

Rapport de Travaux Pratiques BTS SN-IR

PNET - TP 1 SOCK TCP & UDP

Flavian LAXENAIRE

22 octobre 2021

Forme / 20	[coef. 1] →		Fond / 20	[coef. 2] →		Note / 20
Qualité du rapport			Méthodologie			
Expression écrite			Respect du Cahier des Charges			
Pertinence de la rédaction			Qualité technique			
Respect des standards de codage			État d'avancement			Malus

Sommaire

1. Codage du serveur.....	2
2. 2. Codage du client.....	3
3. 3. Gestion multi-clients.....	4
4. 4. Codage en UDP.....	5
4.1. Fichier source côté serveur :	5
4.2. Fichier source côté client:.....	5

1. Codage du serveur

Mise en place d'un gestionnaire de signal d'arrêt.

```
signal(SIGINT, arret); //appelle la fonction arret si CTRL C détecté
void arret()
{
    printf(" \nARRET PROG DE TEST SOCKET  ---> echo \n");

    /* Fermer la socket */
    close(sock_SERVEUR);
    kill(getpid(), SIGKILL); // KILL le programme
}
```

Creation d'un socket server :

```
sock_SERVEUR = socket(PF_INET, SOCK_STREAM, 0); //Creation de la socket TCP
```

Renseignement des champs adresse et port de la structure adresse locale :

```
myaddr.sin_addr.s_addr = INADDR_ANY; //accepte toute les connexions
myaddr.sin_port = htons(5000); //port 5000
```

L'attente de connexions clientes :

```
printf("attente de demande connexion\n");
listen(sock_SERVEUR, 2);
```

L'acceptation d'une nouvelle connexion :

```
while (1)
{
    int read_msg;
    signal(SIGINT, arret);
    //TO DO
    new = accept(sock_SERVEUR, (struct sockaddr *)&from, (void *)&len);
    printf("nouveau client sur fd %d\n", new);
    printf("DEBUT serveur\n");
```

Boucle d'envoi et reception :

```
for (;;)
{
    while ((read_msg = read(new, buf, sizeof(buf))) > 0) //Lecture sur la socket client (telnet)
    {
        buf[read_msg] = '\0';
        write(1, buf, strlen(buf)); //Ecriture sur l'écran du serveur
        memset(buf, '\0', sizeof(buf)); //Mise à 0 du buf
    }
}
```

```
snir@snir-21-22:~/Documents/bts_2nd_annee/dev/pnet/tp1$ ./serv_suj2
3attente de demande connexion
nouveau client sur fd 4
DEBUT serveur
yolo
█
```

```
snir@snir-21-22:~$ telnet 127.0.0.1 5000
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
yolo
```

L 88, col 38 Tail

2. 2. Codage du client

Récupération de l'adresse du serveur :

```
hostAddr = (inet_addr("127.0.0.1"));
```

Renseignement du domaine et du port serveur :

```
sin.sin_family = PF_INET;  
sin.sin_port = htons(5000);
```

Creation de la socket :

