```
/* H****************************************************************************************************
*
* File:       SFTPServer.c
*
* Author1:     Mani Bhargavi Ketha
* Author2:     Misha Yalavarthy
*
* Description: The program uses the Simple File Transfer Protocol to send ASCII and binary files
*                  from the client to the server.
*                 The server accepts two arguments which include the server executable code name
*                 and <port number>.
*                 The server must first receive the output file name from the client and create that file
*                  and then write into that file the contents of
*                  the input file which the client sends.The server must write into the output file in
*                   chunk sizes of 5 bytes.
*                  The server must be able to identify when the file transfer from the client side is
*                   completed and it must terminate its connection after
*                  closing the file.
*
* Date:      October 15,2015
*
*
*
*H*/


#include<stdio.h>
#include<stdlib.h>
#include<sys/socket.h>          // Header defines the socket address stucture
#include<sys/types.h>
#include<netdb.h>      // Header defines the hostent structure
#include<string.h>

int main(int argc, char **argv)
{
   //initializing all the variables
          int sd,b,cd,i=0,r;                    //sd is the socket descriptor returned by the socket function

          /*fname and buff are buffers which hold the value of the output file name and contents on
inputfile respectively*/

          char fname[50],buff[10],temp,temp1;

          char *ptr;

          struct sockaddr_in caddr,saddr;

          FILE *fp;
          socklen_t clen=sizeof(caddr);
```

```
        if(argc != 2)               //checking whether two arguments have been entered
        {
                printf("\nUSAGE: $Server<PORT>\n");
                exit(0);
        }

    sd=socket(AF_INET,SOCK_STREAM,0);
// socket function , AF_INET is for ipv4 type and SOCK_STREAM is used for TCP protocol


    if(sd!=-1)
//socket file descriptor returns two values 0(socket created),-1(socket not created)
        {
                printf("Socket is created\n");
        }
        else
        {
                printf("Socket is not created\n");
        }
//socket has been created

    //initializing address structures
        saddr.sin_family=AF_INET;
        saddr.sin_addr.s_addr=htonl(INADDR_ANY);
        saddr.sin_port=htons(atoi(argv[1]));

        b=bind(sd,(struct sockaddr *) &saddr,sizeof(saddr)); //bind function

    //binding the socket to the port
    if(b==0)
                printf("Bind socket\n");
        else
        {
                printf("Socket not binded\n");
                return -1;
        }

    //check to see if the server is listening
        if(listen(sd,5) != -1) //-1 is used for error checking throughout the program
                printf("Server is listening\n");
        else
        {
                printf("Server not listening\n");
                return -1;
        }


        cd = accept(sd ,(struct sockaddr *) &caddr, &clen);  //connect function
        if(cd!=0)
        {
                printf("Accepted connection to client\n");
        }

        if(read(cd,fname,sizeof(fname),0) != 0)
//The server reads the name of the output file which is to be created into fname buffer
                printf("File recieving initiated\n");

        printf("\nOutput file : %s",fname);

   //Open the file that we wish to create with the output name
   fp = fopen(fname,"w");

   read(cd,buff,sizeof(buff),0);// The server reads the input file contents into buff buffer

   //goes through the loop to write the file on the server side
  /* the server waits for the message "success" to appear in the buffer uptil which point it
continues writing into the file*/
        while(strcmp(buff,"success")!=0)
        {
                ptr = buff;
                /*Server writes in chunks of 5 bytes into the output file. buff has 10 bytes of data.
first write the data in 5 bytes. */
             //Then copy the remaining data in pointer ptr by pointing it to 6th position
                fwrite(ptr,sizeof(char),5,fp);  //using fwrite function to write first 5 chunks data
                if(buff[5] != '\0')
                {
                        ptr = &buff[5];
                        fwrite(ptr,sizeof(char),5,fp);
 // usinf fwrite function writing second 5 chunks data
                }

                bzero(buff,sizeof(buff));
                read(cd,buff,sizeof(buff),0);
        }
/*when the message "success" is read by the server it completes the fwrite function and closes
the file and terminates the socket connection*/

        fclose(fp);//close the file

   //output once the file transer has been completed
   printf("\nThe file has been recieved\n");

        close(sd);      //closing the socket
        close(cd);      //terminating the connection
        return 0;
}
```