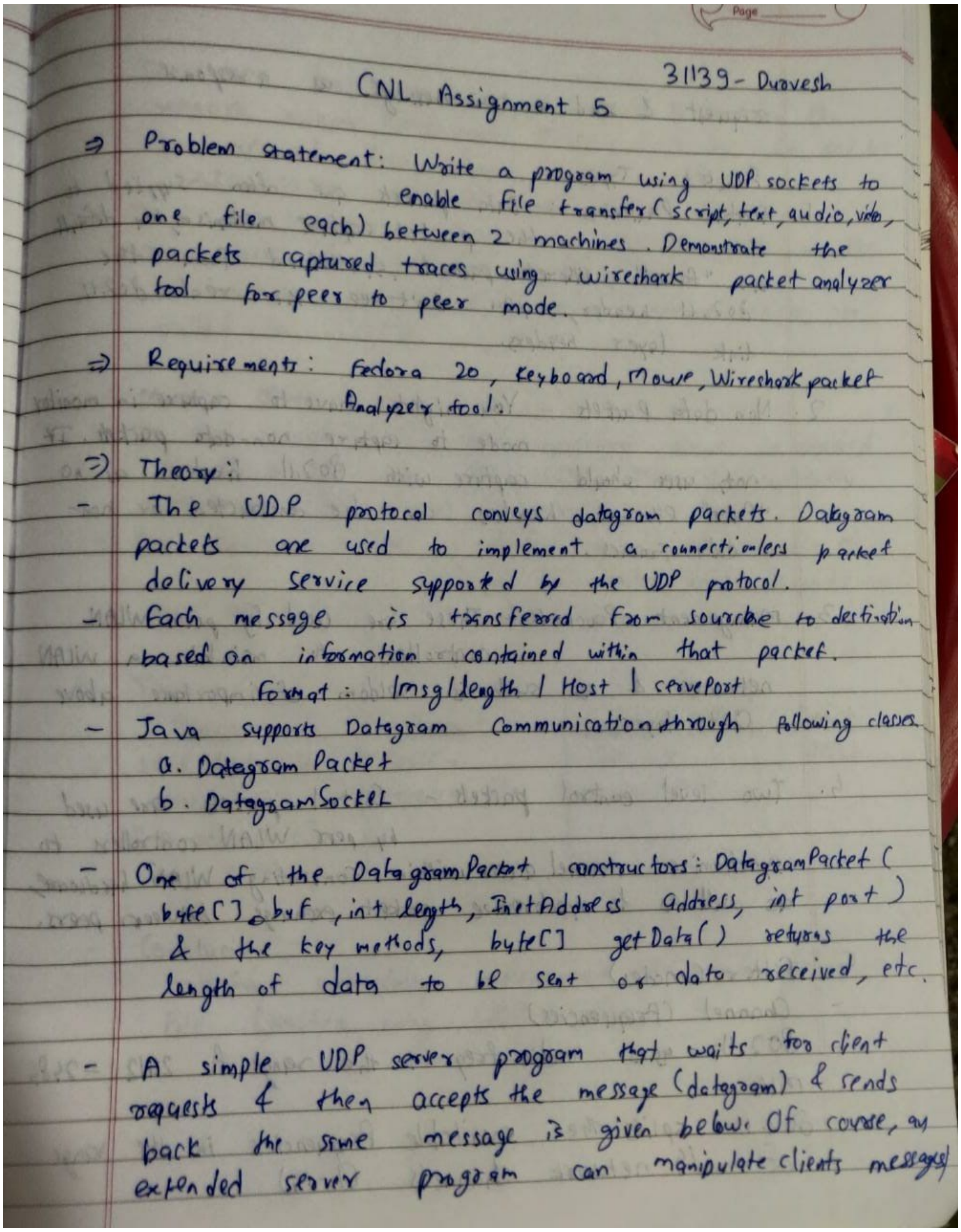
31139-Durvesh-CNLA5

Writeup:



31139 - Durvesh

CNL Assignment 5

⇒ Problem statement: Write a program using UDP sockets to enable file transfer (script, text, audio, video, one file each) between 2 machines. Demonstrate the packets captured traces using wireshark packet analyzer tool for peer to peer mode.

