Joshua Chong
6-1-2020
CSCE 313 - 199
Dr. Tanzir
Due 6-1-2020 : 11:59 PM : CST

A Client Process Speaking to a Server Process

Prompt : Write a report describing the design, and the timing data you collected for data points, text file, and binary files. Compare the difference in time between transferring data points vs entire file.

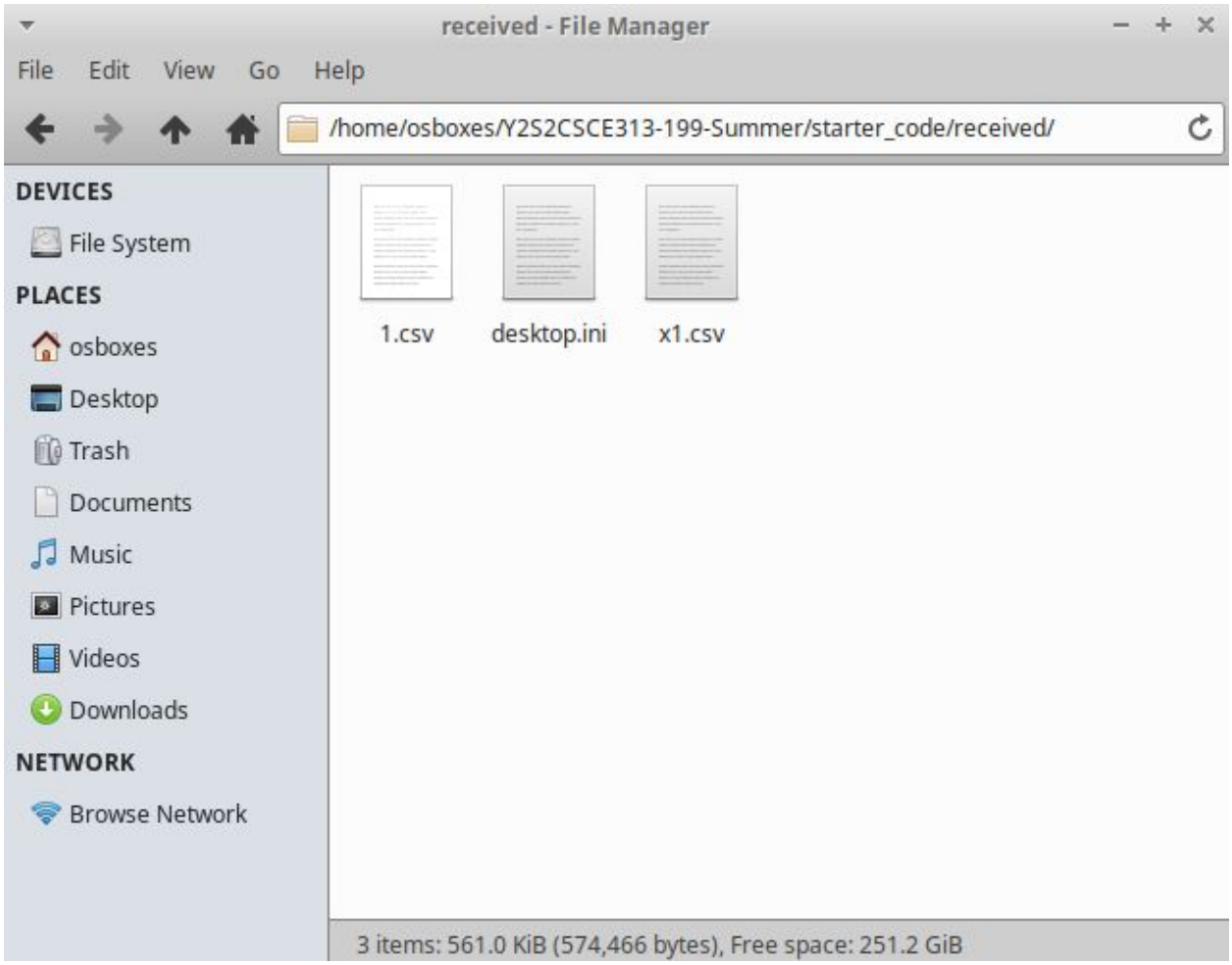
Requesting Data Points: (15 pts)
Relevant Code :

