

# 1) Programme mode non-connecté sans utiliser de fonctions non-bloquante

Juste à prendre 2 clients non-connecté et le serveur attend sur cette socket et reçoit le premier message et ainsi de suite:

```
#include <stdio.h> #include <stdlib.h>
#include <string.h> #include <unistd.h>
#include <sys/types.h> #include <sys/socket.h>
#include "fonctionsSocket.h"

int main(int argc, char *argv[]) {
    char chaine[100];
    int sock; /* descripteur de la socket locale */
    int err; /* code d'erreur */
    /* Verification des arguments */
    if (argc != 3) { printf("usage : client nom_machine no_port\n"); exit(1); }

    sock = socketClient_EAD(argv[1], atoi(argv[2]));
    if (sock < 0) { printf("client: erreur socketClient_EAD\n"); exit(2); }
    /* Saisie de la chaine */
    printf("client: donner une chaine : ");
    scanf("%s", chaine);
    printf("client: envoi de \"%s\"\n", chaine);
    /* Envoi de la chaine */
    size_t len = strlen(chaine) + 1;
    err = send(sock, chaine, len, 0);
    if (err != len) { perror("client: erreur sur le send"); shutdown(sock, 2); exit(3); }
    printf("client : envoi realise\n");
    /* Fermeture de la connexion et de la socket */
    shutdown(sock, 2);
    close(sock);

    return 0;
}
```

```
christophe@christophe-asus:~/CSD/exo/ex10$ ./serveurNB 2500
serveurNB: serveur pret
serveurNB: voila le message recu: "2msg1"
christophe@christophe-asus:~/CSD/exo/ex10$ ./serveurNB 2500
serveurNB: serveur pret
serveurNB: Resource temporarily unavailable
serveurNB: voila le message recu: "hello"
christophe@christophe-asus:~/CSD/exo/ex10$
```

```
christophe@christophe-asus: ~/CSD/exo/ex10
File Edit View Search Terminal Help
client: donner une chaine : 1msg2
client: envoi de "1msg2"
client : envoi realise
christophe@christophe-asus:~/CSD/exo/ex10$ ./client localhos
t 2500
client: donner une chaine : hello
client: envoi de "hello"
client : envoi realise
christophe@christophe-asus:~/CSD/exo/ex10$
```

```
christophe@christophe-asus: ~/CSD/exo/ex10
File Edit View Search Terminal Help
christophe@christophe-asus:~/CSD/exo/ex10$ ./client localhos
t 2500
client: donner une chaine : 2msg1
client: envoi de "2msg1"
client : envoi realise
christophe@christophe-asus:~/CSD/exo/ex10$ ./client localhos
t 2500
client: donner une chaine :
```

```
christophe@christophe-asus:~/CSD/exo/ex10$ ./serveurSelect 2
800
serveurSelect : avant le select
serveur : voila le message recu: "2msg1"
christophe@christophe-asus:~/CSD/exo/ex10$
```

```
christophe@christophe-asus: ~/CSD/exo/ex10
File Edit View Search Terminal Help
christophe@christophe-asus:~/CSD/exo/ex10$ ./client localhos
t 2800
client: donner une chaine : 1msg2
client: envoi de "1msg2"
client : envoi realise
christophe@christophe-asus:~/CSD/exo/ex10$
```

```
christophe@christophe-asus: ~/CSD/exo/ex10
File Edit View Search Terminal Help
christophe@christophe-asus:~/CSD/exo/ex10$ ./client localhos
t 2800
client: donner une chaine : 2msg1
client: envoi de "2msg1"
client : envoi realise
christophe@christophe-asus:~/CSD/exo/ex10$
```

```

#include <string.h> #include <stdio.h> #define LEN 100
#include <stdlib.h> #include <unistd.h>
#include <errno.h> #include <sys/socket.h>
#include <netinet/in.h> #include "fonctionsSocket.h"
#include <sys/ioctl.h> Serveur Non Bloquant

int main(int argc, char *argv[]) {
    int sock_conn;
    int sock_trans1; /* descripteurs des sockets */
    int sock_trans2; /* locales */
    int recu; /* pour savoir si on a recu */
    int on; /* position non bloquante */
    int err; /* code d'erreur */
    char buffer[LEN]; /* buffer de reception */
    /* Verification des arguments */
    if (argc != 2) { printf("Usage: serveurNB no_port\n"); exit(1); }
    sock_conn = socketServeur_EAD(atoi(argv[1]));
    if (sock_conn < 0) { printf("err socketServeur\n"); exit(2); }
    /* Attente socket transmission */
    sock_trans1 = accept(sock_conn, NULL, NULL);
    if (sock_trans1 < 0) { perror("err sur accept 1"); exit(3); }

    sock_trans2 = accept(sock_conn, NULL, NULL);
    if (sock_trans1 < 0) { perror("err sur accept 2"); exit(4); }
    /* Les sockets sont rendues non bloquantes */
    on = 1;
    err = ioctl(sock_trans1, FIONBIO, &on);
    if (err < 0) { perror("serveurNB: erreur ioctl 1"); exit(5); }

    err = ioctl(sock_trans2, FIONBIO, &on);
    if (err < 0) { perror("serveurNB: erreur ioctl 2"); exit(6); }
    printf("serveurNB: serveur pret\n");
    /* Attente de la reception */
    recu = 0;
    while (recu == 0) { /* Reception et affichage du premier
    message en provenance du client */
        err = recv(sock_trans1, buffer, LEN, 0);
        if (err > 0) {
            perror("serveurNB");
            recu = 1;
            break;
        } else if ((err < 0) && (errno != EWOULDBLOCK)) {
            perror("serveurNB: erreur dans la reception 1");
            shutdown(sock_trans1, 2);
            exit(7);
        }
        /* Reception et affichage du deuxieme message en
    provenance du client */
        err = recv(sock_trans2, buffer, LEN, 0);
        if (err > 0) {
            recu = 1;
            break;
        } else if ((err < 0) && (errno != EWOULDBLOCK)) {
            perror("serveurNB: erreur dans la reception 2");
            shutdown(sock_trans2, 2);
            exit(8);
        }
    }
    printf("serveurNB: voila le message recu: \"%s\"\n", buffer);
    /* Arret de la connexion et fermeture des sockets */
    shutdown(sock_trans1, 2); shutdown(sock_trans2, 2);
    close(sock_trans1); close(sock_trans2);
    close(sock_conn);
    return 0;
}

