Assignment -1

Members - Ashutosh Jha (11811) & Tushar Sethi (11840)

Docs Link - Google Docs

Github Link - Github

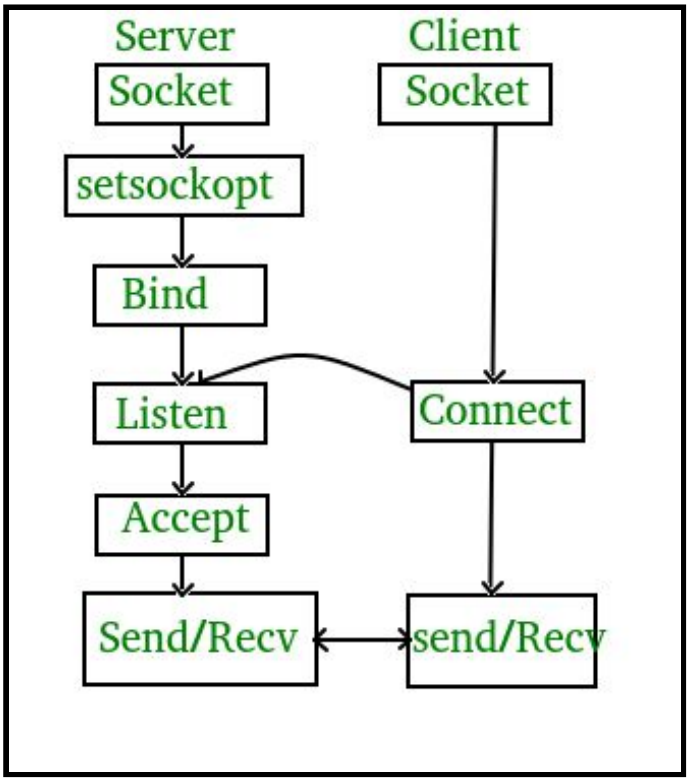
Problem Statement :

Learn the fundamentals of socket programming and implement the following.

1. Create a socket, bind it to a specific address and port, as well as send and receive a TEXT from your machine and receive from the server machine.
2. Implement the functionality of sending files and receiving files with your socket architecture implementing client server paradigm.
3. Implement functionality to exchange files type such as text/images/audio/video.
4. Use only a berkeley socket interface

Client - Server Model

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while another socket reaches out to the other to form a connection. Server forms the listener socket while the client reaches out to the server.



Different stages of Server

1) Socket
2) Bind
3) Listen
4) Accept
5) Read
6) Write
7) Close

Different stages of Client

1) Socket
2) Connect
3) Write
4) Read
5) Close

Server.c

We have included the following header files in server.c

1) stdio.h - Contains declarations for input and output
2) stdlib.h - Defines 4 variable types, several macros and various functions. We will use atoi function to convert the string pointed to, by the argument to an integer
3) string.h - defines one variable type, one macro, and various functions for manipulating arrays of characters

4) unistd.h - for read, write and close
5) sys/types.h - contains definitions of a no. of data types used in system calls
6) netinet/in.h - contains constants and structures needed for internet domain addresses e.g. sockaddr_in
7) sys/socket.h - definitions of structures needed for sockets e.g. defines sockaddr structure
8) netdb.h - contains definition of Hostent structure
9) ctype.h - declares several functions that are useful for testing and mapping characters.
10) sys/stat.h - defines the structure of the data returned by the function fstat()
11) fcntl.h - defines the following requests and arguments for use by the functions fcntl() and open()
12) arpa/inet.h - definitions for internet operations
13) errno.h - contains system error numbers

Error function to exit the program

We have made an error function to exit the program whenever there is an error. We have used perror which is an inbuilt function that interprets the error no. and outputs the output error description by using STDERR

Main function

1) To run the server the user needs to provide 2 arguments, first one the file name, and secondly the port no.