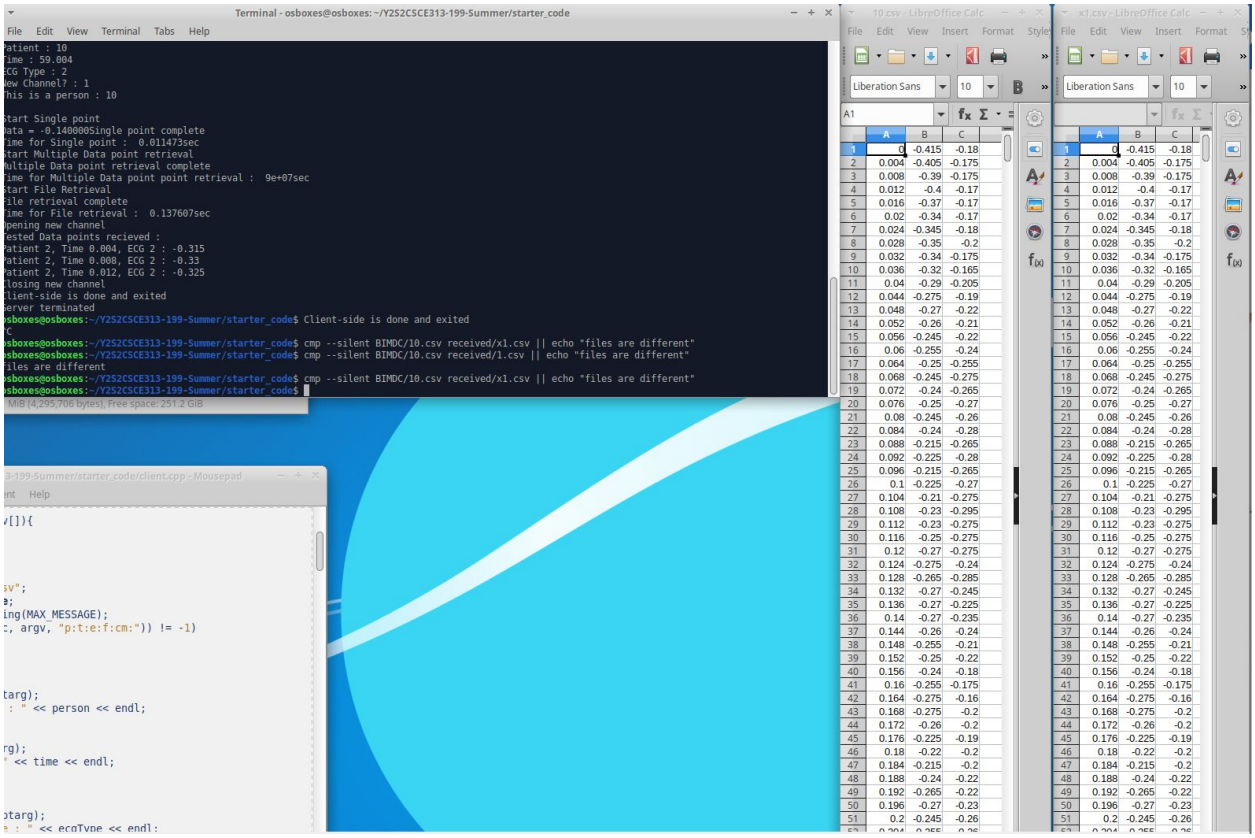
```
// Start 1000 Data point test
    struct timeval s1,e1;
    cout << "Start Multiple Data point retrieval" << endl;
    gettimeofday(&s1, NULL);
    ofstream mF;
    mF.open("received/x1.csv", ios::binary);
    double t = 0;
    while (t < 59.996){
        mF << t << ",";
        datamsg ec1 = datamsg(person, t, 1);
        datamsg ec2 = datamsg(person, t, 2);
        chan.cwrite(&ec1,sizeof(ec1));
        double d1 = 0;
        chan.cread((char*)&d1,sizeof(double));
        mF << d1 << ",";
        chan.cwrite(&ec2,sizeof(ec2));
        double d2 = 0;
        chan.cread((char*)&d2, sizeof(double));
        mF << d2 << endl;
        t += 0.004;
    }
    mF.close();
    gettimeofday(&e1,NULL);

    cout << "Multiple Data point retrieval complete" << endl;
```

```
    cout << "Time for Multiple Data point point retrieval :  " <<
(e1.tv_sec - s1.tv_sec)*1e6 + (e1.tv_usec -s1.tv_usec)*1e-6 << "sec" <<
endl;
    // Stop 1000 Data point test
```