⇒ Requirements: Fedora 20, keyboard, Mouse, Wireshark packet Analyzer tool.

⇒ Theory:

- The UDP protocol conveys datagram packets. Datagram packets are used to implement a connectionless packet delivery service supported by the UDP protocol.
- Each message is transferred from source to destination based on information contained within that packet.
  Format : |msg|length | Host | servePort
- Java supports Datagram Communication through following classes.
  a. Datagram Packet
  b. DatagramSocket

- One of the DatagramPacket constructors: DatagramPacket (byte[] buf , int length, InetAddress address, int port) & the key methods, byte[] getData() returns the length of data to be sent or data received, etc.

- A simple UDP server program that waits for client requests & then accepts the message (datagram) & sends back the same message is given below. Of course, an extended server program can manipulate clients messages request & send a new message as a response.

Packet Types:

1. Data Packets :- Data packets are often supplied to the packet capture mechanism, by default as "fake" Ethernet packets, synthesized from the 802.11 header, you don't see the real 802.11 link layer headers.

2. Non data Packets - You might have to capture in monitor mode to capture non-data packets. If not, user should capture with 802.11 headers as no "fake" ethernet headers can be constructed for non-data frames.

3. management Packets - These are used by peers WLAN controllers to maintain a WLAN network & as such is seldom of importance above OSI layer.

4. Two level control packets - Control packets are used by peer WLAN controllers to synchronize channel access within Consenting WLAN hardware, as well as to synchronize packet exchange between peers.

5. Filter (modes)
- Channel (Frequencies)
- 802.11 uses radio freq in the range of 2412 - 2484 MHz.
- 802.11 splits the available frequencies in the range of 14 network channels (1-14)

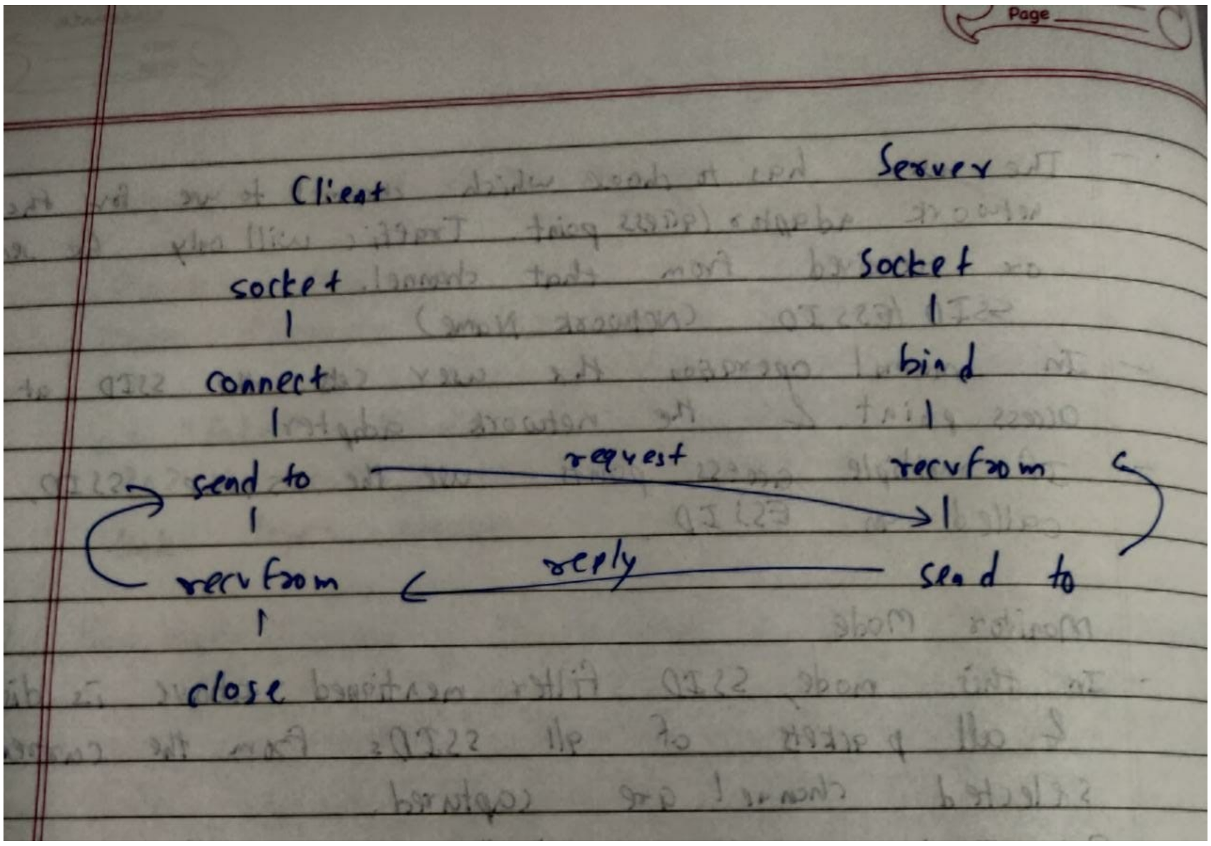
- The user has to choose which channel to use for the network adaptor/access point. Traffic will only be sent to/or received from that channel.
  SSID /ESSID (Network Name)
- In normal operation the user sets the SSID at the access point & the network adapter.
- If multiple access points use the same SSID, its called an ESSID.

