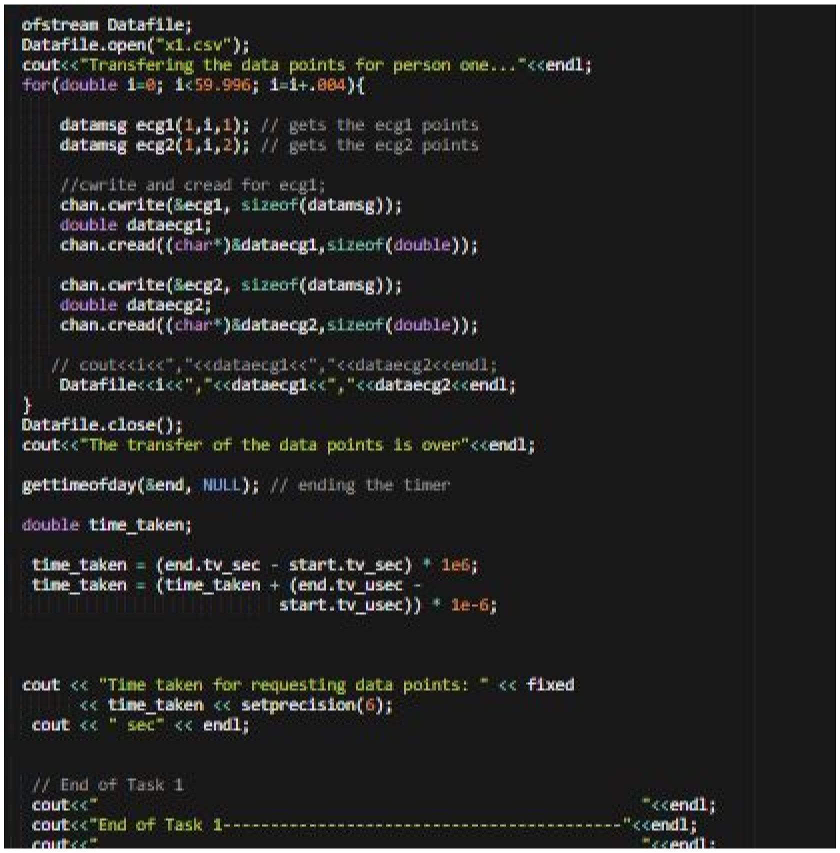
Sumedh Tadimeti
2/21/202
CSCE 313
Programming Assignment 2 - A Client Speaking to a Server Process Report

For this report I will break down each task at a time and will describe my thinking and my design.

Task 1: Requesting Data Points:

