Scott Allen
Monday, June 1, 2020

A Client Process Speaking to a Server Process

My tasks:

Requesting Data Points (15 pts)

```
//GETTING DATA POINT (15 PTS)
if(pdef || tdef || edef){
    FIFORequestChannel chan ("control", FIFORequestChannel::CLIENT_SIDE);
        cout << "Input Recieved"<< endl;
        cout << "Patient Number : " << patient_num;
        cout <<" Time : "<< ecg_time;
        cout <<" ECG : " << ecg<<endl;

        datamsg *data = new datamsg( patient_num, ecg_time, ecg);
        chan.cwrite(data, sizeof(datamsg));
        double recived_val = 0;
        chan.cread( &recived_val, sizeof(double));
        cout << recived_val << endl;
```

Requesting a new data point was simply just sending data message. This was done with the parameters of the patient number, the time in seconds, and the ECG (1 or 2). After this was done we would write to the server with the data message. Once this was done we created a spot for the value to be stored. Once this was done we would read the value the server sent back. For testing purposes I created a nested for loop to iterate through grabbing a number of data points. This was done with the command ./client -p 0. This was the signal for the code to run through my test case and print out the time taken. The graph of collecting data points is displayed below.

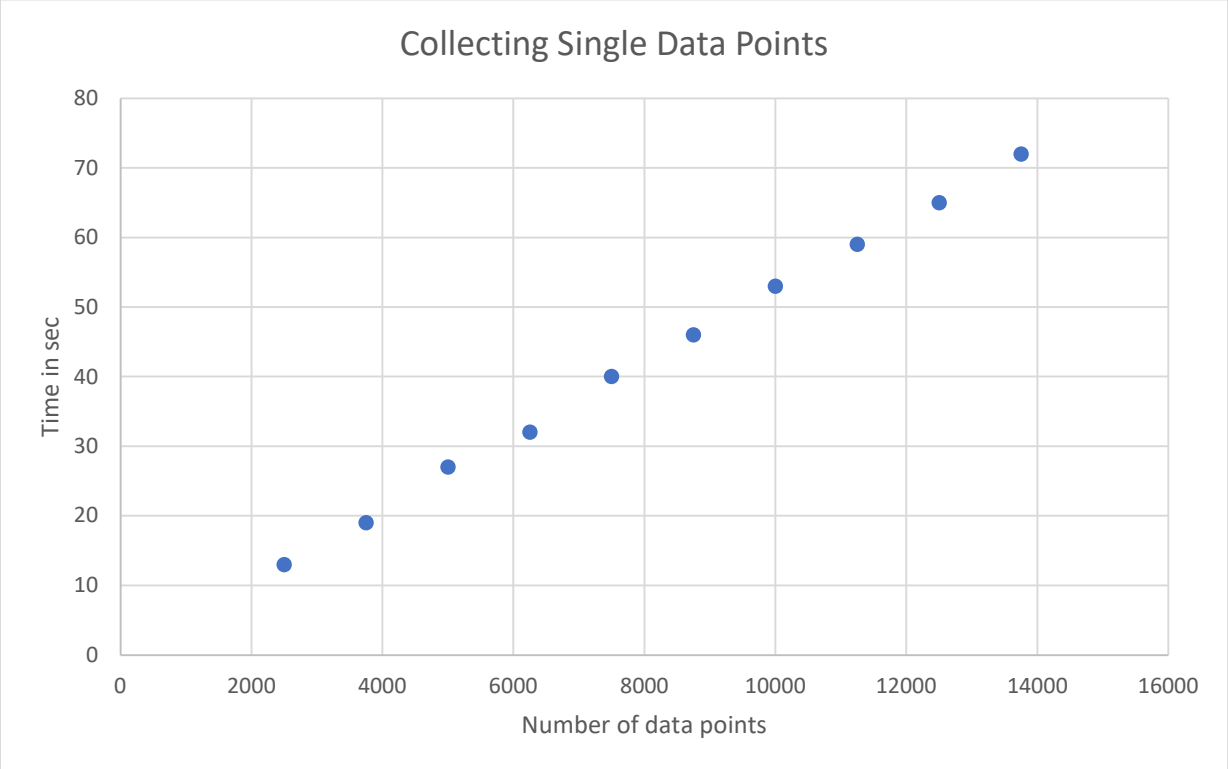
Scott Allen
Monday, June 1, 2020

```
if(patient_num == 0){
    TESTING PURPOSE
    Request at least 1000 data points for a person (both ecg1 and ecg2)
    collect the responces, and put them in a file called x1.csv.
    Compare the file against correspond data pionts in the original file
    and demonstrate that they match. ALso measure the time for collecing
    data points using gettingofday funciton, which has a microsecond granulity


    //ofstream myfile;
    //myfile.open("x1.csv");
    int datapoints_taken = 0;
    for(int X = 10; X <= 60; X +=5){
        struct timeval start, end;
        gettimeofday( &start, NULL);
        for(double time = 0.000; time < X; time += .004){
            datapoints_taken ++;
                //Grab the data points from the server
                datamsg *ecg1 = new datamsg( 1, time, 1);
                chan.cwrite(ecg1, sizeof(datamsg));
                double data1 = 0;
                chan.cread( &data1, sizeof(double));

                datamsg *ecg2 = new datamsg( 1, time, 2);
                chan.cwrite(ecg2, sizeof(datamsg));
                double data2 = 0;
                chan.cread(&data2, sizeof(double));

                //Write to the file
                //myfile << time <<","<<data1<<","<<data2<<endl;
        }
        //myfile.close();
        gettimeofday( &end, NULL);
        double time_taken = (end.tv_sec - start.tv_sec);
        cout << "Time_taken to get "<< datapoints_taken<< " points : " << time_taken << endl;
```



