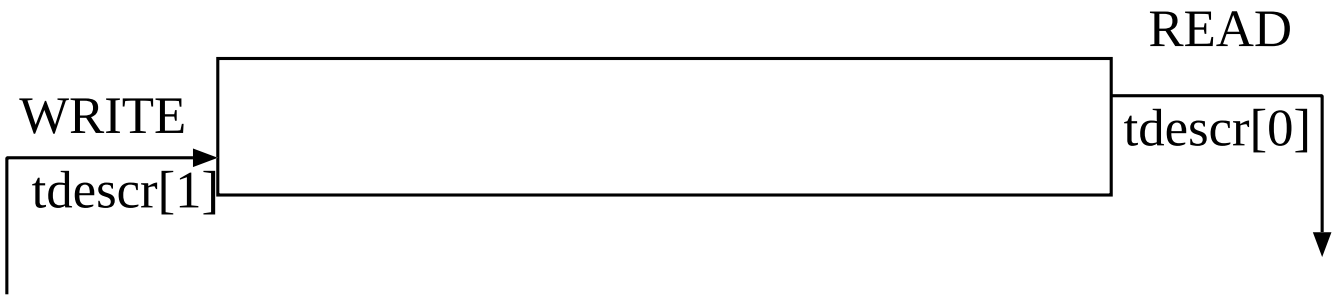
crée une nouveau tube :

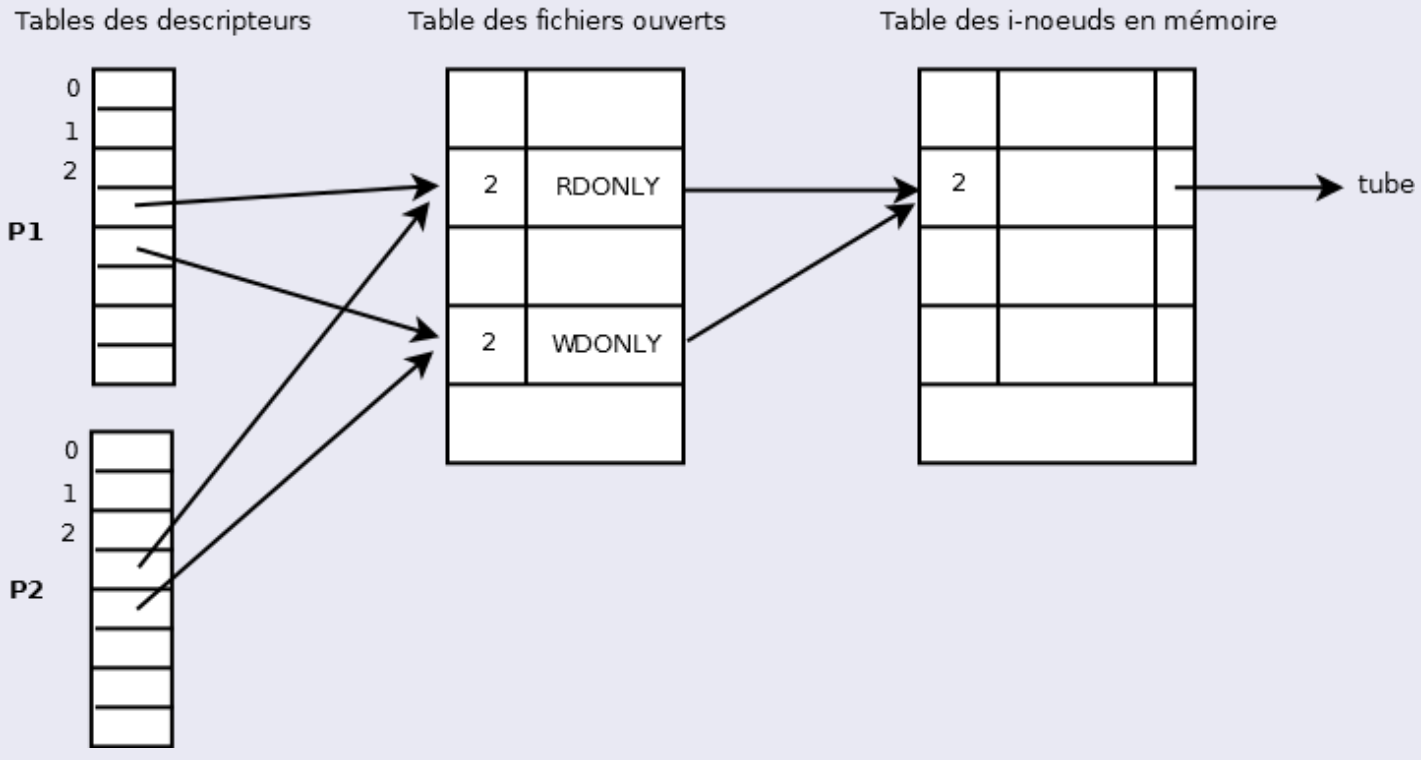
- alloue un (i-)noeud,
- 2 entrées dans la table des fichiers ouverts,
- 2 descripteurs : *tdesc*[0] pour la lecture et *tdesc*[1] pour l'écriture.