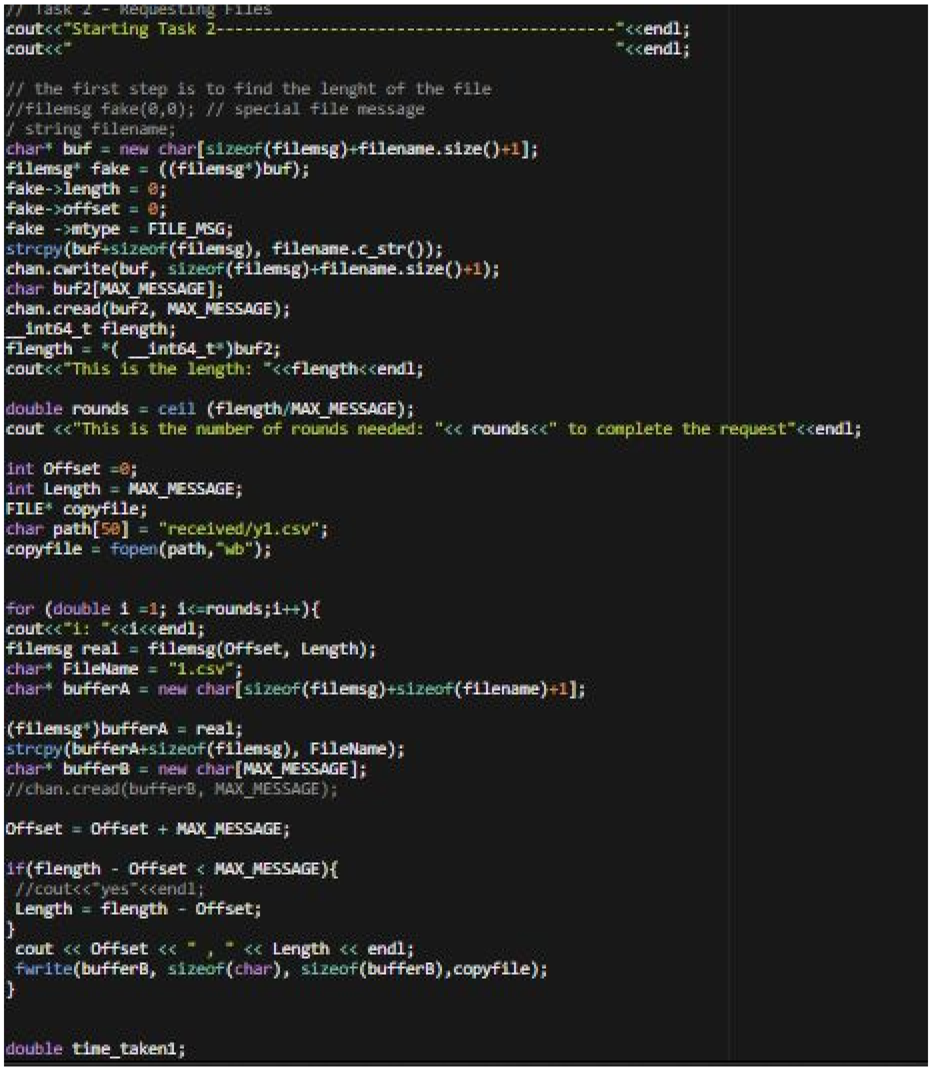
The goal for task one was to transfer all the points from one file into another file. This is done using a for loop and two datamsg messages (one for ecg 1 and another for exg2). The for loop incremented from 0 to 59.996 to parse through all the time and each datamsg collected the value at the time for its respective ecg value. Then I used ofstream to insert this data into a new file using the same for mating as the original files. I used the get time of day function to tally up all the time taken for the for loop and the transfer of the data points. The code is below:



```
ofstream Datafile;
Datafile.open("x1.csv");
cout<<"Transfering the data points for person one..."<<endl;
for(double i=0; i<59.996; i=i+.004){

    datamsg ecg1(1,i,1); // gets the ecg1 points
    datamsg ecg2(1,i,2); // gets the ecg2 points

    //cwrite and cread for ecg1;
    chan.cwrite(&ecg1, sizeof(datamsg));
    double dataecg1;
    chan.cread((char*)&dataecg1,sizeof(double));

    chan.cwrite(&ecg2, sizeof(datamsg));
    double dataecg2;
    chan.cread((char*)&dataecg2,sizeof(double));

   // cout<<i<<","<<dataecg1<<","<<dataecg2<<endl;
    Datafile<<i<<","<<dataecg1<<","<<dataecg2<<endl;
}
Datafile.close();
cout<<"The transfer of the data points is over"<<endl;

gettimeofday(&end, NULL); // ending the timer

double time_taken;

 time_taken = (end.tv_sec - start.tv_sec) * 1e6;
 time_taken = (time_taken + (end.tv_usec -
                          start.tv_usec)) * 1e-6;



cout << "Time taken for requesting data points: " << fixed
     << time_taken << setprecision(6);
 cout << " sec" << endl;


 // End of Task 1
 cout<<"                                                      "<<endl;
 cout<<"End of Task 1-----------------------------------------"<<endl;
 cout<<"                                                      "<<endl;
```

Task 2: Requesting Files:

