IUT du Littoral Côte d'Opale
Département Informatique

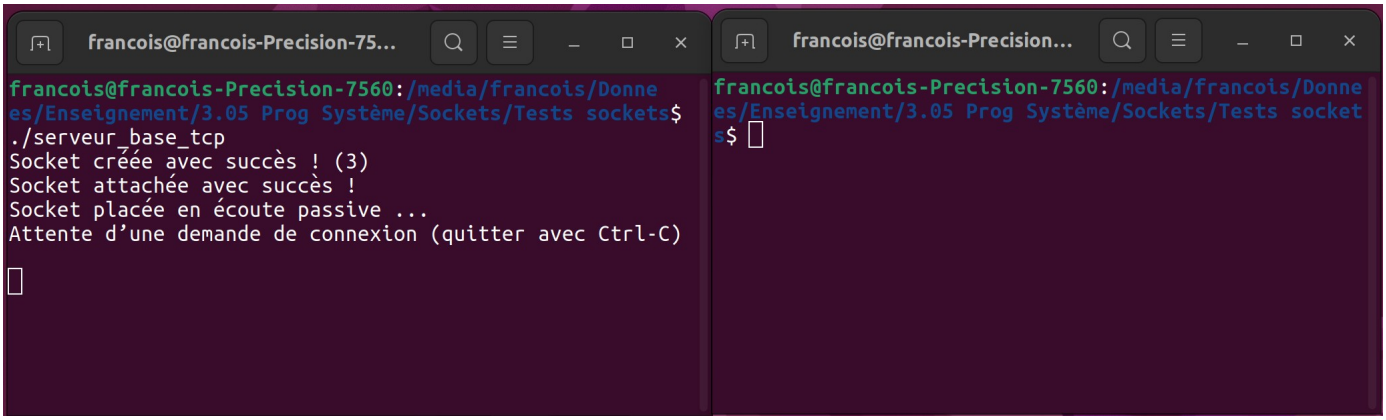
TP Socket
Année 2023- 2024

Le but de ce TP est d'écrire une application qui permet à deux processus d'échanger des données en utilisant le mécanisme des sockets en mode TCP/IP. L'un des processus sera donc le serveur, l'autre le client. L'heure ou la date seront les données échangées entre ces deux processus.

Tout comme les TPs de R3.05, vous ferez ce TP sous Linux et en langage C.

Avant de commencer :

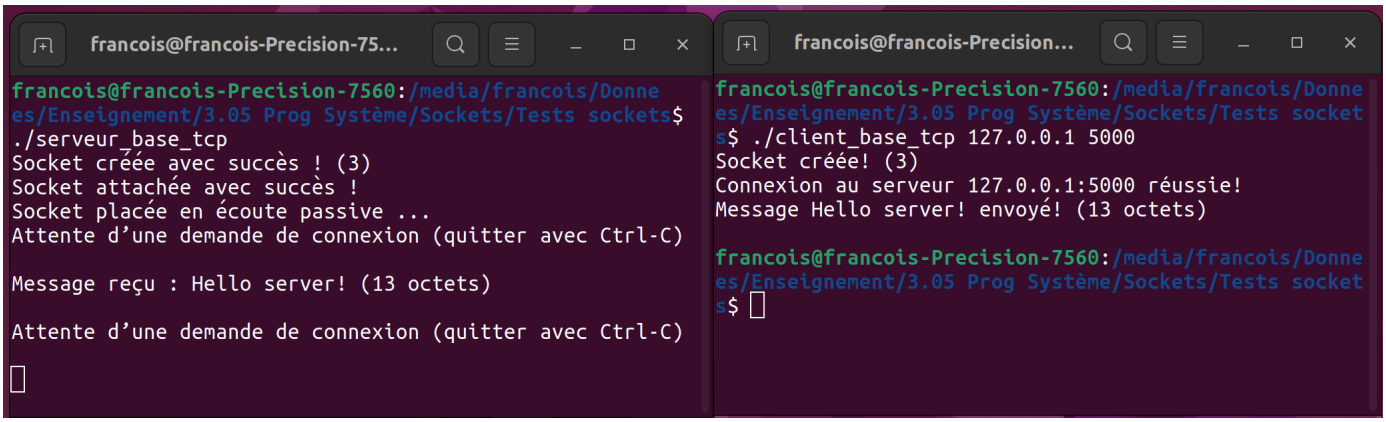
1. Récupérez les 2 fichiers client_base_tcp.c et serveur_base_tcp.c
2. Récupérez les 3 fiches sur le langage C
3. Ouvrir votre dossier contenant les 2 fichiers client et serveur avec VSC et compilez uniquement sans exécuter ces 2 fichiers (si problème, revoir les fiches installation et utilisation du langage C avec VSC)
4. Lancer 2 terminaux côte à côte et allez pour chacun dans le dossier de vos fichiers de programme



