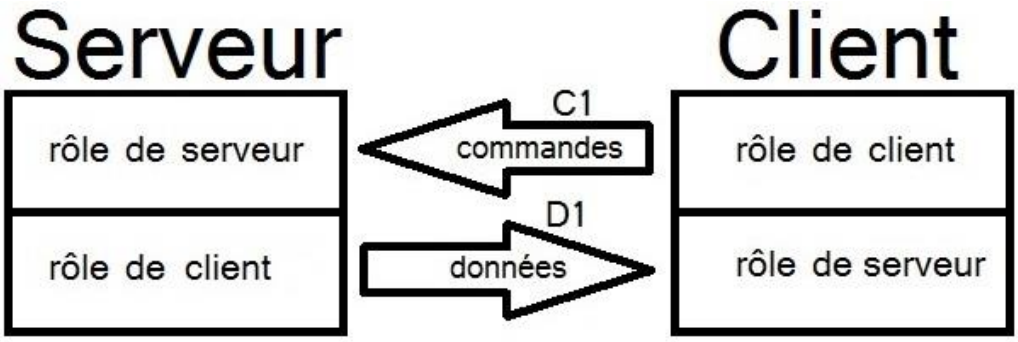
Vous allez écrire un client et un serveur **en python 3 sous Linux**.
Ce sera un simili-telnet.

Il y aura pour chaque client deux sockets :

- une socket de commandes C1 créée lors de la connexion initiale qui permettra d'envoyer les commandes au serveur et de recevoir ses réponses.
- une socket de données D1 créée pour les envois de fichiers (commandes rls ou rpwd) du serveur vers le client (le client sera serveur et le serveur devient client).



Les flèches sur le schéma indiquent les sens de connexion, les données circulant bien sur dans les 2 sens selon les besoins.

I – Connexion et identification :

Une fois le client connecté en TCP au serveur, la séquence de connexion effective sera :

- le client envoie BONJ au serveur.
- Le serveur répond avec WHO.
- Le client envoie son identifiant.
- Le serveur répond avec PASSWD.
- Le client envoie son mot de passe.
- Le serveur vérifie que l'identifiant est connu et que le mot de passe correspond. Si c'est le cas, il répond WELC et attend les commandes du client. Sinon, après 3 essais infructueux, il répond BYE et déconnecte le client.

Les identifiants et mots de passe seront stockés dans un fichier texte contenant sur chaque ligne un identifiant suivi d'un espace et du mot de passe correspondant.

II – Commandes reconnues :

Les commandes que le client peut lancer sont :

En local :

- ls : affiche le contenu de son répertoire courant.
- pwd : affiche le nom de son répertoire courant.
- cd : demande alors le répertoire où se déplacer localement.

En distant :

- rls : affiche le contenu du répertoire courant du serveur.
- rcd : demande alors le répertoire du serveur où se déplacer.
- rpwd : affiche le nom du répertoire courant du serveur.

III – Précisions et indications :

Pour rls et rpwd, le serveur stocke dans un fichier le résultat de la commande /bin/ls ou /bin/pwd dans un fichier (avec une redirection comme vu dans le Chapitre 1 en système avec os.dup2) et envoie le contenu de ce fichier au client qui va afficher ce qu'il a reçu.

Pour cd et rcd, la commande cd étant intégrée au shell (builtin), on ne peut la lancer avec os.execlp (comme on le fait pour rls, rpwd, ls, pwd ou rm). Il faut donc utiliser la fonction python os.chdir (int chdir(const char *path)). Si rcd s'est bien passé, le serveur envoie sur la socket de commande CDOK, sinon il y envoie NOCD. Le client signale à l'utilisateur si tout s'est bien passé.

Pour les commandes locales ou distantes, tout devra être fait en bas-niveau, **vous n'utiliserez pas les fonctions pré-écrites de python de gestion des répertoires (à l'exception de os.chdir).**

Pour les commandes distantes, le fonctionnement le plus simple est le suivant, après connexion authentifiée sur C1:

- Le serveur se met en attente en lecture sur C1.
- Le client lit sur l'entrée standard le nom de la commande et l'envoie au serveur sur C1.
- Si la commande est rcd, alors le serveur attend sur C1 le nom du répertoire destination.
- Le client se met en attente de connexion sur la socket D1.
- Une fois la commande reçue (et le nom du répertoire pour rcd), le serveur génère le résultat de la commande et le sauve dans un fichier (pour rls et rpwd).
- Le serveur se connecte alors au client sur D1.
- Le client se met en attente en lecture sur D1.
- Le serveur envoie le fichier généré par la commande (pour rls et rpwd) ou CDOK (si cd s'est bien passé) ou NOCD (si cd a eu un problème) sur D1.
- Le serveur ferme la connexion sur D1.
- Le client ferme la socket D1.
- On recommence à la première étape.

Il existe bien sur d'autres façons pour arriver à ce résultat. Ceci est juste une méthode, peut-être la plus simple.

IV - Notation :

Vous serez notés sur les critères suivants :
1- Connexion entre un client et d'un serveur concurrent. (1 pt)
2 - Demande par le client de connexion (BONJ) et réponse du serveur (WHO) (2 pts)
3 - Envoi de l'identifiant par le client. (1 pt)
4 - Demande par le serveur du mot de passe (PASSWD). (1 pt)
5 - Envoi du mot de passe par le client. (2 pts)
6 - Vérification par le serveur de id/mdp et envoi au client de la réponse (WELC ou BYE) (4 pts)
7 - commandes locales au client (ls, pwd, cd). (4 pts)
8 - commandes distantes sur le serveur (rls, rpwd, rcd). (5 pts)