Report

CMPE156 Name: Shuli He, ID:she77@ucsc.edu

How to use:

Client:

./rsclient <ipaddress> <port>

example: ./rsclient 127.0.0.1 1234

Server:

./rsserver <port>

example: ./rsserver 1234

Main function:

Input port is verified by portVarify（）function.

```
portVarify(const char* port)
```

Client:

1. Use the socket to create connection with server. Set the time out by

```
int ret_send=setsockopt(sockfd,SOL_SOCKET,SO_SNDTIMEO,(const
char*)&timeout_send,sizeof(timeout_send));
int ret_recv=setsockopt(sockfd,SOL_SOCKET,SO_RCVTIMEO,(const
char*)&timeout_recv,sizeof(timeout_recv));
```

2. Use the **readline()** function to get the input command and send it through the socket. (Limited the command size to 1024 bytes). At last free the readline.

3. Then wait for the reply from server and receive the data by recv(). If some network error and reach the timeout, stop recv().
4. The client print out the return data and then waiting for another command. If exit, close the connection socket.

Server:

1. Create the server socket. Bind the listening port with the server and wait for the coming command.

```
bind(listenfd, (struct sockaddr*)&servaddr, sizeof(servaddr))
```

2. When accept a socket connection, start to receive command from client.

```
connfd = accept(listenfd, (struct sockaddr*)NULL, NULL)
```

3. Handle the command use **popen()** function to get the stdout of the command. Add “2>&1” to the command to get the stderr.
4. Then send back the popen out put to the client and wait for next command. If exit received, close the socket with client and wait from next client connection.