2) First we check whether the user has provided two arguments or not. If not, then we terminate the program.

3) Then, we declare different variables.

4) socket() function is used to create the socket. Socket function returns a file descriptor, if it's less than 0 then there is an error opening socket. This function has 3 arguments, first one is domain, here we will use AF_INET which is an IPv4 protocol, second argument is type, we will use SOCK_STREAM as it for TCP protocol, If we were to use UDP protocol, we would have provide SOCK_DGRAM as second argument, third argument is protocol, which is 0 for TCP
5) If socket() returns less than 0, then we terminate the program, otherwise we continue.

6) Then, we use atoi function, to convert the string pointed to, by the argument to an integer. So we convert the argv[1] to integer and store it in portno

7) Then we get info such as port, using htons - host to network short, converts port no. in host byte order to port no. in network byte order
8) Then we use bind() to assign the address specified by addr. If this file descriptor returns less than 0, then binding fails and we terminate the program
9) Then we use listen(), listen marks the socketfd as passive socket i.e. A socket that will be used for accepting incoming connections using accept
10) Then we specify no. of subsequent client connections, in our case it is five.
11) Then we store the file descriptor returned by accept function in client socket. accept function waits for connect and whenever connect is triggered by client, accept is called by host

12) Then we ask client what operation he/she wants to perform
13) switch is used to handle different operations and sub-operations
14) If the operation code is 1, then the user has selected to send a file. We ask user what type of file he/she wants to send the server
15) A sub-operation code is further read by the server.
16) If the sub-operation code is 1, it means the user wants to send a text file. We read the filename given by the user and start executing code for receiving the text file.
17) Whatever filename the client has sent, we manipulate the file name, as the last 4 chars are the extension, so we remove the extension from the name.
18) We create a new file and then open it and then we keep writing the words through a buffer.
19) We can also count the no. of words and can print every word of the document
20) So we will keep sending the file through buffer until EOF is encountered, meaning there is nothing left to receive
21) In this manner, we receive a text file
22) For other types of file, the implementation is similar. We use memset function. memset copies the character 0 (an unsigned char) to the first 1024 characters of the string pointed to, by the argument buffer. We You'd write the buffer chunks to a file. We first, open an output file for writing. Then, after the read(), we write our bytes. As for chunking, TCP takes care of splitting the stream into packets for us. Once there is nothing left to write or we encounter EOF, we stop and the file is received
23) The other functionality is a chat system. First we clear the buffer as we stream data through the buffer only. Then we use read and write to send messages to and fro. fgets is used to read a line from the specified stream and stores it into the string pointed to by buffer.

24) “Bye” keyword can be used to anytime close the chat option
25) Finally we close all the sockets.

Client.c

We have included the following header files in server.c

1) stdio.h - Contains declarations for input and output
2) stdlib.h - Defines 4 variable types, several macros and various functions. We will use atoi function to convert the string pointed to, by the argument to an integer
3) string.h - defines one variable type, one macro, and various functions for manipulating arrays of characters
4) unistd.h - for read, write and close
5) sys/types.h - contains definitions of a no. of data types used in system calls
6) netinet/in.h - contains constants and structures needed for internet domain addresses e.g. sockaddr_in
7) sys/socket.h - definitions of structures needed for sockets e.g. defines sockaddr structure
8) netdb.h - contains definition of Hostent structure
9) ctype.h - declares several functions that are useful for testing and mapping characters.
10) sys/stat.h - defines the structure of the data returned by the function fstat()
11) fcntl.h - defines the following requests and arguments for use by the functions fcntl() and open()
12) arpa/inet.h - definitions for internet operations

Error function to exit the program

We have made an error function to exit the program whenever there is an error. We have used perror which is an inbuilt function that interprets the error no. and outputs the output error description by using STDERR

