TP 2 : les sockets

Voici le lien de téléchargement des fichiers sources et du makefile: <https://lc.cx/ZyrX>

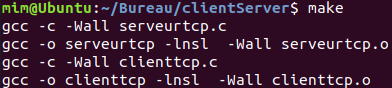
## Lire le code

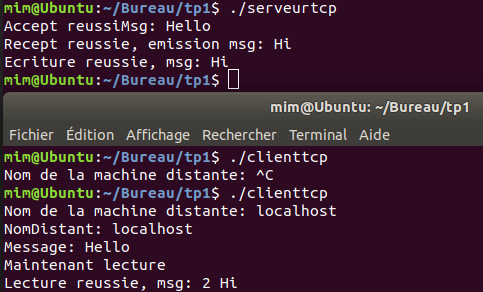
C’est fait.

## Commenter et expliquer

Tout est dans les fichiers sources

## Compiler et tester

Pour compiler faut utiliser la commande make.  




Modifier la saisie afin qu’une chaîne quelconque soit échangée (espaces compris) en remplaçant les scanf par Testez pour vérifier le bon fonctionnement.



## Analysez et commentez les résultats de l’execution de netstat –nutelap (avant, pendant et après le lancement du client) : expliquez.

Avant : ni le serveur ni le client n’est démarré :

## 

## Pendant l’execution : la connexion est établie entre les deux, du coup on voie deux sockets créer chacune dans un sens.

## 

## A la fin de communication : on voit que le client s’est déconnecté, du coup la socket du serveur est en mode TIME\_wait

## 

## Analysez et commentez une capture wireshark avant, pendant et après le lancement du client (en particulier identifiez les valeurs des flags et des numéros de séquences) : expliquez

Avant le lancement du client , il se passe rien le serveur il est en écoute d’une nouvelle connexion , il n y a pas de capture sur Wireshark.

Après le lancement du client , il y a établissement de la communication avec la suite de flags SYN , SYN-ACK, ACK.

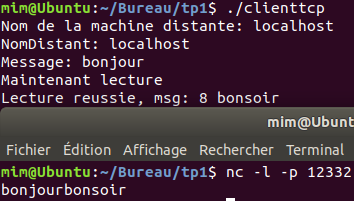


Lors des échanges de messages , c’est des paquets avec le flag PUSH qui sont échangé, avec le numero de Sequence qui correspond au prochain octet en attente. La par exemple le client a envoyé le msg « Hi » donc 2octet , c’est pour qu’on voit un ack = 3.

  
  
lors de la fin de la communication , c’est des messages avec le flag FIN-ACK , FIN , ACK qui sont échangé.



## Utilisez le client pour vous connecter à un serveur lancé avec nc



## Analysez et commentez une capture wireshark

On voit la même chose , que lorsqu’on a utilise no client/serveur :

La phase établissement de la connexion ( les 3 première lignes ).

La phase communication : le client envoi (Hi ) ,le serveur renvoi ack ( et attend le 3eme octet ack=3)

Le serveur envoi (uuuh), le client renvoi ack ( et dit qu’il attend le 6eme octet ( uuuh\0 ) = 5octet donc c’est logique.

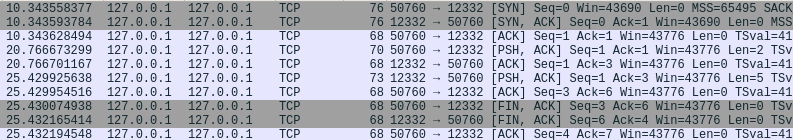
La phase fermeture de la connexion les 3derniere lignes

## Utilisez un client lancé avec nc pour vous connecter au serveur





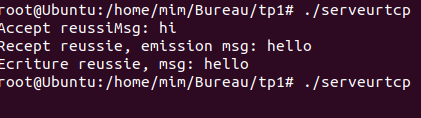
## Analysez et commentez une capture wireshark



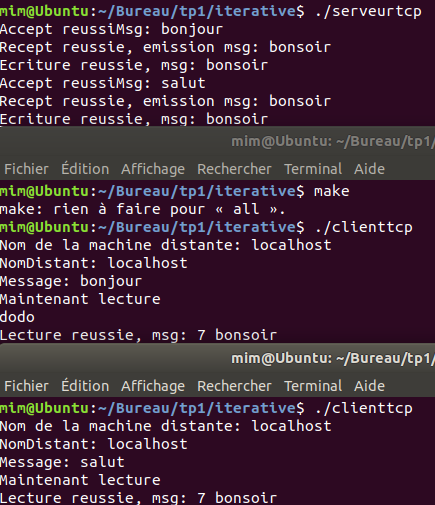
Niveau commentaire c’est exactement la même chose que 1.7 3 phases en total.

## Modifier le serveur pour qu’il utilise l’option SO\_REUSEADDR

Après ajout de l’option :



## Modifier le serveur pour qu’il puisse gérer plusieurs requêtes de clients ITERATIVEMENT



## Modifier le code du serveur et du client pour passer le port du serveur comme argument sur la ligne de commande des deux programmes. (Pour éviter d’avoir à recompiler chaque fois qu’on change le port)



Le code source du resultat sera en pièce jointe avec se document