**Computer Networks – Lab 6**

|  |  |
| --- | --- |
| OSAMA MOHAMMED AFZAL | 237529 |
| MUHAMMAD HASNAIN NAEEM | 212728 |

**Running and Compilation:**

To run it, open up two shell windows (either both on the same machine, or on different machines). In one, run it like this:

**./a.out receive filename\_to\_create.txt 9999**

Then, in the other, run it like this:

**./a.out send filename\_to\_send.txt 127.0.0.1:9999**

(if the receiver is on a different machine than the sender, replace **127.0.0.1** with the IP address of the receiver machine).

If all is successful, you will see output in both windows indicating data is being transferred, and at the end, both programs will exit, and there will be a file **filename\_to\_create.txt** on the receiving machine that is identical to **filename\_to\_send.txt** is on the sending machine.

**How to read and Audio /video File using FFmpeg in C**

**Audio:**

<https://batchloaf.wordpress.com/2017/02/10/a-simple-way-to-read-and-write-audio-and-video-files-in-c-using-ffmpeg/>

**Video:**

<https://batchloaf.wordpress.com/2017/02/12/a-simple-way-to-read-and-write-audio-and-video-files-in-c-using-ffmpeg-part-2-video/>

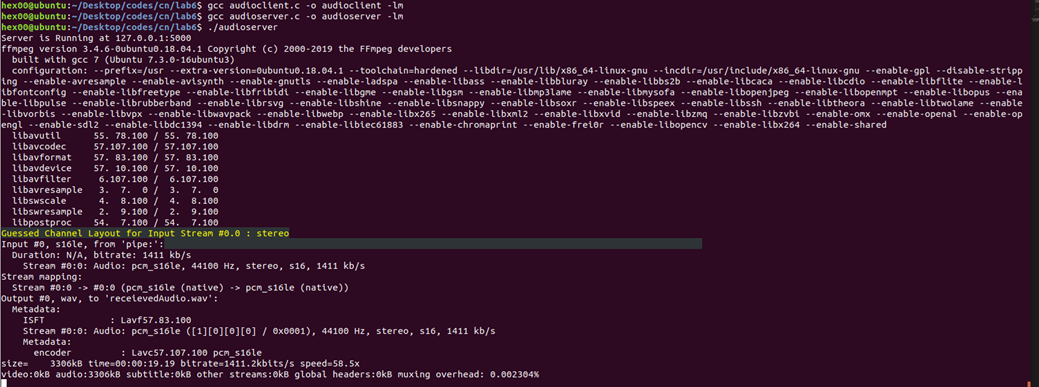
**Conclusion & Takeaways:**

In this lab, we learned and demonstrated how socket programming can be used to send files through TCP/UDP protocols. Moreover, we can utilize the FFMPEG library to perform operations on each frame of video file being sent. Plus, we can perform operations on the audio such as changing number of channels, sampling rate etc.   
To summarize, we looked into performing real-time operations on files whilst sending them from server to client.