Main function

1) To run the client the user needs to provide 3 arguments, The user need to provide 3 arguments, first one the file name, second one -the ip address of server and lastly the port no. first one the file name.
2) If the client is running in the same machine as server, then we provide ip as 127.0.0.1, otherwise we provide the ip of server which can be found out using ifconfig
3) First we check whether the user has provided three arguments or not. If not, then we terminate the program.
4) Then, we declare different variables.
5) socket() function is used to create the socket. Socket function returns a file descriptor, if it's less than 0 then there is an error opening socket. This function has 3 arguments, first one is domain, here we will use AF_INET which is an IPv4 protocol, second argument is type, we will use SOCK_STREAM as it for TCP protocol, If we were to use UDP protocol, we would have provide SOCK_DGRAM as second argument, third argument is protocol, which is 0 for TCP
6) If socket() returns less than 0, then we terminate the program, otherwise we continue.
7) Then, we use atoi function, to convert the string pointed to, by the argument to an integer. So we convert the argv[1] to integer and store it in portno

8) We then get info about the host using gethostbyname
9) Then we get info such as port, using htons - host to network short, converts port no. in host byte order to port no. in network byte order
10) Them we connect to host, if file descriptor is less than 0, then the operation failed and we terminate
11) Once the client connects to server, Then we ask client what operation he/she wants to perform
12) switch is used to handle different operations and sub-operations
13) If the operation code is 1, then the user has selected to send a file. We ask user what type of file he/she wants to send the server
14) We further send a sub operation code to server using write
15) If the sub-operation code is 1, it means the user wants to send a text file. We read the filename given by the user and start executing code for sending the text file.
16) We open the file that we wish to send and then we keep reading the words and send it through buffer
17) We can also count the no. of words and can print every word of the document and moreover, we can get the file size using fseek and ftell functions
18) So we will keep sending the file through buffer until EOF is encountered, meaning there is nothing left to receive
19) In this manner, we receive a text file
20) For other types of file, the implementation is similar. We use the memset function. memset copies the character 0 (an unsigned char) to the first 1024 characters of the string pointed to, by the argument buffer. We write the buffer chunks to a file. We first, open input file for reading. Then, after the read(), we send our bytes. As for chunking, TCP takes care of splitting the stream into packets for us. Once there is nothing left to write or we encounter EOF, we stop and the file is sent

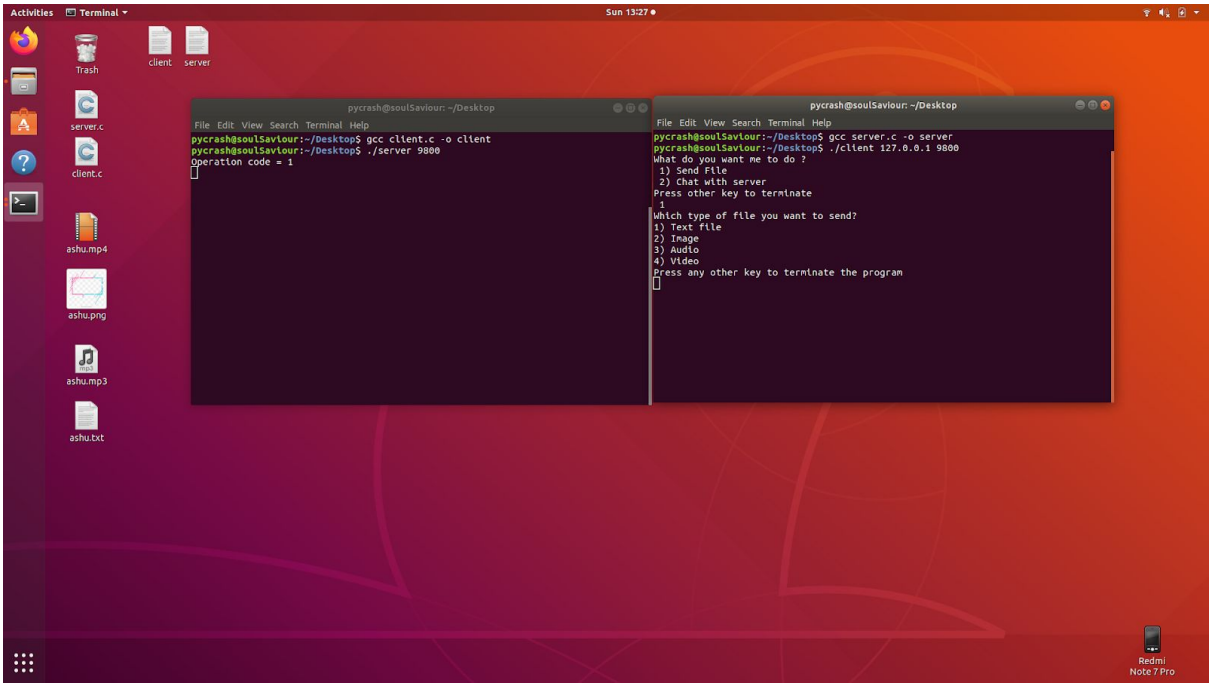
21) The other functionality is a chat system. First we clear the buffer as we stream data through the buffer only. Then we use read and write to send messages to and fro. fgets is used to read a line from the specified stream and stores it into the string pointed to by buffer.

