This function call creates a socket and returns a socket descriptor. The domain parameter specifies a communication domain; this selects the protocol family which will be used for communication. These families are defined in <sys/socket.h>. In this program, the domain AF_INET is used. The socket has the indicated type, which specifies the communication semantics. SOCK_STREAM type provides sequenced, reliable, two-way, connection based byte streams. The protocol field specifies the protocol used. We always use 0. If the system call is a failure, a -1 is returned. The header files used are sys/types.h and sys/socket.h.

2. Filling the fields of the server address structure.

The socket address structure is of type struct sockaddr_in.

```
struct sockaddr_in {

u_short sin_family;
u_short sin_port;
struct in_addr sin_addr;
char sin_zero[8]; /*unused, always zero*/
};
struct in_addr {

u_long s_addr;

};
```

The fields of the socket address structure are
sin_family which in our case is AF_INET
sin_port which is the port number where socket binds
sin_addr which is the IP address of the server machine

The header file that is to be used is **netinet/in.h**

```
struct sockaddr_in servaddr;
servaddr.sin_family = AF_INET;
servaddr.sin_port = htons(port_number);
```

Why htons is used ?. Numbers on different machines may be represented differently (big-endian machines and little-endian machines). In a little-endian machine the low order byte of an integer appears at the lower address; in a big-endian machine instead the low order byte appears at the higher address. Network order, the order in which numbers are sent on the internet is big-endian.