# IOE Server Flow-Chart

We first define a Class named Files, which has 4 attributes, “name”, “Path”, ”size”, and “type”

```
class Files(object):
    name = None
    full_path = None
    size = None
    type = None
```

Define a funtion “getList” to loop through all the file in the current dictory, for each file we find, a new “Files” object is created and added to “List_of_files” for storing the information needed for a transfer and the “LIST” function.

```
def getList():
    List_of_files = []
    curruent_path = os.getcwd()
    file_found = False
    response = ''
    for files in os.listdir(curruent_path):
        new_file = Files()
        new_file.name = files
        new_file.full_path = curruent_path + "\\" + files
        new_file.size = os.path.getsize(curruent_path + "\\" + files)
        fileName, fileExtension = os.path.splitext(new_file.full_path)
        if (fileExtension == '.mov'):
            new_file.type = "viedo"
            List_of_files.append(new_file)
        elif (fileExtension == '.mp3'):
            new_file.type = "music"
            List_of_files.append(new_file)
        elif (fileExtension == '.jpg'):
            new_file.type = "picture"
            List_of_files.append(new_file)
    return List_of_files
```

In the main function, we declare a host and port, and we start the threaded server by calling ThreadedTCPServer((HOST, PORT), ThreadedTCPRequestHandler). “ThreadedTCPRequestHandler” is our self-defined class for handling each incoming request.

```
if __name__ == "__main__":
    quit_server = False
    # Port 0 means to select an arbitrary unused port
    HOST, PORT = "localhost", 10000

    print "\nStart Threaded-Server on PORT %s " % PORT

    server = ThreadedTCPServer((HOST, PORT), ThreadedTCPRequestHandler)
    ip, port = server.server_address
```

The server is started by setting a variable to “threading.Thread(target=server.serve_forever” and calling the start() function. Daemon is set to true so the server will shutdown when this main thread exits.

```
server_thread = threading.Thread(target=server.serve_forever)

# Terminate the server when the main thread terminates
# by setting daemon to True
server_thread.daemon = True
server_thread.start()
```



Jingnan Chen 1073196, Wei Zhang 0951321

# IOE Server Flow-Chart

A while true loop and raw_input is used to detect if the user has entered the “QUIT” command, If “QUIT” has been entered, and if there are no other threads( no connections) then the server will close by calling shutdown() and quit(). If there are connections, the main thread waits but giving the option to forcefully shut down.

```
while True:
    command = raw_input("enter quit to exit server: \n")
    if(command == "QUIT"):
        if(thread_counter ==0):
            print 'Main server thread shutting down the server and terminating'
            server.shutdown()
            quit()
        else:
            print 'Waiting for threads to finish...'
            while(thread_counter !=0):
                force_comment = raw_input("Type FORCEQUIT to type abruptly. \n")
                if(force_comment == "FORCEQUIT"):
                    print 'Bye'
                    os._exit(0)
                quit_server = True
            quit()
```

If the data startswith “LIST” call the “getList” self-defined function and then first loop through the “List_of_file” to append all the nessacry information to response and send the response using “self.request.sendall”

If “List_of_file”, then send “There is no files in the directory” instead.

```
if(data.startswith('LIST')):
    print "\nLIST Command From %s" % thread_name
    response = ""
    List_of_files=getList()
    for files in List_of_files:
        #response = response + str(files.name) + " size: " + str(files.size) + " path: " + str(file
        response = response + str(files.type)+ "\t " + str(files.name) + "\t size: " + str(files.s
    if(response == ""):
        self.request.sendall("There is no files in the directory")
    else:
        self.request.sendall(response)
        response = ""
    print "LIST Task Done for %s" % thread name
```



Jingnan Chen 1073196, Wei Zhang 0951321

# IOE Server Flow-Chart

If the data startswith “READ” then;

First loop through the “List_of_file” to find if the file that the client is trying to read exists, if not send back an error message.

```
elif(data.startswith('READ')):
    List_of_files=getList()
    if(len(data.split(',')) == 2):
        filename = data.split(',')[1]
        file_found = False
        for files in List_of_files:
            if(files.name == filename):
                file found = True
```

Second, send the file size of the file that the client is trying to read to the client, so that it knows when the received file is completed and it can use the size to detect broken files.

```
if(files.name == filename):
    file_found = True
    self.request.sendall(str(files.size))
```

Third, open the file in read byte mode and for each line in the file send that line using self.request.sendall(line) and close the file in the end

```
        f1 = open(files.name, 'rb')
        for line in f1:
            self.request.sendall(line)
        f1.close()
```

If the data startswith “WRITE” then;

Open a file using the open method with the filename receieved and set amount_received variable to 0. Using a while(amount_received < filesize) , a try and catch statement for suddent interrupted connection, and to receive the data by calling self.request.recv(), then write the data received to the file using the write() method. If anything goes wrong, statements in the except block is excuted, closing the file and deleting the broken file.

```
while(amount_received < filesize):
    try:
        mess = self.request.recv(64)
        if mess:
            #print '\nServer Thread recevied %s' % mess
            f1.write(mess)
            amount_received += len(mess)
            print "AR: " + str(amount_received) + " size: " + str(filesize)
        else:
            f1.close()
            break
    except:
        f1.close()
        os.remove(f1.name)
        break

if(amount_received == filesize):
    print "Done Receiving"
    self.request.sendall("From Server: Recevied File: " +str(filename))
```



Jingnan Chen 1073196, Wei Zhang 0951321

# IOE Server Flow-Chart

If the data startswith “BYE” then;

Break out of the while loop, ending the handler function

```
elif(data == "BYE"):
    break;
```

The last few statements in the handler class is for keeping count of how many connections there are. Whenever a connection ends, decreament the thread_counter variable. If the thread count is 0, then return to the main thread.

```
        elif(data == "BYE"):
            break;

thread_counter -= 1
print "" + str(thread_counter)
if(thread_counter == 0):
    return
```



Jingnan Chen 1073196, Wei Zhang 0951321