Monitor Mode
- In this mode, SSID filter mentioned above is disabled & all packets of all SSIDs from the currently selected channel are captured.
- For Ionium system - Whether you will be able to capture in monitor mode depends on the card & drives you've using.

• Test Cases

	Input	Output	Expected O/P	Result
1.	send text file receive.txt "A text File" to server	Server receives file	Server receives file	PASS

• Conclusion -
  We have successfully implemented the code to enable file transfer using UDP sockets.



Code:

serverudp.c

```
#include<sys/socket.h>
#include<arpa/inet.h>
#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>
#include<sys/types.h>
#include<string.h>
#include<stdlib.h>
#define maxlen 70000
#define mlen 100000
int main()
{
        char fileName[100];
        char filebuffer[2000],caufile[maxlen];
        char *vfilep;
        int aufile[70000],vfile[mlen];
        int sd,connfd,len;
        for(int i=0;i<=100;i++)
        {
                fileName[i]='\0';
        }
        struct sockaddr_in servaddr,cliaddr;
        sd = socket(AF_INET, SOCK_DGRAM, 0);
        if(sd==-1)
{
  printf(" socket not created in server\n");
  exit(0);
}
        else
{
  printf("socket created in  server\n");
}
        bzero(&servaddr, sizeof(servaddr));
        servaddr.sin_family = AF_INET;
        servaddr.sin_addr.s_addr = INADDR_ANY;
        servaddr.sin_port = htons(6666);
        memset(&(servaddr.sin_zero),'\0',8);
        if ( bind(sd, (struct sockaddr *)&servaddr, sizeof(servaddr)) != 0 )
        printf("Not binded\n");
        else
        printf("Binded\n");
        len=sizeof(cliaddr);

        // Receive text file

        recvfrom(sd,fileName,1024,0,(struct sockaddr *)&cliaddr, &len);
```

```
          printf("NAME OF TEXT FILE RECEIVED : %s\n",fileName);
          FILE *fp;
          printf("Contents in the received text file : \n");
          recvfrom(sd,filebuffer,1024,0,(struct sockaddr *)&cliaddr, &len);
          printf("%s\n",filebuffer);
          int fsize=strlen(filebuffer);
          fp=fopen(fileName,"w");
          if(fp)
          {
                    fwrite(filebuffer, fsize, 1, fp);
                    printf("File received successfully.\n");
          }
          else
                    printf("Cannot create to output file.\n");
          memset(fileName, '\0', sizeof(fileName));
          fclose(fp);

          // Receiving audio file


          recvfrom(sd,fileName,1024,0,(struct sockaddr *)&cliaddr, &len);
          printf("NAME OF AUDIO FILE RECEIVED : %s\n",fileName);
          FILE *afp;
          int numbytes;
afp=fopen(fileName,"w");
size_t  afsize;
afsize=recvfrom(sd,aufile,70000,0,(struct sockaddr *)&cliaddr, &len);
if(afp)
          {
          fwrite(aufile, afsize, 1, afp);
          printf("File received successfully.\n");
}
```

```
   else
                        printf("Cannot open output file.\n");
            memset(fileName, '\0', sizeof(fileName));
            fclose(afp);

            // Receiving video file

   recvfrom(sd,fileName,1024,0,(struct sockaddr *)&cliaddr, &len);
   printf("VIDEO FILE NAME RECEIVED : %s\n",fileName);
   FILE *vfp;
   vfp=fopen(fileName,"w");
   size_t  vfsize;
   vfsize=recvfrom(sd,vfile,100000,0,(struct sockaddr *)&cliaddr, &len);
   if(vfp)
            {
       fwrite(vfile, vfsize, 1, vfp);
       printf("File received successfully.\n");
   }
   else
       printf("Cannot open output file.\n");
   fclose(vfp);
            close(sd);
            return(0);
}
```