Scott Allen
Monday, June 1, 2020

Requesting Files (35pts)

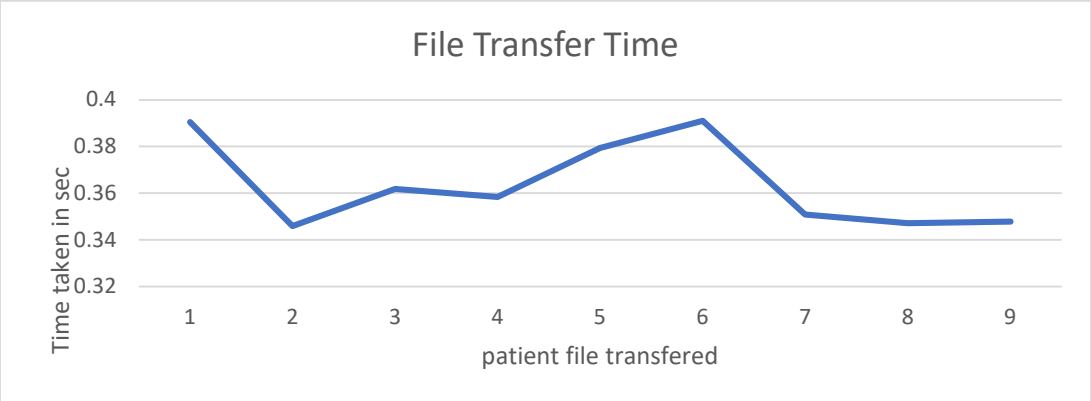
Requesting a file was done in a simiar manner. We wrote to the server and then created a location to store the data we recived from the server. From there we wrote to the new file 64 bits at a time. I recored the time of transfereing files. The graph is attached below.

```
//GETTING FILE MESSAGE (35 PTS)
else if(fdef){
    /*
    Request an entire file by first sending afile messag eto get ts lenght and
    then a series of file messages to get the actulal content of the file. Put
    recieved file under the "recieved/" directory with the same name as the
    orginal file. Compare the file agianst the origanl using linux command diff
    and demonstrate that they are the exact same. Measure the file transfer
    */
    FIFORequestChannel chan ("control", FIFORequestChannel::CLIENT_SIDE);
    cout << "FILENAME = " << filename << endl;
    struct timeval start, end;
    gettimeofday( &start, NULL);
    filemsg *msg = new filemsg(0,0);
    int request = sizeof(filemsg) + filename.size() + 1;
    char *buf = new char[request];
    memcpy(buf, msg, sizeof(filemsg));
    strcpy(buf + sizeof(filemsg), filename.c_str());
    chan.cwrite(buf, request);

    __int64_t filesize;
    chan.cread(&filesize, sizeof(__int64_t));

    string output_path = string("received/") + filename;
    FILE *f = fopen(output_path.c_str() , "wb");
    char *reciever = new char[MAX_MESSAGE];

    while( filesize > 0){
        int r_lenth = min((__int64_t)MAX_MESSAGE, filesize);
        ((filemsg*)buf)->length = r_lenth;
        chan.cwrite(buf, request);
        chan.cread(reciever, r_lenth);
        fwrite(reciever, 1, r_lenth, f);
        ((filemsg *)buf)->offset += r_lenth;
        filesize -= r_lenth;
    }
    fclose(f);
    gettimeofday( &end, NULL);
    double time_taken = (end.tv_usec - start.tv_usec);
    cout << "Time_taken to get the file : " << time_taken << endl;
    // closing the channel  (5 PTS)
    MESSAGE_TYPE m = QUIT_MSG;
    chan.cwrite (&m, sizeof (MESSAGE_TYPE));
}
```



