**TTP 3181- TCP/IP Programming**

T1 2014/2015

Project Title: TCP/IP ASSIGNMENT 2

Leave itblank for this column

Group ID: cpu02

|  |  |  |
| --- | --- | --- |
| **IPC (Choose 2)** |  |  |
| **Pipes/FIFO** | +1 |  |
| **Message Queue** | +2 |  |
| Semaphore | +2 |  |

|  |  |  |
| --- | --- | --- |
| **Processes Control (Choose 2)** |  |  |
| **fork()** | +2 |  |
| **exec()** | +2 |  |
| **Signals** | +2 |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Socket (Choose 1)** |  |  |
| **TCP** | +3 |  |
| UDP | +3 |  |

|  |  |  |
| --- | --- | --- |
| **Others** |  |  |
| **File Manipulation (as Database)** | +2 |  |
| **I/O Multiplexing** | +3 |  |

|  |  |  |
| --- | --- | --- |
| **Miscellaneous** |  |  |
| Bugs-Free and No Logical Error | +2 |  |
| **Documentation** |  |
| **Modular programming technique and good practice** |  |
| Overall Performance | +2 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Student ID | Email address | Signature |
| Tomasi Samuel Asea | 1112702030 | aseasamuel@gmail.com |  |
| Aidin Talaeizad | 1101110243 |  |  |

Contents

[Table of Figures 3](#_Toc396725701)

[1 Introduction 4](#_Toc396725702)

[1.1 Overview 4](#_Toc396725703)

[1.2 Diagram 5](#_Toc396725704)

[2 Background 6](#_Toc396725705)

[3 Solution 6](#_Toc396725706)

[3.1 Limitations 6](#_Toc396725707)

[3.2 Screen Shots 6](#_Toc396725708)

[3.3 Features 8](#_Toc396725709)

[APPENDIX 9](#_Toc396725710)

[Source Code 10](#_Toc396725711)

[Client.c 10](#_Toc396725712)

[user.c 13](#_Toc396725713)

[admin.c 13](#_Toc396725714)

[qoperate.h 14](#_Toc396725715)

[queue.h 15](#_Toc396725716)

[inet.h 15](#_Toc396725717)

[Server.c 16](#_Toc396725718)

# Table of Figures

[Figure 1 System Diagram 6](#_Toc396722683)

[Figure 2 Main Interface 7](#_Toc396722684)

[Figure 3 User Interface 8](#_Toc396722685)

[Figure 4 Admin Interface 8](#_Toc396722686)

# 1 Introduction

# Overview

This project provides the user the ability to use a shopping system using TCP/IP client server method, it provides the user with the ability to perform basic tasks such as viewing the products and signup and signin, it also has a admin interface where he can Add products and view users,

The system uses TCP as its means of connection and uses system call to perform the required tasks, and it uses io multiplexing for multiple client support.

The Users of the System can be from any background as long as they have basin knowledge in using C language interface and are ready to use old user interfaces.

## Diagram

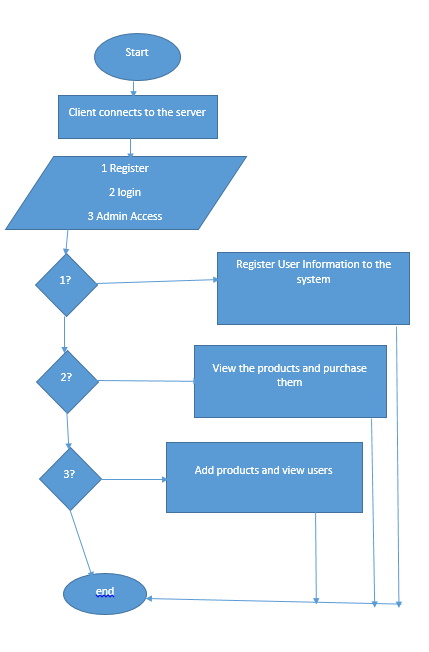


Figure 1 System Diagram

# Background

Online shops are provided by many services such as Amazon and ebay internationally. And Lazada locally. These shops provide the convenience of security and accessibility to a wide range of products. All the purchased products can be reached by mail to the user within a specified amount of time.

# Solution

## Limitations

he System does not use the full features of TCP IP connection and system call, it also has some logical errors in its functionality.