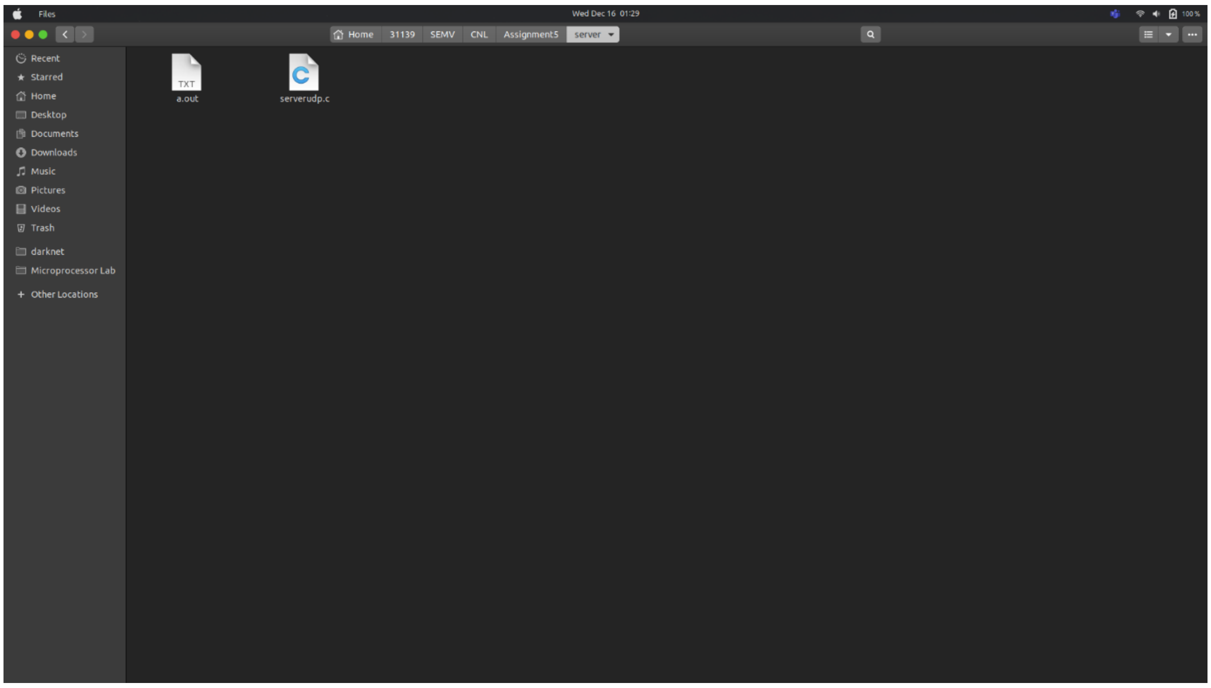
clientudp.c:

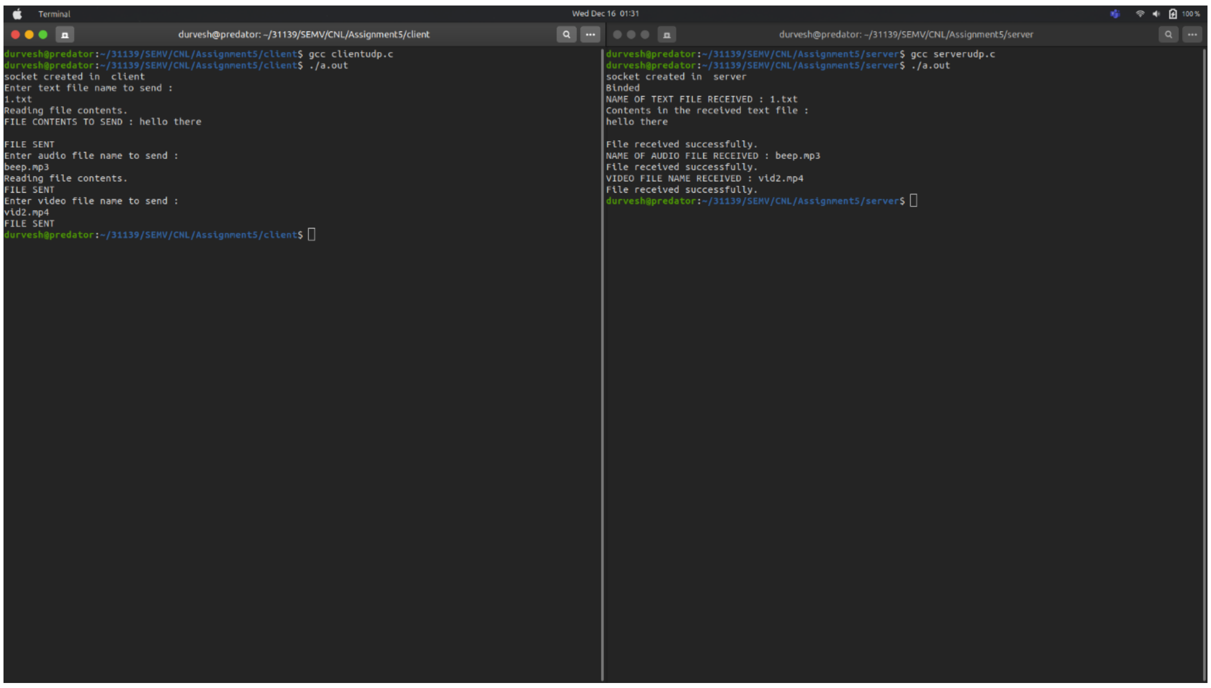
```
#include <stdio.h>
#include <errno.h>
#include <sys/socket.h>
```

```
#include <resolv.h>
#include<netinet/in.h>
#include<sys/types.h>
#include <stdlib.h>
#include<string.h>
#include <unistd.h>
#define maxlen 300000
int main()
{
        char fileName[2000],afileName[2000],vfileName[2000],file_buffer[2000],c,caufile[70000];
  int sockfd,connfd,len,aufile[70000],vfile[100000];
        struct sockaddr_in servaddr,cliaddr;
        sockfd = socket(AF_INET, SOCK_DGRAM, 0);
        if(sockfd==-1)
        {
        printf(" socket not created in client\n");
        exit(0);
        }
        else
        printf("socket created in  client\n");

        bzero(&servaddr, sizeof(servaddr));

        servaddr.sin_family = AF_INET;
        servaddr.sin_addr.s_addr = INADDR_ANY;
        servaddr.sin_port = htons(6666);
        memset(&(servaddr.sin_zero),'\0',8);

        // Transfering text file

        printf("Enter text file name to send : \n");
```

```
   scanf("%s",fileName);
   sendto(sockfd, fileName, strlen(fileName), 0, (struct sockaddr *)&servaddr, sizeof(struct
sockaddr));
   FILE *fp;
   fp=fopen(fileName,"r");
   if(fp)
           {
           printf("Reading file contents.\n");
           fseek(fp,0,SEEK_END);
           size_t file_size=ftell(fp);
           fseek(fp,0,SEEK_SET);
           if(fread(file_buffer,file_size,1,fp)<=0)
      {
        printf("Unable to copy file into buffer or empty file.\n");
        exit(1);
      }
   }
           else
           {
           printf("Cannot open file.\n");
           exit(0);
   }
   printf("FILE CONTENTS TO SEND : %s\n",file_buffer);
   if(sendto(sockfd, file_buffer,strlen(file_buffer), 0,(struct sockaddr *)&servaddr, sizeof(struct
sockaddr))<0)
           printf("FILE WAS NOT SENT\n");
   else
           printf("FILE SENT\n");
   fclose(fp);

           // Transfering audio file
   printf("Enter audio file name to send : \n");
   scanf("%s",afileName);
   sendto(sockfd, afileName, strlen(afileName), 0,(struct sockaddr *)&servaddr, sizeof(struct
sockaddr));
           FILE *afp;
           afp=fopen(afileName,"r");
           fseek(afp,0,SEEK_END);
           size_t  afsize=ftell(afp);
           fseek(afp,0,SEEK_SET);
           if(afp)
           {
                       printf("Reading file contents.\n");
                       if(fread(aufile,afsize,1,afp)<=0)
                       {
                       printf("Unable to copy file into buffer or empty file.\n");
                 exit(1);
              }
           }
           else
           {
                       printf("Could not read audio file.\n");
                       exit(0);
           }

           if(sendto(sockfd, aufile, afsize, 0,(struct sockaddr *)&servaddr, sizeof(struct sockaddr))<0)
                       printf("FILE WAS NOT SENT\n");
           else
              printf("FILE SENT\n");
           fclose(afp);

           // Transfering video file
           printf("Enter video file name to send : \n");
   scanf("%s",vfileName);
   sendto(sockfd, vfileName, strlen(vfileName), 0,(struct sockaddr *)&servaddr, sizeof(struct
sockaddr));
           FILE *vfp;
           vfp=fopen(vfileName,"r");
           fseek(vfp, 0, SEEK_END);
           size_t vfsize = ftell(vfp);
           fseek(vfp, 0, SEEK_SET);

           if(vfp)
           {
                       if(fread(vfile, 1, vfsize, vfp)<=0)
                                   printf("No contents or error reading file \n");
           }
           else
           {
                       printf("Could not read audio file.\n");
                       exit(0);
           }
           if(sendto(sockfd, vfile, vfsize, 0,(struct sockaddr *)&servaddr, sizeof(struct sockaddr))<0)
                       printf("FILE WAS NOT SENT\n");
           else
                       printf("FILE SENT\n");
           fclose(vfp);
   close(sockfd);
           return(0);
}
```