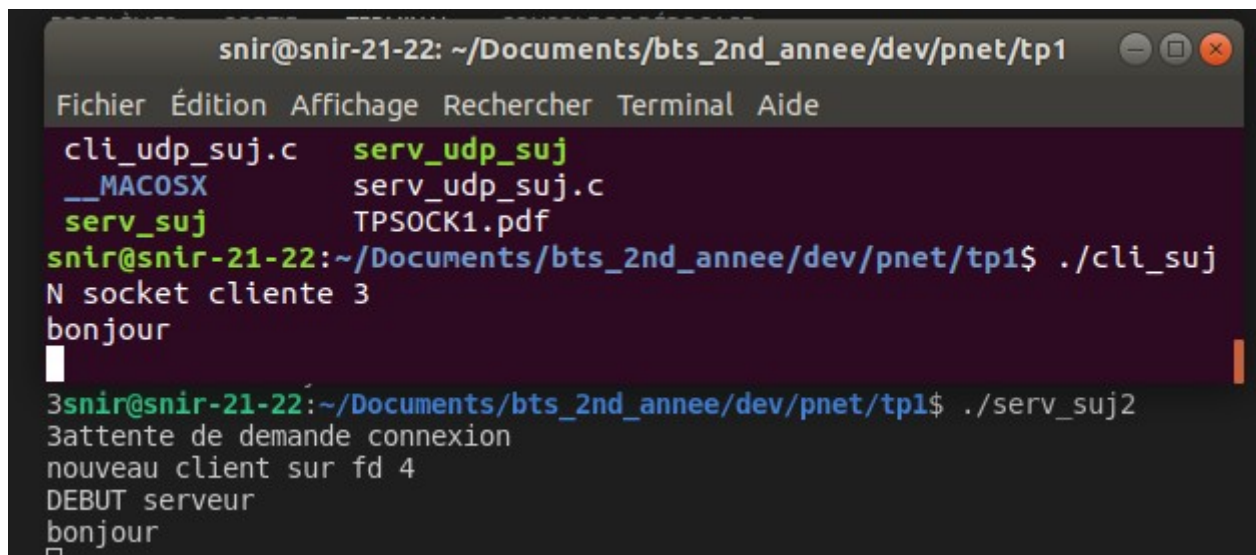
```
sock_CLIENT = socket(PF_INET, SOCK_STREAM, 0);
```

Demande de connexion au serveur :

```
if (connect(sock_CLIENT, (struct sockaddr *)&sin, sizeof(sin)) < 0)  
{  
    perror("client:pb connexion au serveur par socket");  
    exit(-1);  
}
```

Boucle d'envoi et reception :

```
char buf[2000];  
for(;;)  
{  
    while(read(0, buf, 1) > 0)  
    {  
        write(sock_CLIENT, buf, 1);  
    }  
}
```



```
snir@snir-21-22: ~/Documents/bts_2nd_annee/dev/pnet/tp1
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
cli_udp_suj.c  serv_udp_suj
__MACOSX      serv_udp_suj.c
serv_suj      TPSOCK1.pdf
snir@snir-21-22:~/Documents/bts_2nd_annee/dev/pnet/tp1$ ./cli_suj
N socket cliente 3
bonjour
3snir@snir-21-22:~/Documents/bts_2nd_annee/dev/pnet/tp1$ ./serv_suj2
3attente de demande connexion
nouveau client sur fd 4
DEBUT serveur
bonjour
```

3. 3. Gestion multi-clients

```
listen(sock_SERVEUR, 2);
while (1)
{
    int service;
    int read_msg;
    signal(SIGINT, arret);
    //TO DO
    service = accept(sock_SERVEUR, (struct sockaddr *)&from, (void *)&len);
    printf("nouveau client sur fd %d\n", service);
    int pid = fork(); //Duplication du programme pour chaque client
    if (pid == 0)
    {
        close(sock_SERVEUR); //Fermeture de la socket dupliquée
        printf("DEBUT serveur\n");
        int nbclient = nb_client++;

        for (;;)
        {
            while ((read_msg = read(service, buf, sizeof(buf))) > 0)
            {
                buf[read_msg] = '\0';
                write(1, buf, strlen(buf));
                memset(buf, '\0', sizeof(buf));
            }
            signal(SIGINT, arret);
        }
    }
    else if (pid == -1)
    {
        perror("pipe issue");
    }
    else
    {
        close(service);
    }
    if (service < 0)
    {
        perror("accept");
        exit(-1);
    }
}
```

```
snir@snir-21-22: ~/Documents/bts_2nd_annee/dev/pnet/tp1
Fichier Édition Affichage Rechercher Terminal Aide
snir@snir-21-22:~/Documents/bts_2nd_annee/dev/pnet/tp1$ ./cli_suj
N socket cliente 3
bonjour

snir@snir-21-22: ~/Documents/bts_2nd_annee/dev/pnet/tp1
Fichier Édition Affichage Rechercher Terminal Aide
snir@snir-21-22:~/Documents/bts_2nd_annee/dev/pnet/tp1$ ./cli_suj
N socket cliente 3
salut

3snir@snir-21-22:~/Documents/bts_2nd_annee/dev/pnet/tp1$ ./serv_suj2
3attente de demande connexion
nouveau client sur fd 4
DEBUT serveur
nouveau client sur fd 4
DEBUT serveur
bonjour
salut
```