5. Exécutez d'abord le serveur dans le premier terminal : *./serveur_base_tcp*
6. Exécutez ensuite le client avec les paramètres suivants : *./client_base_tcp 127.0.0.1 5000*

   127.0.0.1 indique l'adresse IP du serveur à contacter (ici en local)

   5000 indique le port à utiliser (si problème de blocage en ré-utilisant le même port, essayez 5001, ou des ports non assignés comme 5016 à 5019, 5035 à 5041, … Voir : iana - Service Name and Transport Protocol Port Number Registry) – plus d'explications sur stackoverflow et serverfault.



Pour démarrer ce tp, dupliquez les 2 fichiers exemples dans 2 nouveaux fichiers client_date_tcp.c et serveur_date_tcp.c. Vous pouvez compilez et testez vos 2 nouveaux fichiers.

En vous appuyant sur ce code existant, écrire les deux programmes respectant les étapes suivantes et l'organigramme de la page suivante :

Client	**Serveur**
1. Création de la socket de dialogue par le client	1. Création de la socket d'écoute par le serveur
2. Attachement de l'adresse locale à la socket	2. Attachement de l'adresse locale à la socket
3. Demande effectuée par le client au serveur d'une connexion	3. Déclaration du nombre maximum de connexions autorisées
4. Envoi de la demande : heure ou date	4. Attente du serveur d'une demande de connexion
5. Réception de la réponse du serveur	5. Réception de la demande du client
6. Fermeture de la socket de dialogue par le client	6. Traitement de la demande du client
	7. Envoi de la réponse au client
	8. Fermeture de la socket de dialogue par le serveur
	9. Fermeture de la socket d'écoute par le serveur

Pour récupérer l'heure et la date, on utilisera les deux fonctions suivantes, dont le code est donné en annexe :

1. void lire_heure(char* heure) : cette fonction permet de récupérer l'heure
2. void lire_date(char* date) : cette fonction permet de récupérer la date

Remarque : la fonction *popen* utilisée dans ces 2 fonctions permet d'ouvrir un pipe et d'exécuter une commande shell dont la sortie pourra être lue au travers du pipe si l'option « r » est indiquée. Au contraire le pipe pourrait servir à envoyer des données en entrée de la commande si « w » est spécifié. Voir les manpages de *popen* pour plus d'information.