Á noter

- L'écriture se fait en fin de tube/fichier.
- La lecture en début de tube/fichier.


READ
WRITE
tdescr[0]
tdescr[1]


READ
WRITE
tdescr[0]
tdescr[1]


Tables des descripteurs
Table des fichiers ouverts
Table des i-noeuds en mémoire
0
1
2
P1
0
1
2
P2
2
RDONLY
2
WDONLY
2
tube

Lecture

Avec *read*() :

1. **si le tube n'est pas vide** $\Rightarrow$ lecture du nbre de caractères demandés (ou moins si pas assez)

Lecture

Avec *read*() :

1. **si le tube n'est pas vide** $\Rightarrow$ lecture du nbre de caractères demandés (ou moins si pas assez)
2. **si le tube est vide**
   $\rightarrow$ si nbre d'écrivains $= 0 \Rightarrow$ EOF, renvoie 0.
   $\rightarrow$ si nbre d'écrivains > 0
   - lecture bloquante $\Rightarrow$ attente de remplissage
   - lecture non-bloquante $\Rightarrow$ retour et renvoie de -1.

Ecriture

Avec *write*() :

1. **si nbre de lecteur = 0** $\Rightarrow$ envoie du signal SIGPIPE (message "Broken Pipe")
   Comme plus de lecteur et qu'il ne peut plus y en avoir, l'écrivain devient inutile (terminaison du processus ou retour -1).

Ecriture

Avec *write*() :

1. **si nbre de lecteur = 0** $\Rightarrow$ envoie du signal SIGPIPE (message "Broken Pipe")
   Comme plus de lecteur et qu'il ne peut plus y en avoir, l'écrivain devient inutile (terminaison du processus ou retour -1).
2. **si nbre de lecteur > 0**
   - écriture bloquante $\Rightarrow$ écrit en une seule fois sinon attend que le tube se vide suffisamment.
   - écriture non-bloquante :
     a. si $n \leq$ PIPE_BUF, et au moins n emplacements libres dans le tube, écriture.
     b. si $n \leq$ PIPE_BUF, et pas assez de place dans le tube $\rightarrow$ pas d'écriture et retour -1.
     c. si $n >$ PIPE_BUF, écriture d'un nbre inférieur à n (et retour -1 ! ?).

Tubes nommés

Contrairement aux tubes anonymés, les tubes nommés sont référencés dans le SGF

```
#include <sys/types.h>;
#include <sys/stat.h>;
int mkfifo(const char *ref, mode_t droits);
```

Propiétés

- Création avec *mkfifo*.
- Ouverture avec *open**
- Lecture ou écriture comme tube anonyme
- Il faut supprimer physiquement le tube (*unlink* ou *rm*).
  - *close* ne suffit pas

Tubes nommés : ouverture

Ouverture avec *open* : ouverture en lecture ou en écriture **bloquante**

$\Rightarrow$ permet la synchronisation à l'ouverture

Propriétés

- ouverture en lecture bloquante en cas :
  - d'absence d'écrivain
  - de processus bloqué sur une ouverture en écriture.
- ouverture en écriture bloquante en cas :
  - d'absence de lecteur
  - de processus bloqué sur une ouverture en lecture.

Chapitre 2 (Partie 2) : Système de Gestion des Fichiers (sous Linux)

Pastille 6 : les tubes

Vincent Colotte

Université de Lorraine - FST Nancy