4. 4. Codage en UDP

4.1. Fichier source côté serveur :

```

main(int argc, char **argv)
{
    sin_dest.sin_port = htons(4000); /* Numero de port du client */
    sin_dest.sin_family = PF_INET;
    sin_dest.sin_addr.s_addr = INADDR_ANY;
    sin_src.sin_port = htons(5000); /* Numero de port du serveur */
    hostAddr = (inet_addr("127.0.0.1"));

    if ((long)hostAddr != (long)-1)
    {
        bcopy(&hostAddr, &sin_src.sin_addr.s_addr, sizeof(hostAddr));
        sin_src.sin_family = PF_INET;
    }

    if ((sock_SERVEUR = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP)) < 0)
    {
        perror("client:pb creation socket");
        exit(-1);
    }

    if (bind(sock_SERVEUR, (struct sockaddr *)&sin_src, sizeof(sin_src)) < 0)
    {
        perror("erreur de bind");
        exit(-1);
    }

    for (;;)
    {
        taille = sizeof(sin_dest); // REINIT du champ taille a chaque reception

        /*lecture reponse client */
        for (i = 0; i < sizeof(txt); *(txt + i++) = '\0')
            ; /* init du buffer de reception */
        /* CODER la RECEPTION sur socket recvfrom(      ) */
        pos = recvfrom(sock_SERVEUR, txt, sizeof(txt), 0, (struct sockaddr *)&sin_src, &taille);
        printf("reçu :%s\n", txt);
        /* renvoie vers le client prendre sendto()*/
        sendto(sock_SERVEUR, txt, pos, 0, (struct sockaddr *)&sin_dest, sizeof(sin_dest));
        printf("envoyé :%s\n", txt);
    }
    close(sock_SERVEUR); //fermeture socket serveur
    exit(0);
}

```

4.2. Fichier source côté client:

```
void arret()
{
    printf(" \nARRET PROG CLIENT SOCKET  ---> echo \n");
    /* TODO Q2 Fermer la socket */
    close(sock_CLIENT);
    kill(getpid(), SIGKILL);
}

main(int argc, char **argv)
{
    hostAddr = (inet_addr("127.0.0.1"));
    bcopy(&hostAddr, (struct sockaddr_in *)&sin.sin_addr, sizeof(hostAddr));
    sin.sin_family = PF_INET;
    /* TODO Q2 Renseigner le port du serveur */
    sin.sin_port = htons(5000);
    /* Creation socket */
    sock_CLIENT = socket(PF_INET, SOCK_STREAM, 0);

    if (connect(sock_CLIENT, (struct sockaddr *)&sin, sizeof(sin)) < 0)
    {
        perror("client:pb connexion au serveur par socket");
        exit(-1);
    }
    char buf[2000];
    for(;;)
    {
        while(read(0, buf, 1) > 0)
        {
            write(sock_CLIENT, buf, 1);
        }
    }
    close(sock_CLIENT); /* ferme socket cliente**/
    exit(0);
}
```



The screenshot shows a terminal window titled "snir@snir-21-22: ~/Documents/bts_2nd_annee/dev/pnet/tp1". The window contains the following text:

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
snir@snir-21-22:~/Documents/bts_2nd_annee/dev/pnet/tp1$ ./cli_udp
_suj
N socket cliente 3
salut
envoyé : salut

reçu : salut
[ ]
snir@snir-21-22:~/Documents/bts_2nd_annee/dev/pnet/tp1$ ./serv_udp_suj
N socket serveur 3
reçu :salut
envoyé :salut
```