**Task 1**

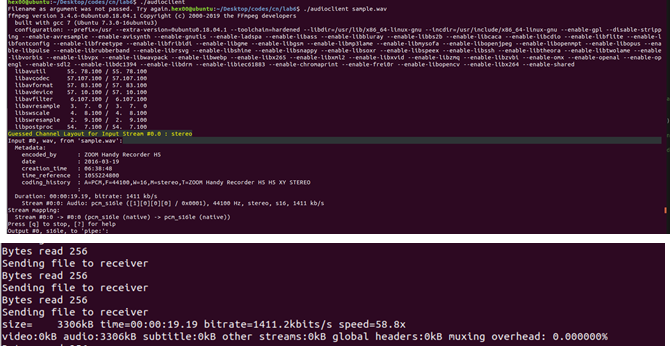
**Audioserver.c**

|  |
| --- |
| #include <unistd.h>  #include <errno.h>  #include <arpa/inet.h>  #include <sys/socket.h>  #include <sys/types.h>  #include <netinet/in.h>  #include <netdb.h>  #include <stdio.h>  #include <string.h>  #include <stdlib.h>    int main(void)  {  // variables decleration  int SenderSock = 0; //socket descriptor  int connectionSock = 0;  int bytesRead = 0;  struct sockaddr\_in serv\_addr;  int16\_t bufferread[256]; //for reading file content    // socket creation  SenderSock = socket(AF\_INET, SOCK\_STREAM, 0);  printf("Server is Running at 127.0.0.1:5000\n");  memset(&serv\_addr, '0', sizeof(serv\_addr)); //filling block of memory with 0  memset(bufferread, '0', sizeof(bufferread)); //filling 256 memory locations with 0  serv\_addr.sin\_port = htons(5000);  serv\_addr.sin\_family = AF\_INET;  serv\_addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);    // binding to local host  bind(SenderSock, (struct sockaddr\*)&serv\_addr,sizeof(serv\_addr));    // waiting for the sender  if(listen(SenderSock, 10) == -1)  {  printf("No response from sender\n");  }    // sending the file  while(1)  {  connectionSock = accept(SenderSock, (struct sockaddr\*)NULL ,NULL); //connection established between sender and receiver  FILE \*audioFile; // file pointer to our file  // reading 2 audio channels.  audioFile = popen("ffmpeg -y -f s16le -ar 44100 -ac 2 -i - receievedAudio.wav", "w");  // reading file from buffer and writing to a new file  while((bytesRead = read(connectionSock, bufferread, 256)) > 0) //reading segments and checking if there are any segments  {  fwrite(bufferread, 1, bytesRead, audioFile); //writing segments to a file recievedAudio.wav  if (bytesRead < 256) // because only last segment would be less than 256  {  printf("End of file\n");  break;  }  }    fclose(audioFile); //closing file  close(connectionSock); //closing socket  }  return 0;  } |

****

**Audioclient.c**

|  |
| --- |
| #include <unistd.h>  #include <errno.h>  #include <arpa/inet.h>  #include <sys/socket.h>  #include <sys/types.h>  #include <netinet/in.h>  #include <netdb.h>  #include <stdio.h>  #include <string.h>  #include <stdlib.h>    int main(int argc, char\*\* argv)  {  if(argc < 2){  printf("Filename as argument was not passed. Try again.");  return -1;  }  //variable decleration  int receiverSock = 0;  struct sockaddr\_in serv\_addr;  int16\_t bufferread[256]={0}; //for reading file content    // socket creation  receiverSock = socket(AF\_INET, SOCK\_STREAM, 0);  serv\_addr.sin\_port = htons(5000);  serv\_addr.sin\_family = AF\_INET;  serv\_addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");    // initiating connection on socket  if(connect(receiverSock, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr))<0)  {  printf("Failed to connect to server \n"); //could not connect  return 1;  }    // Open WAV file with FFmpeg and read raw samples via the pipe.  FILE \*audioFile;  char init\_command[] = "ffmpeg -i ";  char end\_command[] = " -f s16le -ar 44100 -ac 2 -";  char command[100];  strcpy(command, init\_command);  strcat(command, argv[1]);  strcat(command, end\_command);  audioFile = popen(command, "r");    if(audioFile==NULL)  {  printf("failed to open file \n"); //video file does not exists  return 1;  }    while(1)  {  // reading from a video file  int nread = fread(bufferread,1, 256, audioFile); //segments upto length of 256  printf("Bytes read %d \n", nread); //reading file in segments    // creating segments and sending them  write(receiverSock, bufferread, nread); //writing bytes to buffer  if(nread > 0)  {  printf("Sending file to receiver \n");    }    if (nread < 256) // because only last segment would be less than 256  {  if (feof(audioFile))//end of file  printf("End of file\n");  break;  }  }    pclose(audioFile);  close(receiverSock); //closing socket  return 0;  } |

****

Design a GNU C program in Linux environment, in which server transfer Audio file to Client .