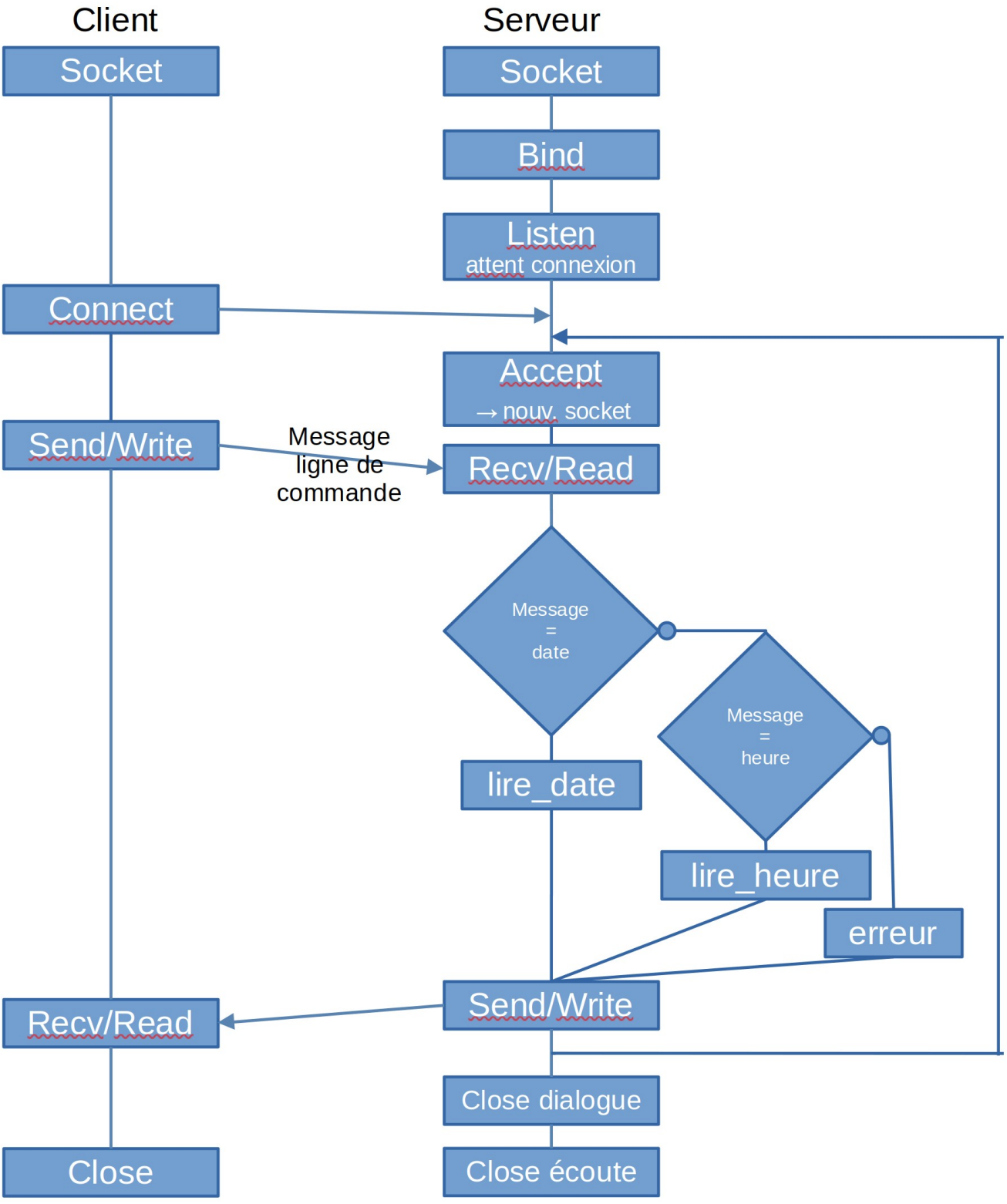
ANNEXE :

```
void lire_heure(char* heure){
    FILE *fpipe;

    fpipe = popen("date '+%X'","r");
    if (fpipe == NULL){
        perror("popen" );
        exit(-1);
    }
    fgets(heure, LG_MESSAGE, fpipe);
    pclose(fpipe);
}
```

```
void lire_date(char* date){
    FILE *fpipe;

    fpipe= popen("date '+%A %d %B %Y'","r");
    if (fpipe == NULL){
        perror("popen" );
        exit(-1);
    }
    fgets(date, LG_MESSAGE, fpipe);
    pclose(fpipe);
}
```


Client
Serveur
Socket
Socket
Bind
Listen
attent connexion
Connect
Accept
→ nouv. socket
Send/Write
Message
ligne de
commande
Recv/Read
Message
=
date
Message
=
heure
lire_date
lire_heure
erreur
Recv/Read
Send/Write
Close dialogue
Close
Close écoute