TD6 – Redirection des flux d'entrée-sortie et communication entre processus par tubes

1 Objectif

Ces exercices introduisent aux mécanismes de redirection de Posix (`fcntl()`, `dup()`, `dup2()`), ainsi que la communication de données par tubes (`pipe()`).

2 Redirection du shell

Ecrire un programme `tst_redirect.exe` qui simule la commande au shell suivante :

```
% ls * foo bar >! out 2>&1
```
[1]

Ce programme fera appel aux fonctions `fork()` et `execXX()`, et non à la fonction `system()`.

Indication Exécutez `ls` avec une liste de fichiers dont certains existent dans le répertoire et d'autres pas. Par exemple : `% ls * foo bar`
La liste des fichiers du répertoire s'affiche - sur la sortie standard (stdout, descripteur de fichier 1), ainsi que des messages d'erreur, un pour chaque fichier inexistant - sur le flux d'erreur standard (stderr, descripteur de fichier 2).
Pour rediriger ces deux flux *sans conflit* sur un même fichier, la commande est celle ci-dessus. La syntaxe spéciale `2>&1` fait du descripteur 2 un *synonyme* du descripteur 1. Comme 1 a lui-même été redirigé sur `out` juste avant, 2 désigne aussi `out`, avec partage du pointeur d'entrée-sortie.

3 Tubes du shell

Écrire un programme `tst_pipes.exe` qui simule la commande au shell suivante :
`ps | grep emacs | wc -l`

4 Anneau à jetons

Les réseaux à jetons en anneau sont un moyen simple de synchronisation entre plusieurs stations opérant en parallèle. Plusieurs réseaux commerciaux reposent sur ce type de protocole. Nous allons ici réaliser un *simulateur* d'un tel anneau.

4.1 Structure de l'anneau (`ring.exe`)

Un réseau en anneau à la structure représentée sur la figure 1. Il est composé de `N` stations $S_0, \ldots S_{N-1}$ (`N=5` sur la figure). Pour nous, une station correspondra à l'exécution d'un fichier binaire indépendant, nommé `station.exe`, qui prend un unique argument, son numéro d'ordre (`i= 0, ... N-1`). Chaque station est connectée à ses deux voisines par l'intermédiaire de `N` tubes (p_i sur la figure). Le flux de communication dans les tubes est orienté comme indiqué sur la figure et les tubes sont branchés sur l'entrée et la sortie standard de chaque station. Toutes les stations sont (potentiellement[2]) actives en parallèle.

[1] Le point d'exclamation permet d'écraser un fichier existant
[2] Le protocole choisi (cf. § 4 .3) confèrera une stricte séquentialité au fonctionnement de ce système en parallèle.

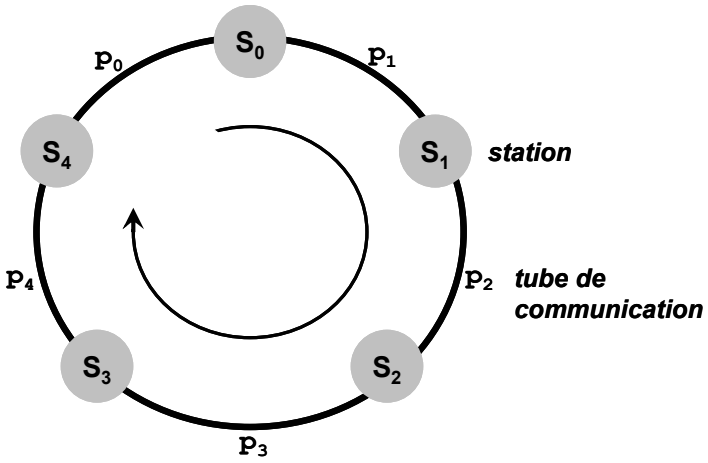

Figure 1: Schéma de l'anneau à jetons (N=5)

Ecrire un programme `ring.exe` qui met en place cette structure de processus et les connexions correspondantes. Ce programme recevra un unique argument : le nombre `N` de stations de l'anneau (en pratique, moins d'une dizaine).
Pour tester ce programme, vous pouvez écrire une version de `station.exe` qui recopie sur la sortie standard ce qui est lu sur l'entrée (en l'affichant sur l'erreur standard, `stderr`).