Scott Allen
Monday, June 1, 2020

Requesting a New Channel (15pts)

We had to use the control channel first to create a new side channel. This took me a while to figure out because I was trying to store the new channel name as a string. I then realized I had to be using a char array to store. Once I created a new side channel I requested a single data point then closed both channels.

```
//REQUESTING A NEW CHANNEL (15 PTS)
else if (cdef){
    /*
    Ask the server to create a new channel for you tby sending a special NEWCHANNEL_MSG
    request and join that channel. After the channel is created, demonstrated that you
    can use that to speak to the server. Sending a few data points requests and recieving
    their responces is adequte for that demonstration.
    */
    FIFORequestChannel chan ("control", FIFORequestChannel::CLIENT_SIDE);
    MESSAGE_TYPE new_channel = NEWCHANNEL_MSG;
    chan.cwrite(&new_channel, sizeof(new_channel));
    char new_channel_name [30];
    chan.cread(&new_channel_name , sizeof(new_channel_name) );
    //cout << new_channel_name << endl;

    FIFORequestChannel new_chan (new_channel_name, FIFORequestChannel::CLIENT_SIDE);

    //using the new channel to get a data point
        datamsg *data = new datamsg( 1, 1, 1);
        new_chan.cwrite(data, sizeof(datamsg));
        double recived_val = 0;
        new_chan.cread( &recived_val, sizeof(double));
        cout << recived_val << endl;

    // closing the channels  (5 PTS)
    MESSAGE_TYPE m = QUIT_MSG;
    new_chan.cwrite (&m, sizeof (MESSAGE_TYPE));
    chan.cwrite (&m, sizeof (MESSAGE_TYPE));
}
```

Scott Allen
Monday, June 1, 2020

Run the server as a child process (15 pts)

Running the server as a child process was quite easy once I looked up a video on creating a fork. I created an a fork at the beginning of the program and then made an if/else statement asking for the PID. If the PID was 0 then the process was the parent process. Else it was child and I would run the server. I did not need a wait() because the server would stay open until a quit message was sent.

```
else{
    cout << "Running server as a child process pid = " << pid << endl;
    char* args[] = {"./server", NULL};
    //cout << args[0] <<endl;
    execv(args[0], args);


}
```