**Task2:**

Design a GNU C program in Linux environment, in which server transfer video file to Client.

|  |
| --- |
| Task 2 |
| #include <unistd.h>  #include <stdio.h>  #include <sys/socket.h>  #include <arpa/inet.h>  #include <stdlib.h>  #include<fcntl.h>  #include <netinet/in.h>  #include <string.h>  #define SEND "send"  #define RECEIVE "receive"  int isSend;  int validate(int argc, char \*argv[]){  char\* method, filename, IP\_ADDR;  int PORT;  int isValid = 0;  if (argc < 4){  printf("\nInsufficient Arguments");  exit(0);  }  else{  method = argv[1];  if(strcmp(method,SEND) == 0 ){  isSend = 1;  isValid = 1;  }  else if(strcmp(method,RECEIVE) == 0){  isSend = 0;  isValid = 1;  }  else{  isSend = -1;  printf("Method specified does not exist");  exit(-1);  }  isValid = 1;  }  return isValid;  }  void sendC(char \*args[]){  int sock, bytes\_rec;  struct sockaddr\_in server;  char buffer[5000] = {0};  double buff[1] = {0};  int addlen = sizeof(server), opt = 1;  FILE \*fp;  fp = fopen(args[2],"rb");  if (fp == NULL) {  perror("fopen()");  exit(EXIT\_FAILURE);  }  //Creating a socket  if ((sock = socket(AF\_INET, SOCK\_STREAM, 0)) == 0)  {  perror("socket failed");  exit(EXIT\_FAILURE);  }  if (setsockopt(sock, SOL\_SOCKET, SO\_REUSEADDR | SO\_REUSEPORT,  &opt, sizeof(opt)))  {  perror("setsockopt");  exit(EXIT\_FAILURE);  }  server.sin\_family = AF\_INET;  server.sin\_addr.s\_addr = INADDR\_ANY;  server.sin\_port = htons(atoi(args[4]));  //binding Socket to the port  if(bind(sock,(struct sockaddr\*)&server,sizeof(server))<0){  perror("Failed to bind");  exit(EXIT\_FAILURE);  }  if(listen(sock,10)<0){  perror("listen");  exit(EXIT\_FAILURE);  }  if((sock = accept(sock,(struct sockaddr\*)&server,(socklen\_t\*)&addlen)) <0){  perror("accept");  exit(EXIT\_FAILURE);  }  while(1){  double bytes\_read = fread(buffer,1,sizeof(buffer),fp);  if (bytes\_read == 0)  break;  void \*p = buffer;    while(bytes\_read > 0){  int bytes\_written = send(sock,p,bytes\_read,0);  bytes\_read -= bytes\_written;  p += bytes\_written;  }  }  }  int receive(char \*args[]){  FILE \*fp;  fp = fopen(args[2],"wb");  if (fp == NULL) {  perror("fopen()");  exit(EXIT\_FAILURE);  }  int sock = 0, n;  struct sockaddr\_in serv\_addr;  char buffer[5000] = {0};  char\* pbuffer = buffer;  double size\_rec = 0.0;    if ((sock = socket(AF\_INET, SOCK\_STREAM, 0)) < 0)  {  printf("\n Socket creation error \n");  return -1;  }  serv\_addr.sin\_family = AF\_INET;  serv\_addr.sin\_port = htons(atoi(args[3]));    // Convert IPv4 and IPv6 addresses from text to binary form  if(inet\_pton(AF\_INET, "10.7.24.193", &serv\_addr.sin\_addr)<=0)  {  printf("\nInvalid address/ Address not supported \n");  return -1;  }  if (connect(sock, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr)) < 0)  {  printf("\nConnection Failed \n");  return -1;  }  // will remain open until the server terminates the connection  while ((n = recv(sock, pbuffer, 5000, 0)) > 0) {  fwrite(buffer,n,1,fp);  bzero(buffer,sizeof(buffer));  }  fclose(fp);  close(sock);  }  int main(int argc, char \*argv[]){  if(validate(argc,argv)){    if(isSend){  sendC(argv);  }  else{  receive(argv);  }  }  else{  exit(0);  }    } |

README BEFORE RUNNING:

* File takes 4 arguments
  + Method ( send or receive) : STRING
  + Filename + extension : STRING
  + IP address : INT (only if you are server else dont give)
  + PORT : INT
* The above arguments are space seperated.