22) “Bye” keyword can be used to anytime close the chat option

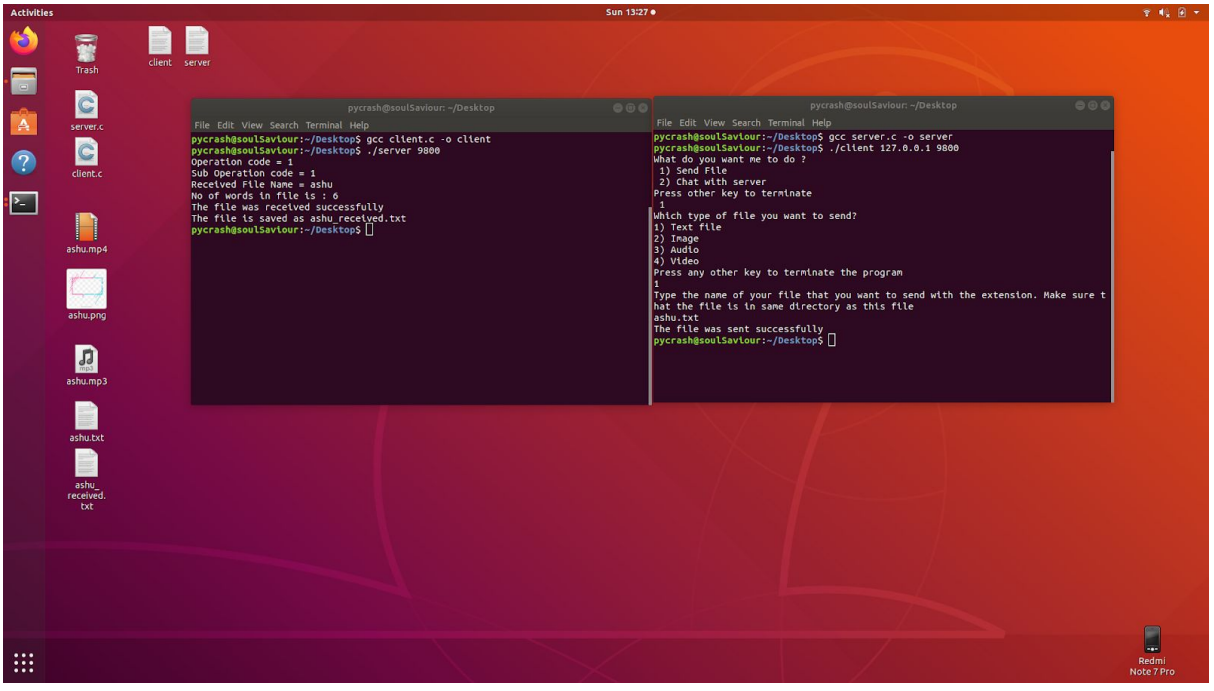
23) Finally we close all the sockets.

