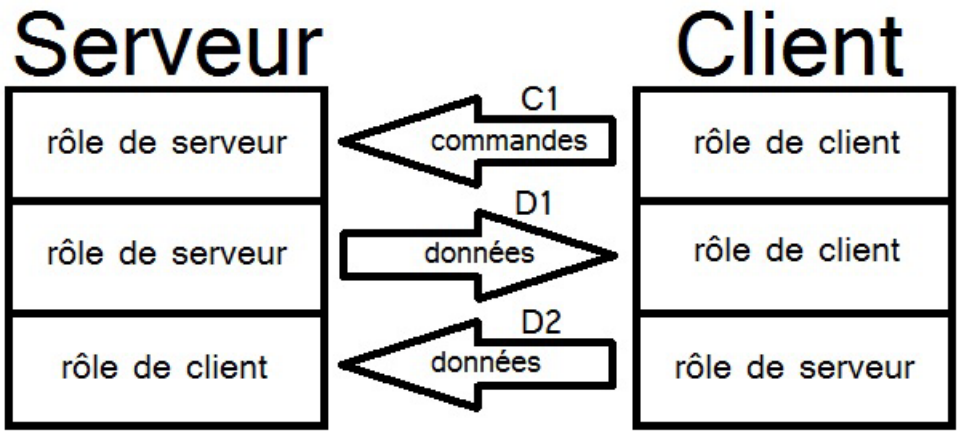
Vous allez écrire un client et un serveur **en C sous Linux**.
Ce sera un simili-FTP.

Il y aura pour chaque client trois 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 downl, rls ou rpwd) du serveur vers le client (le client sera serveur et le serveur devient client).
- Une socket de données D2 créée pour les envois de fichiers (commande upld) du client vers le serveur.



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.
- rm : demande alors le nom du fichier à supprimer 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.
- upld : demande alors un nom de fichier qui sera envoyé au serveur.
- downl : demande alors un nom de fichier qui sera envoyé par le serveur, s'il existe.

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 avec dup2) et envoie le contenu de ce fichier au client qui va afficher ce qu'il a reçu (pas d'utilisation de readdir ni d'opendir ni de getcwd).

Pour cd et rcd, la commande cd étant intégrée au shell (builtin), on ne peut la lancer avec execlp (comme on le fait pour rls, rpwd, ls, pwd ou rm). Il faut donc utiliser la fonction C 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 l'envoi de fichier, la commande upld est d'abord envoyée au serveur, qui attend alors le nom de fichier sur C1 . Une fois reçu, si le serveur a le droit de créer le fichier, il envoie le message RDY au client et se met en attente de connexion sur la socket D2. Le client se connecte et envoie le fichier que le serveur écrira sur le disque. Si le serveur n'a pas les droits nécessaires pour créer le fichier voulu, il envoie le message FBDN au client qui abandonne le transfert.

Pour la réception de fichier, la commande downl est envoyée au serveur, qui attend alors le nom de fichier. Si celui-ci existe et est accessible (i.e. peut être ouvert en lecture par le serveur), alors le serveur se connecte à la socket D1 du client et envoie le fichier sinon il envoie au client le message UNKWN.