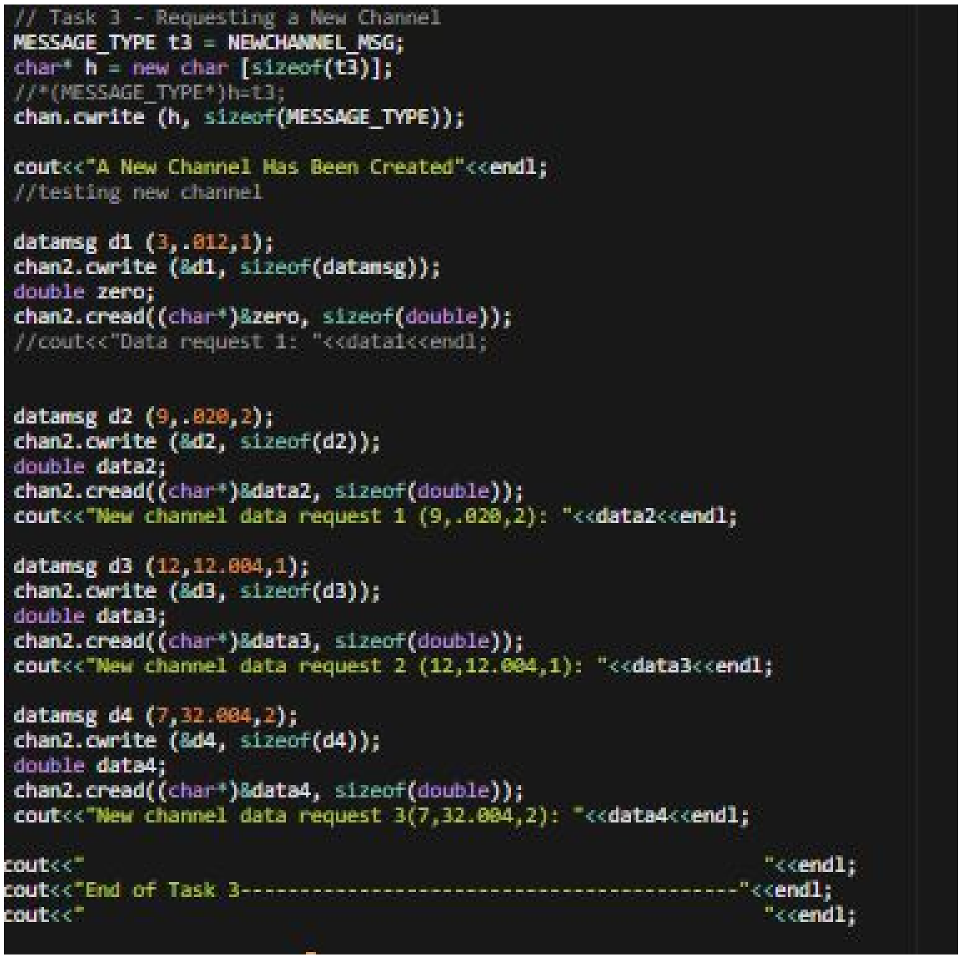
The first thing that I did was get the length of a given file. I did this by sending a special filemsg where the offset and the length were zero. Then I created two buffers, one buffer was the combined size of filemsg and the name of the file plus one, the other had the size of MAX_MESSAGE. Then once I received the length of the file, I divided it by the MAX_MESSAGE size and ceiled to obtain the number of packets I need to transfer the file in its completion. Then It was time to actually transfer the file. This algorithm was similar to getting the length of the file, but you had to increase the offset by the MAX_MESSAGE every iteration of the for loop. Finally on the last packet there might be less that MAX_MESSAGE left. If this is the case, I set the length equal to the length of the file minus the current offset value. I used fopen and fwrite to transfer the file. The code is below:



```
// Task 2 - Requesting Files
cout<<"Starting Task 2-----------------------------------------"<<endl;
cout<<"                                                         "<<endl;

// the first step is to find the lenght of the file
//filemsg fake(0,0); // special file message
/ string filename;
char* buf = new char[sizeof(filemsg)+filename.size()+1];
filemsg* fake = ((filemsg*)buf);
fake->length = 0;
fake->offset = 0;
fake ->mtype = FILE_MSG;
strcpy(buf+sizeof(filemsg), filename.c_str());
chan.cwrite(buf, sizeof(filemsg)+filename.size()+1);
char buf2[MAX_MESSAGE];
chan.cread(buf2, MAX_MESSAGE);
__int64_t flength;
flength = *( __int64_t*)buf2;
cout<<"This is the length: "<<flength<<endl;

double rounds = ceil (flength/MAX_MESSAGE);
cout <<"This is the number of rounds needed: "<< rounds<<" to complete the request"<<endl;

int Offset =0;
int Length = MAX_MESSAGE;
FILE* copyfile;
char path[50] = "received/y1.csv";
copyfile = fopen(path,"wb");


for (double i =1; i<=rounds;i++){
cout<<"i: "<<i<<endl;
filemsg real = filemsg(Offset, Length);
char* FileName = "1.csv";
char* bufferA = new char[sizeof(filemsg)+sizeof(filename)+1];

(filemsg*)bufferA = real;
strcpy(bufferA+sizeof(filemsg), FileName);
char* bufferB = new char[MAX_MESSAGE];
//chan.cread(bufferB, MAX_MESSAGE);

Offset = Offset + MAX_MESSAGE;

if(flength - Offset < MAX_MESSAGE){
 //cout<<"yes"<<endl;
 Length = flength - Offset;
}
 cout << Offset << " , " << Length << endl;
 fwrite(bufferB, sizeof(char), sizeof(bufferB),copyfile);
}


double time_taken1;
```

Task 3:

