① Code du client en C

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <errno.h>
#include <arpa/inet.h>

int main(int argc, char **argv) {
  char *IP = "127.0.0.1";
  u_short server_port = 9999;

  int res;
  char *buff = "HELLO";

  int fd = socket(AF_INET, SOCK_STREAM, 0);
  if (fd < 0) {
    perror("Client: socket() failed");
    exit(1);
  }

  struct sockaddr_in addr;
  memset(&addr, 0, sizeof(addr));

  addr.sin_family = AF_INET;
  addr.sin_port = htons(server_port);
  inet_pton(AF_INET, IP, &addr.sin_addr);

  res = connect(fd,
                (struct sockaddr *) &addr,
                sizeof(addr));
  if (res < 0) {
    fprintf(stderr, "Cannot connect: %s\n",
            strerror(errno));
    exit(1);
  }

  // send(fd, buffer, strlen(buffer), 0);
  int todo = strlen(buff);
  char *p = buff;
  while (todo>0) {
    int res = send(fd, p, todo, 0);
    if (res == -1) {
      perror("Client: cannot send message");
      exit(1);
    }
    p += res;
    todo -= res;
  }

  shutdown(fd, SHUT_RDWR);
  close(fd);
  return 0;
}
```

② Code du serveur en C

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <netdb.h>

#define fail_if(cond,msg) \
  if (cond) {             \
    perror(msg);          \
    exit(1);              \
  }

int main(int argc, char **argv) {
  u_short port = 9999;

  char buff[512];
  int res;

  int master = socket(AF_INET, SOCK_STREAM, 0);
  fail_if( master<0, "Server: error socket");

  int on = 1;
  res = setsockopt(master, SOL_SOCKET,
              SO_REUSEADDR, &on, sizeof(on));
  fail_if(res<0, "Server: error setsockopt");

#ifdef SO_REUSEPORT
  res = setsockopt(master, SOL_SOCKET,
              SO_REUSEPORT, &on, sizeof(on));
  fail_if(res<0, "Server: error setsockopt");
#endif

  struct sockaddr_in serv_addr;
  memset(&serv_addr, 0, sizeof(serv_addr));
  serv_addr.sin_family = AF_INET;
  serv_addr.sin_port = htons(port);
  serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);

  res = bind(master,
             (struct sockaddr *) &serv_addr,
             sizeof(struct sockaddr_in));
  fail_if(res<0, "Server: error bind");


  res = listen(master, SOMAXCONN);
  fail_if(res<0, "Server: error listen");


  fprintf(stderr, "Server ready\n");

  int clilen = sizeof(struct sockaddr_in);
  struct sockaddr_in cli_addr;
  int client_socket =
    accept(master,
           (struct sockaddr *) &cli_addr,
           (socklen_t *) & clilen);
  fail_if(client_socket < 0,
          "Server: error accept");

  res = recv(client_socket, buff, 512, 0);
  fail_if(res <0, "Server: recv() failed");

  buff[res] = '\0';
  printf("Got '%s'\n", buff);

  shutdown(client_socket, SHUT_RDWR);
  close(client_socket);
  return 0;
}
```

③ Structures de données

```c
struct sockaddr { // Generic
    unsigned short    sa_family;
    char              sa_data[14];        };
struct sockaddr_in { // IPv4
    u_short        sin_family;
    u_short        sin_port;
    struct in_addr sin_addr;
    char           sin_zero[8];           };
struct sockaddr_in6 { // IPv6
    u_int16_t       sin6_family;
    u_int16_t       sin6_port;
    u_int32_t       sin6_flowinfo;
    struct in6_addr sin6_addr;
    u_int32_t       sin6_scope_id;        };
```

① Code du client en python

```python
import socket

with socket.socket(socket.AF_INET,
                   socket.SOCK_STREAM) as s:
    s.connect(('127.0.0.1', 9999))
    s.sendall(b'Hello, world')
```

② Code du serveur en Python

```python
import socket

s = socket.socket(socket.AF_INET,
                  socket.SOCK_STREAM)
s.bind( ('127.0.0.1', 9999) )
s.listen()

clisock, address = s.accept()
msg = clisock.recv(255)
print(f"I received {msg} from {address}")

clisock.close()
s.close()
```