Input Output :

```
osboxes@osboxes:~/Y2S2CSCE313-199-Summer/starter_code$ ./client -p 10 -t 59.004 -e 2 -c
Patient : 10
Time : 59.004
ECG Type : 2
New Channel? : 1
This is a person : 10

Start Single point
Data = -0.140000Single point complete
Time for Single point :  0.011473sec
Start Multiple Data point retrieval
Multiple Data point retrieval complete
Time for Multiple Data point point retrieval :  9e+07sec
Start File Retrieval
File retrieval complete
Time for File retrieval :  0.137607sec
Opening new channel
Tested Data points recieved :
Patient 2, Time 0.004, ECG 2 : -0.315
Patient 2, Time 0.008, ECG 2 : -0.33
Patient 2, Time 0.012, ECG 2 : -0.325
Closing new channel
Client-side is done and exited
Server terminated
```


received - File Manager
File Edit View Go Help
/home/osboxes/Y2S2CSCE313-199-Summer/starter_code/received/
DEVICES
File System
PLACES
osboxes
Desktop
Trash
Documents
Music
Pictures
Videos
Downloads
NETWORK
Browse Network
1.csv
desktop.ini
x1.csv
3 items: 561.0 KiB (574,466 bytes), Free space: 251.2 GiB



Cmp --silent BIMDC/10.csv received/x1.csv || echo “files are different”
Checks byte by byte for a difference in content, file 10 is compared because the person input was Patient 10 taken from one of the input tests within the handout. The above line input outputs “files are different” if a discrepancy exists. An example case of a difference in file is the 1.csv copy that is done within the later code when copying file rather than just by data point and is different from 10.csv.

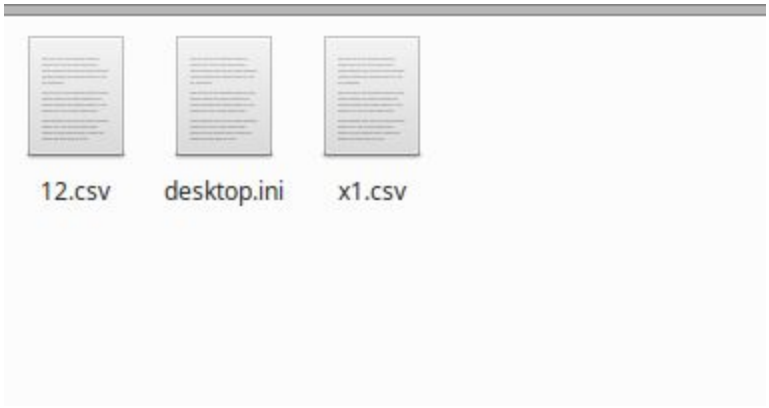
Requesting Files: (35 points)
What is the main bottleneck here? Can you change the transfer time by varying the bottleneck?
The main bottleneck is the buffer capacity that, if a request is larger than its capacity, will return nothing in response. Transfer time can change by changing the buffer capacity.
Relevant Code :

```
    // Start File Retrieval test
    struct timeval s2,e2;
    cout << "Start File Retrieval" << endl;
    gettimeofday(&s2, NULL);
    // offset = 0, length = 0
    filemsg *f0 = new filemsg(0,0);
    string fNameReq = fileName;
    int req = sizeof(filemsg) + fNameReq.size() + 1;
```

```
    char *buf = new char[req];
    memcpy(buf, f0, sizeof(filemsg));
    strcpy(buf+sizeof(filemsg), fNameReq.c_str());
    chan.cwrite(buf, req);

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

    string output_path = string("received/" + fNameReq);
    FILE *f = fopen(output_path.c_str(), "wb");
    char *receiver = new char[MAX_MESSAGE];
    while (fSize > 0 ){
        int req_len = min((__int64_t)MAX_MESSAGE, fSize);
        ((filemsg *)buf)->length = req_len;
        chan.cwrite(buf, req);
        chan.cread(receiver, req_len);
        fwrite(receiver, 1, req_len, f);
        ((filemsg *)buf)->offset += req_len;
        fSize -= req_len;
    }
    fclose(f);
    delete buf;
    delete receiver;
    gettimeofday(&e2, NULL);
    cout << "File retrieval complete" << endl;
    cout << "Time for File retrieval :  " << (e2.tv_sec - s2.tv_sec)*1e6 +
(e2.tv_usec -s2.tv_usec)*1e-6 << "sec" << endl;
    // Stop File Retrieval test
```