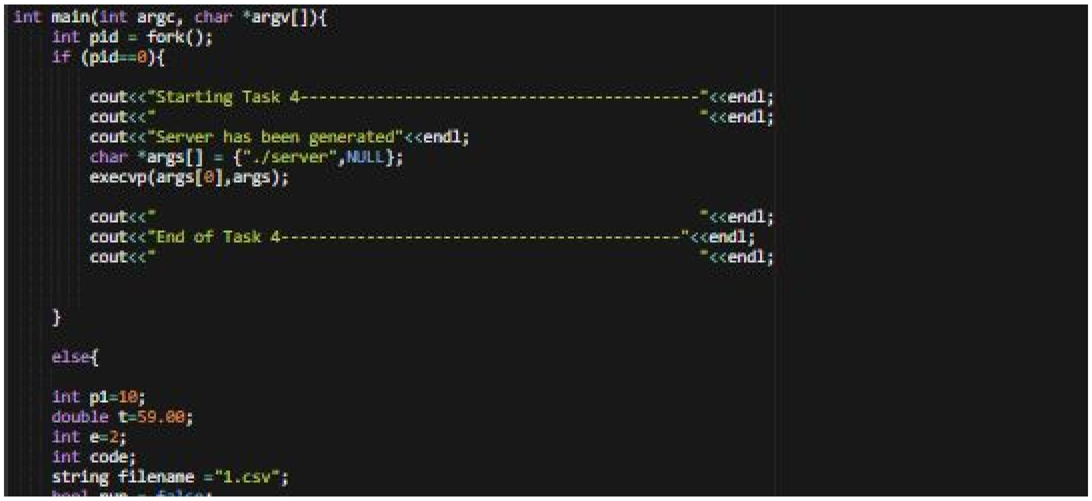
For task three, I requested a new channel. I did this my creating a MESSAGE_TYPE and setting it equal to NEWCHANNEL_MSG. This asked the server to create a new channel for me. I used the original channel to send the NEWCHANNEL_MSG, but after I did that used the new channel for all the example datamsg requests. The code is below:



```
 // Task 3 - Requesting a New Channel
 MESSAGE_TYPE t3 = NEWCHANNEL_MSG;
 char* h = new char [sizeof(t3)];
 //*(MESSAGE_TYPE*)h=t3;
 chan.cwrite (h, sizeof(MESSAGE_TYPE));

 cout<<"A New Channel Has Been Created"<<endl;
 //testing new channel

 datamsg d1 (3,.012,1);
 chan2.cwrite (&d1, sizeof(datamsg));
 double zero;
 chan2.cread((char*)&zero, sizeof(double));
 //cout<<"Data request 1: "<<data1<<endl;


 datamsg d2 (9,.020,2);
 chan2.cwrite (&d2, sizeof(d2));
 double data2;
 chan2.cread((char*)&data2, sizeof(double));
 cout<<"New channel data request 1 (9,.020,2): "<<data2<<endl;

 datamsg d3 (12,12.004,1);
 chan2.cwrite (&d3, sizeof(d3));
 double data3;
 chan2.cread((char*)&data3, sizeof(double));
 cout<<"New channel data request 2 (12,12.004,1): "<<data3<<endl;

 datamsg d4 (7,32.004,2);
 chan2.cwrite (&d4, sizeof(d4));
 double data4;
 chan2.cread((char*)&data4, sizeof(double));
 cout<<"New channel data request 3(7,32.004,2): "<<data4<<endl;

cout<<"                                                       "<<endl;
cout<<"End of Task 3----------------------------------------"<<endl;
cout<<"                                                       "<<endl;
```

Task 4:

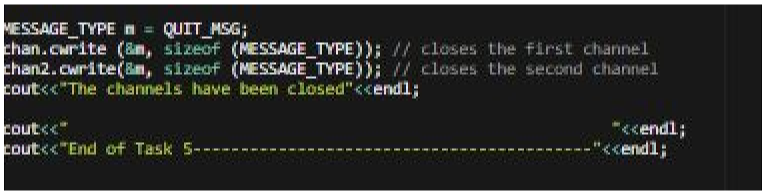
For task 4, I ran the server as a child process of the client. I did this using the fork, wait and execvp functions. I used fork to get the process ID, and if that ID was zero I ran the server, if it was not 0, I ran the rest of my code. A screenshot of my code is below:



```
int main(int argc, char *argv[]){
    int pid = fork();
    if (pid==0){

        cout<<"Starting Task 4------------------------------------------"<<endl;
        cout<<"                                                         "<<endl;
        cout<<"Server has been generated"<<endl;
        char *args[] = {"./server",NULL};
        execvp(args[0],args);

        cout<<"                                                         "<<endl;
        cout<<"End of Task 4------------------------------------------"<<endl;
        cout<<"                                                         "<<endl;


    }

    else{

    int p1=10;
    double t=59.00;
    int e=2;
    int code;
    string filename ="1.csv";
```

Task 5:

The fifth task was to close the channels. I did this by setting MESSAGE_TYPE equal to QUIT_MSG and sent this message through all the channels that I created in my code (Which is two). A screenshot of my code is below:



```
MESSAGE_TYPE m = QUIT_MSG;
chan.cwrite (&m, sizeof (MESSAGE_TYPE)); // closes the first channel
chan2.cwrite(&m, sizeof (MESSAGE_TYPE)); // closes the second channel
cout<<"The channels have been closed"<<endl;

cout<<"                                                         "<<endl;
cout<<"End of Task 5------------------------------------------"<<endl;
```

Analysis:

When comparing the times. It was significantly faster to request the file as opposed to transferring all the data points. This is because it is faster to send filmsges as opposed to sending datamsges.