Copiez le fichier `/pub/2A/ASR-Reseau/serveur-cle-valeur.sh` dans votre répertoire `TP3`. Vous devrez en compléter le code dans l'exercice 2.

Exercice 1. netcat (client/serveur).

Nous avons déjà utilisé la fonctionalité "client" de `netcat` pour nous connecter à différents serveurs : `netcat` établit une connexion avec un serveur, puis tout ce qu'il lit sur son stdin, il l'envoit au serveur, et tout ce que le serveur lui envoie, il l'écrit sur son stdout.

`netcat` offre aussi une fonctionalité "serveur". Dans ce mode d'utilisation, netcat attend sur un port qu'un client se connecte, puis tout ce qu'il lit sur son stdin il l'envoie au client, et tout ce que le client lui envoie, il l'écrit sur son stdout. Le mode "listen" est sélectionné grâce à l'option `-l` ; il faut aussi l'option `-p` suivi du numéro de port sur lequel écouter.

Dans un terminal, faites démarrer `netcat` en mode serveur sur le port 8080 de la manière suivante :

```
netcat -l -p 8080
```

Dans un autre terminal, faites démarrer un client qui se connecte à ce serveur :

```
netcat localhost 8080
```

Maintenant, les lignes que vous entrez dans le terminal serveur apparaissent aussi sur le terminal client et toutes les lignes que vous entrez dans le terminal client apparaissent aussi sur le terminal serveur.

Exercice 2. Serveur bash clé/valeur. Cela va peut-être vous surprendre, mais on peut réaliser un serveur grâce à un script bash. Dans cet exercice, vous allez réaliser de cette manière un serveur clé/valeur tel qu'on l'a conçu en TD.

La table associative est réalisée de la manière suivante : pour chaque entrée (K,V) on créer un fichier K dans lequel on met la valeur V.

Pour réaliser le serveur, nous allons utiliser la fonctionalité "listen" de `netcat` :

```
netcat -l -p PORT
```

La commande ci-dessus écoute sur le `PORT` et attend qu'un client s'y connecte. Lorsque cela se produit, tout ce que `netcat` lit sur son `stdin`, il l'envoie au client et tout ce que le client lui envoie il l'écrit sur son `stdout`.

Pour pouvoir interagir avec le client, il faudrait pouvoir lire le `stdout` de `netcat` pour obtenir les requêtes du client, et, pour chaque requête, on voudrait envoyer une réponse sur le `stdin` de `netcat` pour que celui-ci la transmette au client.

On peut résoudre une partie de notre problème en utilisant un tube (pipe) :

```
interaction | netcat -l -p PORT
```

où `interaction` serait une fonction bash qui écrirait des réponses sur son `stdout`. Malheureuseument, on voudrait aussi que cette fonction reçoive sud son `stdin` les requêtes du client : c'est à dire ce qui est écrit sur le `stdout` de `netcat`. Il faudrait faire une sorte de boucle pour connecter le `stdout` de `netcat` sur la droite au `stdin` d'`interaction` sur la gauche.

Ceci peut être réalisé grâce à la notion de tube nommé (named pipe ou fifo). L'idée est de créer un objet tube et de le rendre accessible dans un répertoire comme si c'était un fichier. Un processus peut alors ouvrir ce fichier en lecture, un autre en écriture pour ainsi établir un tube entre eux deux.

Un tube nommé est créé par la commande `mkfifo` :

```
FIFO="/tmp/$USER-fifo-$$"
mkfifo "$FIFO"
```

La commande ci-dessus crée un tube nommé dans le répertoire `/tmp`. `$USER` substitue le nom de l'utilisateur et `$$` le pid du processus courant. Cette technique permet d'éviter les collisions si plusieurs utilisateurs et/ou plusieurs processus veulent créer des fifos dans `/tmp`.

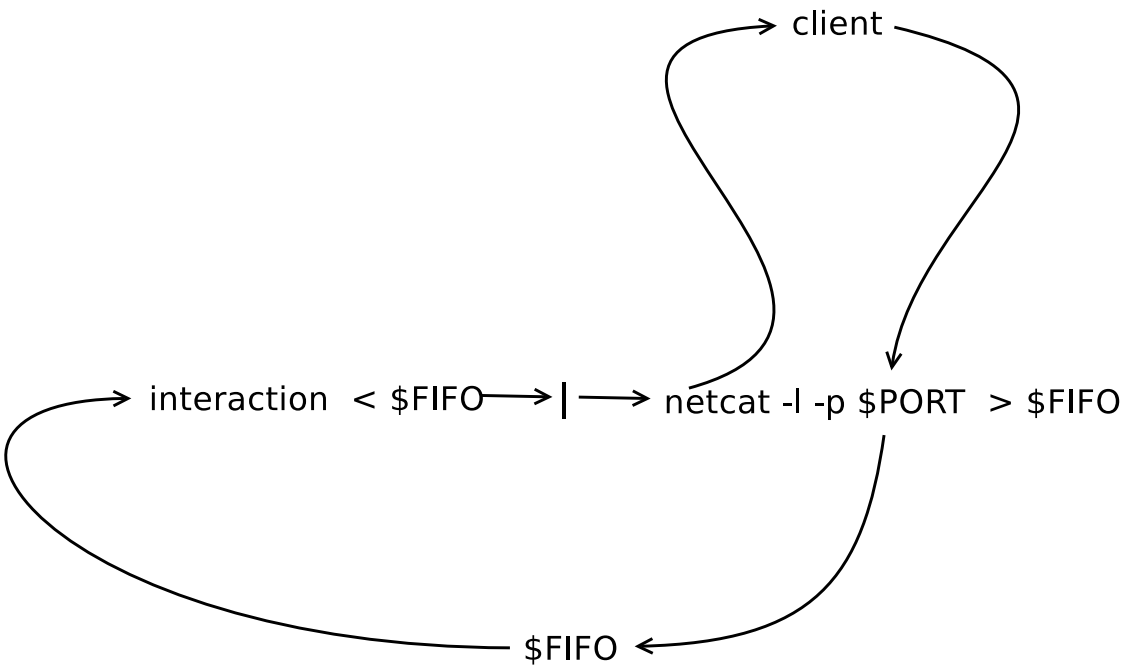
On peut à présent utiliser ce `$FIFO` pour rediriger la sortie de `netcat` dans l'entrée d'`interaction` :

```
interaction < "$FIFO" | netcat -l -p PORT > "$FIFO"
```

De cette manière, on a créé 2 tubes :

- un tube de `interaction` dans `netcat` grâce à l'opérateur `|`
- un tube de `netcat` dans `interaction` au travers de `$FIFO`

Le flot des communications est illustré dans le diagramme suivant :



Maintenant étudiez le code du script `serveur-cle-valeur.sh` pour le comprendre. Vous allez devoir compléter les définitions des fonctions implémentant les différentes commandes du serveur. Ces définitions contiennent les caractères `???` là où vous devez mettre votre code. Appuyez-vous sur l'automate que vous avez réalisé en TD.

Vous testerez ce serveur en vous y connectant grâce à `netcat`. N'utilisez pas `telnet` sinon il vous faudra gérer dans votre serveur les fins de lignes particulières de `telnet` (`\r\n` au lieu de `\n`).