Indication Le programme `ring.exe` va créer `N` processus fils, un pour chaque station.
- Le processus père doit impérativement attendre la fin de tous ses fils (`wait()`),
- Les processus fils et le père lui-même doivent fermer (`close()`) tous les descripteurs de fichiers qui leur sont inutiles.

4.2 Données circulant sur l'anneau (paquet)

Sur l'anneau tout entier circule un seul *paquet* à la fois. Ce paquet unique est composé des éléments suivants :
- un *jeton* qui a trois valeurs possibles : LIBRE, ACCUSÉ DE RÉCEPTION, et OCCUPÉ ;
- le numéro (de `0` à `N-1`) de la station *émettrice* du message ;
- le numéro (de `0` à `N-1`) de la station *réceptrice* du message ;
- le *message* lui-même, une chaîne de caractères quelconque de longueur fixe `MSG_LEN`.

4.3 Protocole de l'anneau (`station.exe`)

Le principe du protocole de l'anneau à jeton est simple. Quand une station voit passer le paquet avec un jeton LIBRE, elle peut lui attacher un message et propager le paquet. Quand la station destinatrice reçoit le paquet, elle le « consomme », marque le jeton en ACCUSÉ DE RÉCEPTION et le propage à nouveau. Le paquet va donc faire un tour complet pour revenir à son émetteur qui prendra acte de l'ACCUSÉ DE RÉCEPTION, remettra le jeton dans l'état LIBRE, et passera le paquet à la station suivante. Bien entendu, ceci ne fonctionne que si chaque station propage à sa voisine les paquets qui ne la concernent pas.
Plus précisément, chaque station déroule l'algorithme suivant :
1. La station se met en attente (bloquante) du paquet sur son entrée standard ;
2. Quand le paquet arrive, la station prépare une nouvelle configuration du paquet :

- Si le jeton est LIBRE, cela signifie qu'il n'y a pas de message valide. La station engage un dialogue au terminal avec l'utilisateur, en lui demandant s'il a un message à transmettre et, si oui, lequel et à quelle station.
  Si l'utilisateur veut envoyer un message, la station prépare un paquet en marquant le jeton OCCUPÉ, en renseignant les numéros des stations et en y copiant le message. En revanche si l'utilisateur ne souhaite pas envoyer de message, le paquet reçu n'est pas modifié.
- Si le jeton n'est pas LIBRE, cela signifie que le paquet contient un message valide. Il y a alors plusieurs cas, dont les deux premiers ne sont pas forcément exclusifs[3] :
  - Si la station courante est la destinatrice de ce message, elle marque le jeton en ACCUSÉ DE RÉCEPTION ;
  - Si la station courante était l'émettrice du message, si le jeton est un ACCUSÉ DE RÉCEPTION, cela signifie que son message a été reçu correctement par son destinataire et que le paquet a fait un tour complet ; la station remet alors le jeton dans l'état LIBRE ;
  - Si aucune des deux conditions précédentes n'est vraie, le paquet est inchangé.

3. Le paquet (nouveau ou inchangé) est envoyé sur la sortie standard à la station suivante.

4.4 Initialisation de l'anneau

Nous supposerons que c'est le rôle de la station de numéro 0 d'envoyer un premier paquet (LIBRE) sur le réseau.

4.5 Réalisation de `station.exe`

Indication pour le dialogue avec l'utilisateur L'entrée et la sortie standard de chaque station sont occupées à assurer la connectivité du réseau. Elles ne peuvent donc plus être utilisées pour dialoguer au terminal. Reste l'erreur standard (`stderr`) mais si on peut y écrire, il n'est pas conseillé d'y lire.
Pour dialoguer avec l'utilisateur, chaque station ouvrira donc, par `fopen()`, le périphérique nommé `/dev/tty`, le *terminal de contrôle*. Attention, pour éviter les problèmes, il convient d'ouvrir ce terminal deux fois, une fois en entrée, une autre fois en sortie.
`fopen()`, retourne des `FILE *`, qui sont directement utilisables avec `fprintf()` et `fscanf()`.

Indications pour le débuggage Vous avez intérêt à tracer le fonctionnement de vos stations. Les messages de trace peuvent être envoyés sur `stderr`, ou sur le flux de sortie associé à `/dev/tty`.

[3] Cas d'une station s'envoyant un message à elle-même.