Assignment 1 Documentation

1.Application Protocol

Client:

At the very beginning, while executing the client program, the client is required to specify the IP address and port number. Otherwise the program will pop out an error message to remind the client to input such variables.

```
if (argc != 3)
        err_quit("usage: a.out <IPaddress> <Port Number>");
```

At first we create an infinite loop so that the client can keep entering commands.

```
while(1){
```

Client is asked to enter commands. If the client enters “exit” , the program will jump out of the while loop and the program will be terminated.

```
        printf("Please enter:\n");
        //capture user input
        fgets(buf, MAXLINE, stdin);
        if(strcmp(buf,test)==0) break;
```

After that the client program will send the command that user enters to server.

```
        //send command to server
        Write(sockfd,buf,strlen(buf));
```

If the server did response, the program will read the response and print it out.

```
        //read server response
        if((n = read(sockfd, recvline, MAXLINE))>0)
            recvline[n] = 0;
            if (fputs(recvline, stdout) == EOF)
                err_sys("fputs error");
```

On the other hand, if client fail to receive response from server, it will pop out an error message.

```
        if(n<0)
            err_sys("read error");
    }
```

Server:

Also, at the very beginning, while executing the server program, the client is required

to specify the port number. Otherwise the program will pop out an error message to remind the client to input such variable.

```
if(argc !=2)
        err_quit("usage: a.out <Port Number>");
```

After TCP connection, the server will wait until there is a connection from client.

```
for ( ; ; ) {
        connfd = Accept(listenfd, (SA *) NULL, NULL);
```

The server will start reading message from the client after connection is established. If it receive message from the client, it will store it in the variable “recvline”.

```
        //Read and Write Loop Start----
        while ( (n = read(connfd, recvline, MAXLINE)) > 0){
            recvline[n] = 0;//null terminate
            char path[MAXLINE];

            //execute bash command
            fp = popen(recvline,"r");
```

The program will retrieve the result responded from the bash and store in the variable “path”. It might not get the result all in once, so I create a while loop and concatenate the result onto variable “buff” before `fgets` returns `NULL`

```
            //retrieve response
             while (fgets(path, PATH_MAX, fp) != NULL){
                strncat(buff, path,strlen(path));
            }
            printf("%s",buff);
```

Later on, the server will send the result to the client. Besides the program will clear out the data store in the variable “buff”

```
            Write(connfd,buff,strlen(buff));
            bzero(buff,strlen(buff));
            pclose(fp);
        }
        //Read and Write Loop Ends
```

The server will close connection if the client terminate it.

```
printf("%s\n","connection ended");
Close(connfd);
```

2. Test File Usage:

1.change direction to HW1/bin

2.enter$ sudo ./test_main (The IP is set to 127.0.0.1. The port number is 1234)

3.There will be two windows pop out. Please enter command in the “myclient” window.

4.The window will print “The result is valid” if the local result correspond the result that the server returns.

Explanation:

1.I have a hard time retrieve the output result directly from the “myclient” program and compare with local result. Therefore, I wrote another C program called “test_client” and I compare the local result and the server result first in that program.

2.The “test_main” program is just execute “myserver” and “test_client” together and show the result.