Screenshots

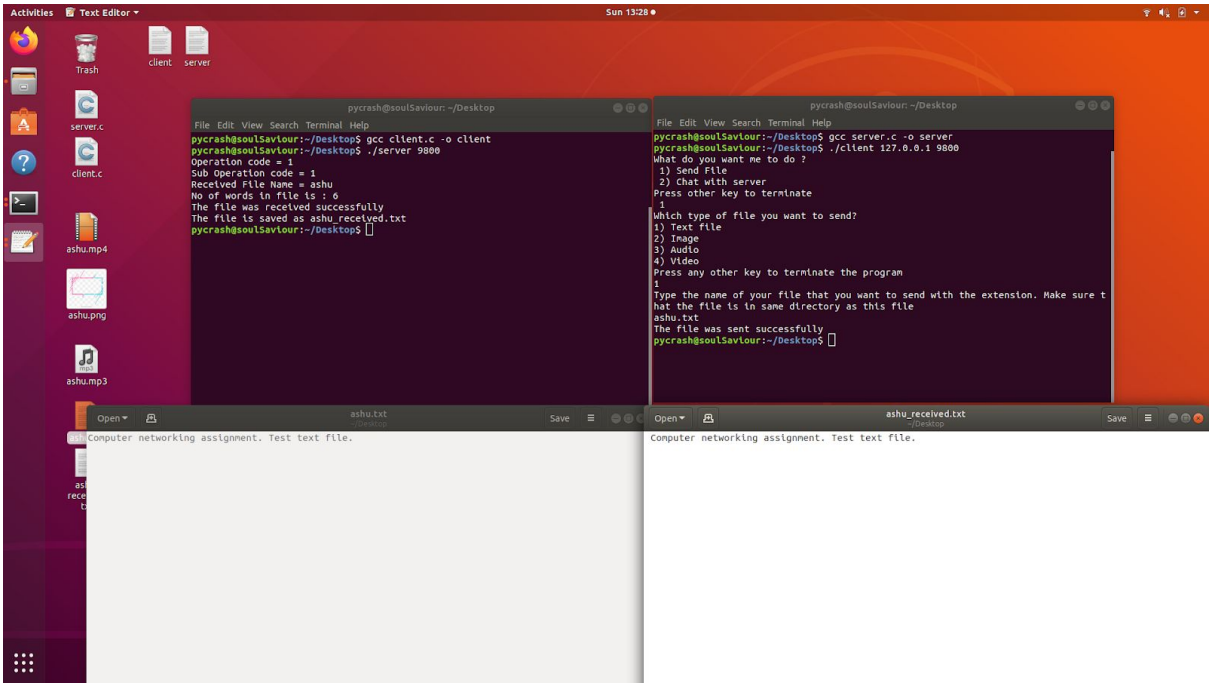
1) Opening the server ant the client



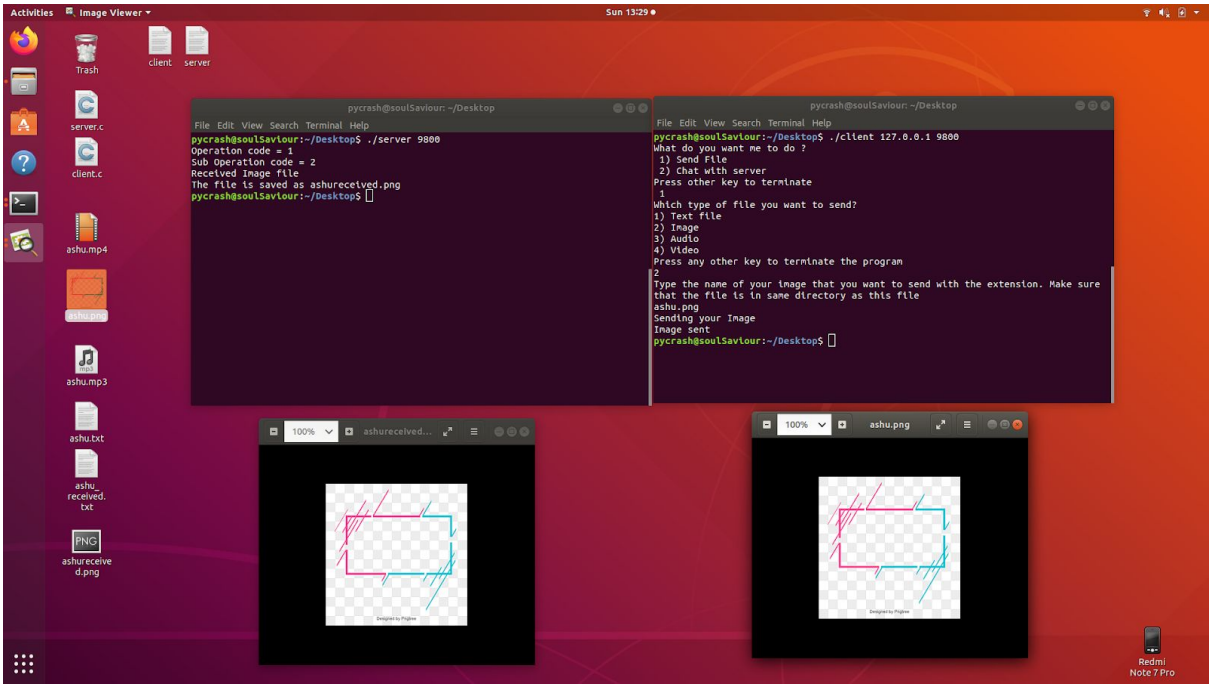
