Tubes

Thèmes

- Utilisation des tubes
- Pipelines

Ressources Pour ce TP, comme pour les suivants, vous pourrez vous appuyer sur

- Le polycopié intitulé « Systèmes d'exploitation : Unix », qui fournit une référence généralement suffisante sur la sémantique et la syntaxe d'appel des différentes primitives de l'API Unix. Chaque section du sujet de TP indique la (ou les) section(s) du polycopié correspondant au contenu présenté.
- Les pages du manuel en ligne (commande `man`), et plus particulièrement les sections 2 et 3.

1 Minishell : Redirection des entrées/sorties d'un processus

Terminez l'étape ⑯ du TP4, si ce n'est pas déjà fait.

2 Manipulation d'un tube de communication entre 2 processus

L'objectif est de tester l'effet de différents programmes utilisant les tubes de communication entre processus.

> Pour chaque scénario, prédire l'effet attendu **avant** de lancer le programme. Dans le cas où l'effet obtenu n'est pas l'effet attendu, proposer une explication (ou une correction de votre programme :)).

Accès aux tubes 1 Écrire un programme `tube1.c` qui comporte les étapes suivantes :

- Un processus père crée un fils, puis crée un tube.
- Le père écrit un entier dans le tube.
- Le fils lit un entier dans le tube et affiche l'entier lu.

Accès aux tubes 2 Écrire un programme `tube2.c` qui comporte les étapes suivantes :

- Un processus père crée un tube, puis écrit un entier dans le tube.
- Il crée ensuite un fils qui lit un entier dans le tube et affiche l'entier lu.

Couplage en lecture 1 Écrire un programme `tube3.c` qui comporte les étapes suivantes :

- Un processus père crée un tube, puis un fils.
- Le père écrit une série d'entiers dans le tube, puis attend (par appel à `pause()`), puis se termine.
- Le fils effectue une boucle consistant à lire un entier du tube, afficher l'entier et la valeur retournée par `read(...)`, jusqu'à ce que cette valeur soit ≤ 0. Une fois sorti de la boucle, le fils affiche un message `"sortie de boucle"`, puis se termine.

Couplage en lecture 2 Créer le fichier `tube4.c` par copie du fichier `tube3.c`. Modifier le fichier `tube4.c` de manière à remplacer l'appel à `pause()` par la commande `sleep(10)`.

Couplage en écriture Écrire un programme `tube5.c` qui comporte les étapes suivantes :

- Un processus père crée un un tableau de `N` octets, puis un tube, puis un fils.
- Le père effectue une boucle infinie qui :
  - écrit le contenu du tableau dans le tube,
  - attend 1 seconde,
  - puis affiche la valeur retournée par `write(...)`.
- Le fils a le comportement suivant :
  - il attend un signal (qu'il devra ignorer, et qui pourra être envoyé via le terminal),
  - il lit une série de 10*N octets,
  - puis il se termine.

En profiter pour évaluer la taille d'un tampon.

3 Minishell : Tubes

Etape 19 (Tubes simples) Complétez le minishell pour permettre de composer des commandes en les reliant par un tube.

Exemple : `ls | wc -l` doit lancer les commandes `ls` et `wc`, la sortie standard de `ls` étant connectée à l'entrée standard de `wc` par un tube.

Etape 20 (Pipelines) Étendez la fonctionnalité précédente en offrant la possibilité d'enchaîner une séquence de filtres liés par des tubes, de sorte à obtenir un traitement en pipeline.

Exemple : `cat toto.c lulu.c | grep int | wc -l`

Etape 21 (Rendu) Comme aux TP précédents et même si l'étape 20 n'est pas achevée (voire commencée), archivez votre travail sur le `minishell` via la commande `make archive`. Le résultat est un fichier nommé `minishell-votreidentifiant.tar`. Chargez ce fichier dans la section rendu, dans la zone qui correspond à votre groupe de TD.