Input Output :

```
osboxes@osboxes:~/Y2S2CSCE313-199-Summer/starter_code$ ./client -p 10 -t 59.004 -e 2 -f 12.csv -c
Patient : 10
Time : 59.004
ECG Type : 2
Name of File : 12.csv
New Channel? : 1
This is a person : 10

Start Single point
Data = -0.140000Single point complete
Time for Single point :  0.005071sec
Start Multiple Data point retrieval
Multiple Data point retrieval complete
Time for Multiple Data point point retrieval :  8.8e+07sec
Start File Retrieval
File retrieval complete
Time for File retrieval :  0.034739sec
Opening new channel
Tested Data points recieved :
Patient 2, Time 0.004, ECG 2 : -0.315
Patient 2, Time 0.008, ECG 2 : -0.33
Patient 2, Time 0.012, ECG 2 : -0.325
Closing new channel
Client-side is done and exited
Server terminated
osboxes@osboxes:~/Y2S2CSCE313-199-Summer/starter_code$ Client-side is done and exited
diff BIMDC/12.csv received/12.csv
osboxes@osboxes:~/Y2S2CSCE313-199-Summer/starter_code$
```



(20 pts) : diff was used to compare the two files, and reported that there was no difference from a successful run.
(10 pts) : fopen() under the action “wb” according to ibm, opens an empty binary file or an existing one that is overwritten. Binary requirement is satisfied.
(5 pts) : Large file transfer complete.

Requesting a New Channel: (15 pts)
Relevant Code :

```
// Start New channel if needed
    if (newChannel){
        nCmsg newC = nCmsg();
        cout << "Opening new channel" << endl;
```

```
        chan.cwrite((char*)&newC, sizeof(nCmsg));
        FIFORequestChannel nC ("data1_", FIFORequestChannel::CLIENT_SIDE);
        cout << "Tested Data points recieved : " << endl;
        datamsg test1 = datamsg(2,0.004,2);
        nC.cwrite(&test1,sizeof(test1));
        double testD1 = 0;
        nC.cread((char*)&testD1, sizeof(double));
        cout << "Patient 2, Time 0.004, ECG 2 : " << testD1 << endl;
        datamsg test2 = datamsg(2,0.008,2);
        nC.cwrite(&test2,sizeof(test2));
        double testD2 = 0;
        nC.cread((char*)&testD2, sizeof(double));
        cout << "Patient 2, Time 0.008, ECG 2 : " << testD2 << endl;
        datamsg test3 = datamsg(2,0.012,2);
        nC.cwrite(&test3,sizeof(test3));
        double testD3 = 0;
        nC.cread((char*)&testD3, sizeof(double));
        cout << "Patient 2, Time 0.012, ECG 2 : " << testD3 << endl;
        MESSAGE_TYPE nCQuit = QUIT_MSG;
        cout << "Closing new channel" << endl;
        nC.cwrite(&nCQuit, sizeof(MESSAGE_TYPE));
    }
```

Input is derived from the previous question's input/ouput :

```
Opening new channel
Tested Data points recieved :
Patient 2, Time 0.004, ECG 2 : -0.315
Patient 2, Time 0.008, ECG 2 : -0.33
Patient 2, Time 0.012, ECG 2 : -0.325
Closing new channel
Client-side is done and exited
```

A few data points were tested as requested and successfully opened and closed the new channel that was opened due to "-c" in the command line input.

Run the Server as a child process (15 pts)