2) Sending the file



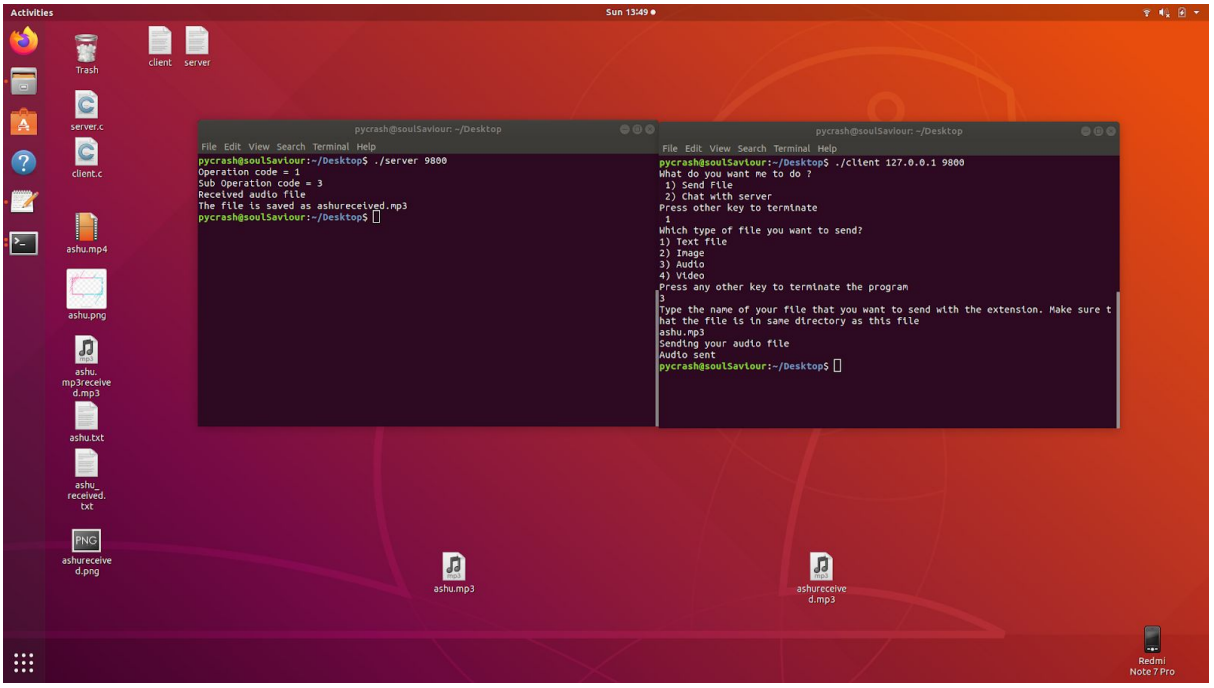
3) Sending a txt file



