Travaux Pratiques

Les sockets

Exemple 1 : Exemple de programme client-serveur en C (l'aide de sockets TCP)

Code du Server

```
/****************** SERVER CODE ****************/


#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>


int main(){

  int welcomeSocket, newSocket;

  char buffer[1024];

  struct sockaddr_in serverAddr;

  struct sockaddr_storage serverStorage;

  socklen_t addr_size;


  /*---- Create the socket. The three arguments are: ----*/

  /* 1) Internet domain 2) Stream socket 3) Default protocol (TCP in this case) */

  welcomeSocket = socket(PF_INET, SOCK_STREAM, 0);


  /*---- Configure settings of the server address struct ----*/

  /* Address family = Internet */

  serverAddr.sin_family = AF_INET;

  /* Set port number, using htons function to use proper byte order */

  serverAddr.sin_port = htons(7891);

  /* Set IP address to localhost */
```

```
  serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
  /* Set all bits of the padding field to 0 */
  memset(serverAddr.sin_zero, '\0', sizeof serverAddr.sin_zero);

  /*---- Bind the address struct to the socket ----*/
  bind(welcomeSocket, (struct sockaddr *) &serverAddr, sizeof(serverAddr));

  /*---- Listen on the socket, with 5 max connection requests queued ----*/
  if(listen(welcomeSocket,5)==0)
    printf("Listening\n");
  else
    printf("Error\n");

  /*---- Accept call creates a new socket for the incoming connection ----*/
  addr_size = sizeof serverStorage;
  newSocket = accept(welcomeSocket, (struct sockaddr *) &serverStorage,
&addr_size);

  /*---- Send message to the socket of the incoming connection ----*/
  strcpy(buffer,"Hello World\n");
  send(newSocket,buffer,13,0);

  return 0;
}
```

Code du Client

```
/****************** CLIENT CODE ***************/


#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>


int main(){

  int clientSocket;

  char buffer[1024];

  struct sockaddr_in serverAddr;

  socklen_t addr_size;


  /*---- Create the socket. The three arguments are: ----*/

  /* 1) Internet domain 2) Stream socket 3) Default protocol (TCP in this case) */

  clientSocket = socket(PF_INET, SOCK_STREAM, 0);


  /*---- Configure settings of the server address struct ----*/

  /* Address family = Internet */

  serverAddr.sin_family = AF_INET;

  /* Set port number, using htons function to use proper byte order */

  serverAddr.sin_port = htons(7891);

  /* Set IP address to localhost */

  serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");

  /* Set all bits of the padding field to 0 */

  memset(serverAddr.sin_zero, '\0', sizeof serverAddr.sin_zero);


  /*---- Connect the socket to the server using the address struct ----*/
```

```
  addr_size = sizeof serverAddr;

  connect(clientSocket, (struct sockaddr *) &serverAddr, addr_size);


  /*---- Read the message from the server into the buffer ----*/

  recv(clientSocket, buffer, 1024, 0);


  /*---- Print the received message ----*/

  printf("Data received: %s",buffer);


  return 0;

}
```

pour tester le code il suffit de lancer le serveur sur un terminal, puis exécutez le client sur un autre terminal (ou exécuter le serveur comme un processus d'arrière-plan, puis exécuter le client sur le même terminal).

Exemple 2

Que fait ce programme ? expliquer ? essayer de compléter le code qui manque

```
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define MYPORT 3334

int main()
{
int sockfd;  /* socket file descriptor */
struct sockaddr_in my_addr;

sockfd = socket(AF_INET, SOCK_STREAM, 0);
if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1)
{
  perror("Server-socket() error lol!");
  exit(1);
}
else
  printf("Server-socket() sockfd is OK...\n");
```

```
/* host byte order */
my_addr.sin_family = AF_INET;
/* short, network byte order */
my_addr.sin_port = htons(MYPORT);
my_addr.sin_addr.s_addr = INADDR_ANY;
/* zero the rest of the struct */
memset(&(my_addr.sin_zero), 0, 8);

if(bind(sockfd, (struct sockaddr *)&my_addr, sizeof(struct
sockaddr)) == -1)
{
  perror("Server-bind() error lol!");
  exit(1);
}
else
  printf("Server-bind() is OK...\n");

/*....other codes....*/

return 0;
}
```