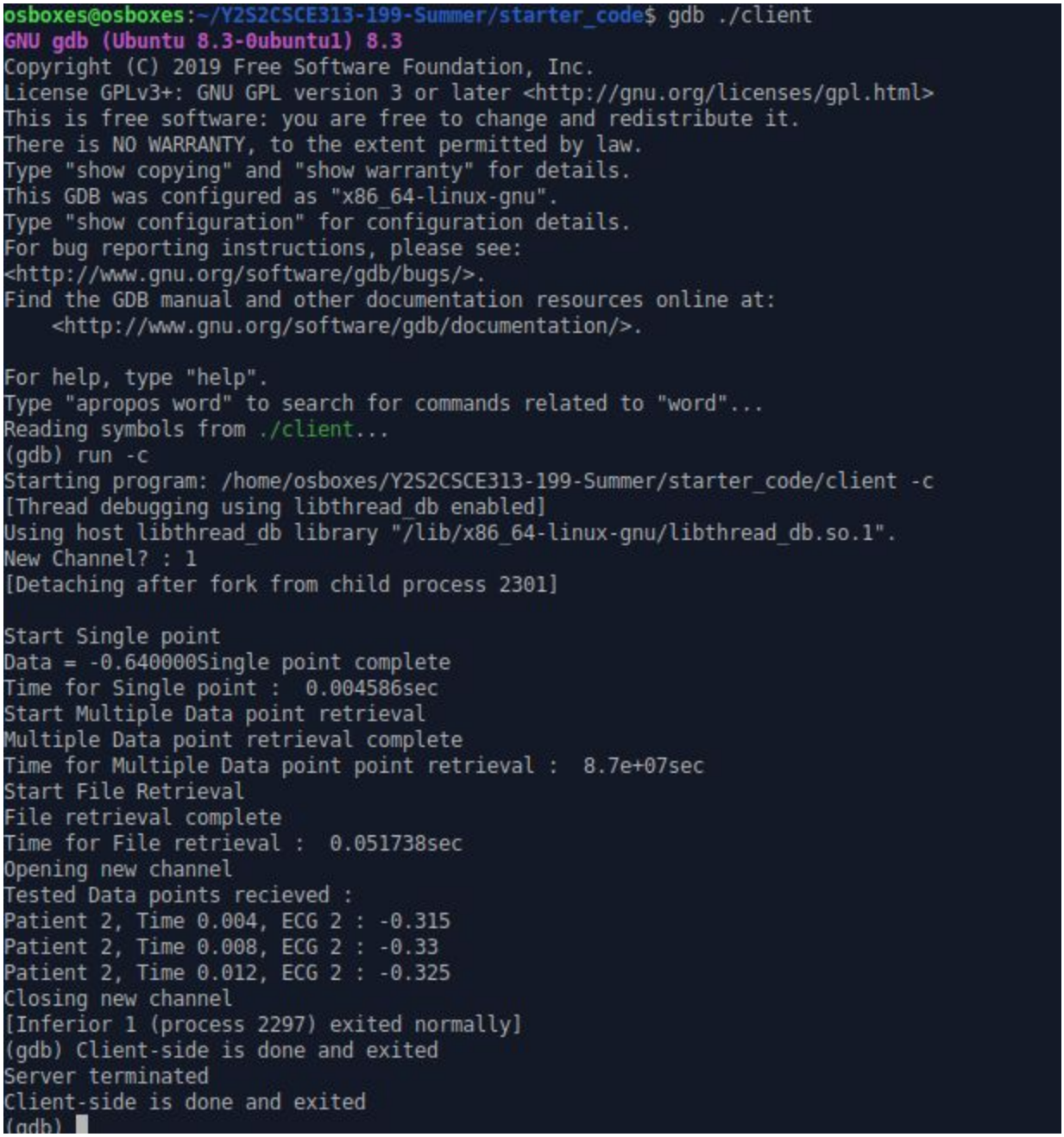
Relevant Code :

```
int main(int argc, char *argv[]){
      int opt;
      int person = 1;
      double time = 0;
      int ecgType = 0;
```

```
    string fileName = "1.csv";
    bool newChannel = false;
    string bufCap = to_string(MAX_MESSAGE);
  while ((opt = getopt(argc, argv, "p:t:e:f:cm:")) != -1)
  {
      switch (opt)
      {
      case 'p':
          person = atoi(optarg);
          cout << "Patient : " << person << endl;
          break;
      case 't':
          time = stod(optarg);
          cout << "Time : " << time << endl;

          break;
      case 'e':
          ecgType = atoi(optarg);
          cout << "ECG Type : " << ecgType << endl;

          break;
      case 'f':
          fileName = optarg;
          cout << "Name of File : " << fileName << endl;

          break;
      case 'c':
          newChannel = true;
          cout << "New Channel? : " << newChannel << endl;

          break;
      case 'm':
          bufCap = optarg;
          cout << "Memory Capacity : " << bufCap << endl;

      case '?':
          cout << "Incorrect input" << endl;
```

```
            break;
        default:
            cout << "No input" << endl;
        }
    }
    int pid = fork();
    if (pid == 0){
        // Got points off last time I didn't auto summon rip

        bufCap = "./server -m " + bufCap;
        char *serverEntry = new char[bufCap.size() +1];
        copy(bufCap.begin(),bufCap.end(),serverEntry);
        serverEntry[bufCap.size()] = '\0';

        char *args[] = { serverEntry, NULL};
        execvp(args[0], args);
    }
```

Input Output :



Client successfully forked as shown.
Input : gdb ./client
Run -c

Closing Channels (5 pts)
Channels successfully closed as seen in the prior problem's evidence.

Report's answer :
The design I took to collect the data points was to have 2 data messages for both ecg1 and ecg2 and cycle through until the end of file whose time was 59.996 seconds. The text and

binary files were copied part by part depending on the buffer capacity and accounted for the offset based on the req_len. The difference in time between the transferring of data points vs the entire file is significantly shorter when copying by file since it is able to copy it by chunks rather than just data points alone with one example being Multiple Data Points (The entire file for convenience) took 8.7e+07 seconds while File retrieval took only 0.51738 seconds.

Extra :

Input and Output for -p 10 -t 59.004 -e 2 when run under gdb