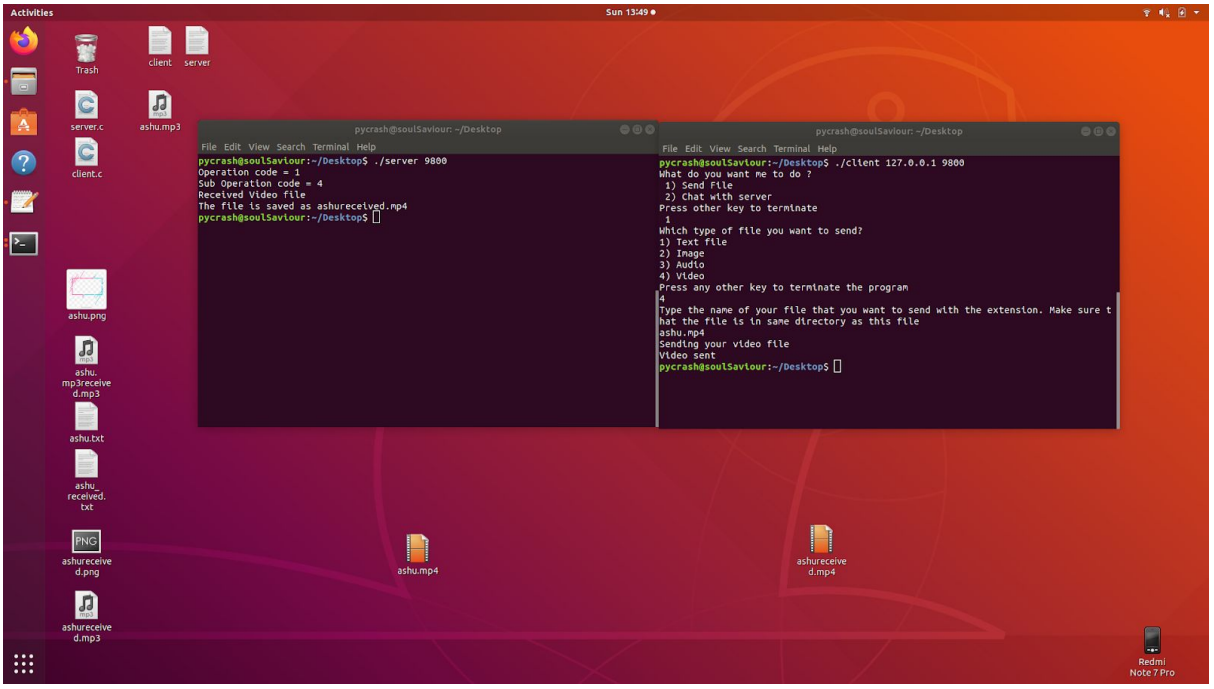
4) Sending an image



5) Sending an audio



6) Sending an video



7) Chat System