```

```

#include <string.h> #include <stdio.h> #define LEN 100
#include <stdlib.h> #include <unistd.h>
#include <sys/socket.h> #include <netinet/in.h>
#include "fonctionsSocket.h" Serveur Select

int main(int argc, char *argv[]) {
    int sock_conn;
    int sock_trans1; /* descripteurs des sockets locales */
    int sock_trans2; /* descripteurs des sockets locales */
    int err; /* code d'erreur */
    char buffer[LEN]; /* buffer de reception */
    fd_set read_set; /* ensemble de desc pour select */
    /* Verification des arguments */
    if (argc != 2) { printf("usage : serveur no_port\n"); exit(1); }
    sock_conn = socketServeur_EAD(atoi(argv[1]));
    if (sock_conn < 0) { printf("err socketServeur_EAD\n"); exit(2); }
    /* Attente des connexions */
    sock_trans1 = accept(sock_conn, NULL, NULL);
    if (sock_trans1 < 0) { perror("err sur accept 1"); exit(3); }
    sock_trans2 = accept(sock_conn, NULL, NULL);
    if (sock_trans2 < 0) { perror("err sur accept 2"); exit(4); }

    FD_ZERO(&read_set); //mise dans l'ensemble
    FD_SET(sock_trans1, &read_set);
    FD_SET(sock_trans2, &read_set);

    printf("serveurSelect : avant le select\n");

    err = select(FD_SETSIZE, &read_set, NULL, NULL, NULL);
    if (err < 0) { perror("erre dans select"); exit(5); }
    //test si le descripteur est prêt après un appel a select
    if (FD_ISSET(sock_trans1, &read_set) != 0) {
        /* Reception du message */
        err = recv(sock_trans1, buffer, LEN, 0);
        if (err < 0) {
            perror("serveurSelect : Erreur dans le recv 1");
            exit(6);
        }
    } else if (FD_ISSET(sock_trans2, &read_set) != 0) {
        /* Reception du message */
        err = recv(sock_trans2, buffer, LEN, 0);
        if (err < 0) {
            perror("serveurSelect : Erreur dans le recv 2");
            exit(7);
        }
    }
    /* Affichage de la chaine */
    printf("serveur : voila le message recu: \"%s\"\n",
    buffer);

    /* Arret de la connexion et fermeture */
    shutdown(sock_trans1, 2);
    shutdown(sock_trans2, 2);
    close(sock_trans1);
    close(sock_trans2);
    close(sock_conn);
    return 0;
}

```

