Mise en Œuvre 2/2

11 décembre 2013

On désire créer un shell « distribué » qui permet l'exécution de commandes sur un serveur distant.

1 Fonctionnalités du shell

Fonctionnalités souhaitées :

- Exécution de commandes de distantes
- Exécution de commandes mixtes (en parties distantes, en partie locales)

Un membre prefixé par `s:addIP:port` doit être exécuté à distance, en se connectant à un serveur ayant pour adresse `addIP` et écoutant sur le port spécifié.

Ainsi la commande :

```
s:10.0.0.1:1234 ls | grep truc
```

doit exécuter ls sur l'ordinateur ayant pour adresse IP 10.0.0.1 et écoutant sur le port 1234 et envoyer le résultat de cette commande sur l'entrée standard d'un grep exécuté en local sur votre ordinateur.

2 Création d'un serveur

2.1 Principes

La création d'un serveur en C passe la l'utilisation des commandes :

- **socket** pour la création d'une socket
- **bind** pour nommer la socket
- **listen** pour attendre les connexions sur une socket donnée
- **accept** pour accepter une connexion sur une socket

L'architecture générique d'un serveur qui attend des connexions afin de les traiter est toujours la même :

```
#include <sys/types.h>
#include <sys/socket.h>
#include <linux/in.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <signal.h>

int main(int argc, char **argv)
{
        struct sockaddr_in SockAdr;
```

```
        int idSocket;
        int fdsocket;
        int taille;

        idSocket = socket(AF_INET, SOCK_STREAM,0);

        SockAdr.sin_family = AF_UNIX;
        SockAdr.sin_port = htons(1234);
        SockAdr.sin_add_.s_addr = htonl(INADDR_ANY);

        if(bind(idSocket,&SockAdr,sizeof(SockAdr)) == -1)
        {

        }

        listen(idSocket,10)

        while(1)
        {
                taille = sizeof(struct sockaddr_in);
                fdsocket = accept(idSocket,&sockAdr,&taille);
                // tratier la connexion
        }
}
```

Afin de créer des connexions concurrentes, il est necessaire de créer un nouveau fil d'exécution pour chaque nouvelle connexion. Cela peut se faire à l'aide fork, mais il est plus élégant de créer un nouveau thread.

La création d'un nouveau thread se fait à l'aide de la commande

`pthread_create`

Cette commande prend 4 arguments dont un pointeur sur fonction ainsi que qu'un pointeur vers le paramètre que l'on souhaite donner à cette fonction (cf man pthread_create).

On prendra garde à bien libérer les ressources du thread à la fin de son exécution. Pourquoi l'utilisation de join ne semble pas appropriée ici ?

3 Création d'un client

La création d'un client est nettement plus simple et passe par l'utilisation de `connect`.

Par exemple, pour se connecter à la machine 10.0.0.1 sur le port 1234 on pourra procéder ainsi :

```
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdlib.h>
int main ()
{
int sockfd,taille;
struct sockaddr_in adresse;
```

```
sockfd = socket (AF_INET, SOCK_STREAM, 0);
adresse.sin_family = AF_INET;
inet_aton("10.0.0.1",(struct in_addr*)&(adresse.sin_addr.s_addr));
adresse.sin_port = htons(1234);

taille = sizeof (adresse);
if(connect (sockfd, (struct sockaddr *)&adresse, taille) == -1)
{
perror ("Erreur");
exit (1);
}

// travail avec la socket

close (sockfd);
exit(0);
}
```