R3.05 - TP 5
Tubes (suite)

Synchronisation de processus

L'un des usages des tubes est la synchronisation de processus.

L'ascenseur

Prenons le cas concret d'un processus gérant un ascenseur dans un immeuble de bureaux. Quand une personne appuie sur le bouton d'appel à un étage, le processus envoie l'ascenseur à l'étage où le bouton a été appuyé. Tant que l'ascenseur n'a pas fini sa course (envoi à l'étage et retour au RDC), le processus ne prend aucune autre demande.

Mise en œuvre

On va considérer que :

- Il est 18H, tout le monde quitte l'immeuble. Donc, une fois arrivé à un étage où il a été appelé, l'ascenseur redescend au RDC pour déposer les personnes. A cette heure-là, personne ne remonte ni n'utilise l'ascenseur pour se rendre à un autre étage.
- L'ascenseur met 2 secondes pour passer un étage (vers le haut ou le bas)
- Il y a un RDC et 5 étages dans l'immeuble (de **`0`** à **`5`**)

Il y a un processus qui gère les demandes (appelons-le **`call`**) et un processus qui gère les mouvements de l'ascenseur (appelons-le **`lift`**). Ce sont 2 processus distincts qui communiquent entre eux par des tubes nommés :

- **`call`** écrit dans un tube (appelons-le **`call2lift`**) lu par **`lift`**, des chiffres indiquant à quel étage l'ascenseur est demandé.
- **`lift`** écrit dans un tube (appelons-le **`lift2call`**) lu par **`call`**, le caractère **`#`** à chaque fois qu'il a terminé sa course (collecte des usagers à l'étage + retour et dépose au RDC). Avec ce **`#`**, il indique ainsi qu'il est de nouveau libre.

Le processus **`lift`** fait une pause pour simuler les temps de déplacement de l'ascenseur (à calculer, en secondes).

Le processus **`call`** utilise aussi un tube nommé (appelé **`call_pipe`**) pour lire les appels des usagers. Écrivez ce programme **`call`** qui gère les appels des usagers et le

programme **lift** qui gère les mouvements de l'ascenseur. Pour tester vous pourrez envoyer des demandes d'usagers en faisant des :

```
echo "N° étage" > call_pipe
```

exemple :

```
echo "2" > call_pipe
```

Important : mettez bien un espace entre le **chiffre** et le **>** car :

```
echo 2> call_pipe
```

ne signifie pas qu'on souhaite envoyer un caractère **2** dans le fichier **call_pipe** mais que le **STDERR** de **echo** (tout seul) doit être envoyé vers **call_pipe**, ce qui n'est évidemment pas ce qu'on souhaite faire !

call et **lift** sont des programmes qui tournent en permanence : toujours en attente d'une demande pour **call** (tube **call_pipe**) et d'un envoi à l'étage pour **lift** (tube **call2lift**).

Astuce : faites des lectures dans les tubes caractère par caractère, ce sera plus simple puisqu'un étage tient sur 1 chiffre, comme le **#** qui indique le retour à l'état libre. Vous vous simplifierez ainsi le traitement si plusieurs demandes se suivent dans le "tuyau".

Le programmes **call** et **lift** doivent afficher ce qu'ils font :

```
call : appel au Xème étage
call : ascenseur signale qu'il est libre
lift : ascenseur appelé au Xème étage. Temps estimé : Y secondes
lift : ascenseur retourne au RDC. Temps estimé : Z secondes
lift : ascenseur libre. Temps total : T secondes
```

Vous pouvez tester plusieurs appels successifs en ouvrant plusieurs terminaux pour y taper un appel d'ascenseur (car l'appel est bloquant en principe)

Amélioration

Vous avez certainement noté que les écritures dans **call_pipe** sont bloquantes.

Ca vient du fait que le processus **call** attend que l'ascenseur soit libre (donc que **lift** ait écrit **#** dans **lift2call**) pour lire de nouveau dans **call_pipe**.

Proposez une solution pour éviter ces blocages. Vous devez mettre en place un buffer de mémorisation des demandes et faire en sorte que les tubes **`call_pipe`** et **`lift2call`** ne soient pas bloquants.