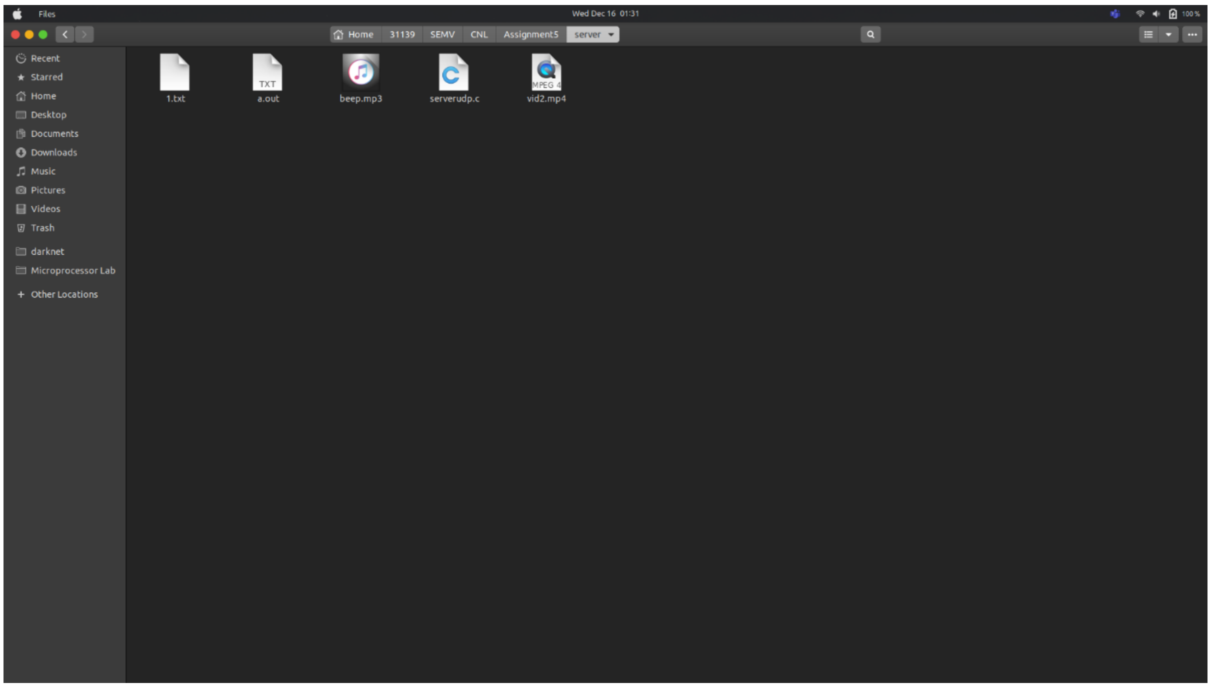
Outputs:




Terminal
Wed Dec 16 01:31
100 %
durvesh@predator: ~/31139/SEMV/CNL/Assignment5/client
durvesh@predator: ~/31139/SEMV/CNL/Assignment5/server

```
durvesh@predator:~/31139/SEMV/CNL/Assignment5/client$ gcc clientudp.c
durvesh@predator:~/31139/SEMV/CNL/Assignment5/client$ ./a.out
socket created in  client
Enter text file name to send :
1.txt
Reading file contents.
FILE CONTENTS TO SEND : hello there

FILE SENT
Enter audio file name to send :
beep.mp3
Reading file contents.
FILE SENT
Enter video file name to send :
vid2.mp4
FILE SENT
durvesh@predator:~/31139/SEMV/CNL/Assignment5/client$
```

```
durvesh@predator:~/31139/SEMV/CNL/Assignment5/server$ gcc serverudp.c
durvesh@predator:~/31139/SEMV/CNL/Assignment5/server$ ./a.out
socket created in  server
Binded
NAME OF TEXT FILE RECEIVED : 1.txt
Contents in the received text file :
hello there

File received successfully.
NAME OF AUDIO FILE RECEIVED : beep.mp3
File received successfully.
VIDEO FILE NAME RECEIVED : vid2.mp4
File received successfully.
durvesh@predator:~/31139/SEMV/CNL/Assignment5/server$
```


Files
Wed Dec 16 01:31
100 %
Home
31139
SEMV
CNL
Assignment5
server
Recent
Starred
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
darknet
Microprocessor Lab
Other Locations
1.txt
a.out
beep.mp3
serverudp.c
vid2.mp4