Exemple 3

Compiler et exécuter le programme suivant :

```
/************tcpclient.c************************/
/* Header files needed to use the sockets API. */
/* File contains Macro, Data Type and */
/* Structure definitions along with Function */
/* prototypes. */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <unistd.h>
#include <errno.h>
/* BufferLength is 100 bytes */
#define BufferLength 100
/* Default host name of server system. Change it to your
default */
/* server hostname or IP.  If the user do not supply the
hostname */
/* as an argument, the_server_name_or_IP will be used as
default*/
```

```
#define SERVER "The_server_name_or_IP"
/* Server's port number */
#define SERVPORT 3111

/* Pass in 1 parameter which is either the */
/* address or host name of the server, or */
/* set the server name in the #define SERVER ... */
int main(int argc, char *argv[])
{
/* Variable and structure definitions. */
int sd, rc, length = sizeof(int);
struct sockaddr_in serveraddr;
char buffer[BufferLength];
char server[255];
char temp;
int totalcnt = 0;
struct hostent *hostp;
char data[100] = "This is a test string from client lol!!! ";

/* The socket() function returns a socket */
/* descriptor representing an endpoint. */
/* The statement also identifies that the */
/* INET (Internet Protocol) address family */
/* with the TCP transport (SOCK_STREAM) */
/* will be used for this socket. */
/******************************************/
/* get a socket descriptor */
if((sd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
{
perror("Client-socket() error");
exit(-1);
}
else
printf("Client-socket() OK\n");
/*If the server hostname is supplied*/
if(argc > 1)
{
/*Use the supplied argument*/
strcpy(server, argv[1]);
printf("Connecting to the f***ing %s, port %d ...\n", server,
SERVPORT);
}
else
/*Use the default server name or IP*/
strcpy(server, SERVER);

memset(&serveraddr, 0x00, sizeof(struct sockaddr_in));
serveraddr.sin_family = AF_INET;
serveraddr.sin_port = htons(SERVPORT);
```

```
if((serveraddr.sin_addr.s_addr = inet_addr(server)) ==
(unsigned long)INADDR_NONE)
{

/* When passing the host name of the server as a */
/* parameter to this program, use the gethostbyname() */
/* function to retrieve the address of the host server. */
/***************************************************/
/* get host address */
hostp = gethostbyname(server);
if(hostp == (struct hostent *)NULL)
{
printf("HOST NOT FOUND --> ");
/* h_errno is usually defined */
/* in netdb.h */
printf("h_errno = %d\n",h_errno);
printf("---This is a client program---\n");
printf("Command usage: %s <server name or IP>\n", argv[0]);
close(sd);
exit(-1);
}
memcpy(&serveraddr.sin_addr, hostp->h_addr,
sizeof(serveraddr.sin_addr));
}

/* After the socket descriptor is received, the */
/* connect() function is used to establish a */
/* connection to the server. */
/***********************************************/
/* connect() to server. */
if((rc = connect(sd, (struct sockaddr *)&serveraddr,
sizeof(serveraddr))) < 0)
{
perror("Client-connect() error");
close(sd);
exit(-1);
}
else
printf("Connection established...\n");

/* Send string to the server using */
/* the write() function. */
/***************************************/
/* Write() some string to the server. */
printf("Sending some string to the f***ing %s...\n", server);
rc = write(sd, data, sizeof(data));

if(rc < 0)
{
perror("Client-write() error");
rc = getsockopt(sd, SOL_SOCKET, SO_ERROR, &temp, &length);
if(rc == 0)
```

```
{
/* Print out the asynchronously received error. */
errno = temp;
perror("SO_ERROR was");
}
close(sd);
exit(-1);
}
else
{
printf("Client-write() is OK\n");
printf("String successfully sent lol!\n");
printf("Waiting the %s to echo back...\n", server);
}

totalcnt = 0;
while(totalcnt < BufferLength)
{

/* Wait for the server to echo the */
/* string by using the read() function. */
/*************************************/
/* Read data from the server. */
rc = read(sd, &buffer[totalcnt], BufferLength-totalcnt);
if(rc < 0)
{
perror("Client-read() error");
close(sd);
exit(-1);
}
else if (rc == 0)
{
printf("Server program has issued a close()\n");
close(sd);
exit(-1);
}
else
totalcnt += rc;
}
printf("Client-read() is OK\n");
printf("Echoed data from the f***ing server: %s\n", buffer);

/* When the data has been read, close() */
/* the socket descriptor. */
/*************************************/
/* Close socket descriptor from client side. */
close(sd);
exit(0);
return 0;
}
```

.