## Screen Shots

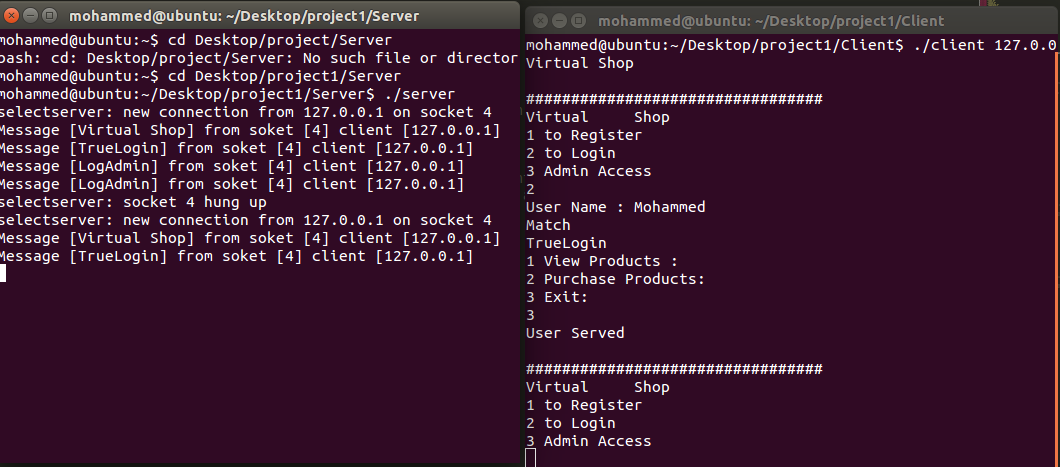


Figure 2 Main Interface

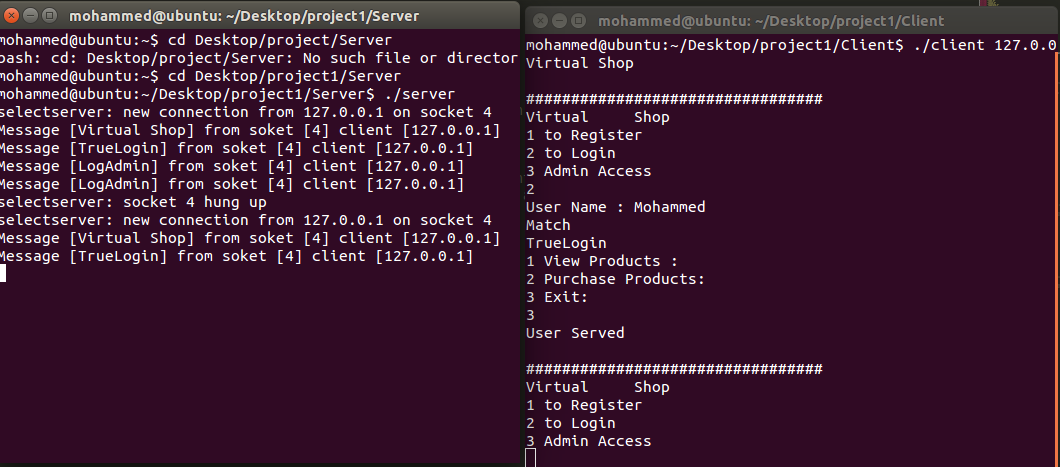


Figure 3 User Interface

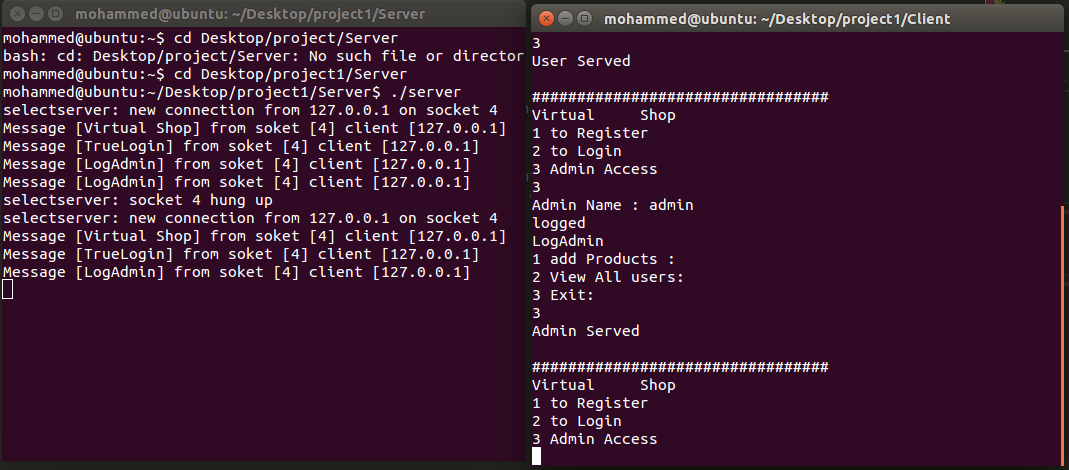


Figure 4 Admin Interface

## Features

1. Two login privileges, one for users and one for administrators.
2. Users are able to view the list products.
3. admin are able to add the product they like into the system and view all the users.

# APPENDIX

# Source Code

# Client.c

#include "inet.h"

#include "queue.h"

#include "qoperate.h"

#include <string.h>

#define BUFSIZE 1024

main(int argc, char \*argv[]){

int sockfd;

int filefd, nbyte;

int pID, pID1;

int i, j, x, y;

char buffer[BUFSIZE+1], choice;

struct sockaddr\_in serv\_addr;

int pipefd[2];

char \*msg = "Virtual Shop", \*msgU = "TrueLogin", \*msgA = "LogAdmin";

char str[20];

char pass[20] = "admin";

char User[20];

char admin[20];

char flag = 'T';

if(argc <= 1){

printf("How to use : %s remoteIPaddress [example: ./client 127.0.0.1]\n", argv[0]);exit(1);}

bzero( (char \*)&serv\_addr, sizeof(serv\_addr) ); //

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_port = htons (SERV\_TCP\_PORT);

inet\_pton (AF\_INET, argv[1], &serv\_addr.sin\_addr);

if( (sockfd = socket(AF\_INET, SOCK\_STREAM, 0) ) < 0) {

perror("Client: socket() error\n");

exit(1);}

if(connect(sockfd,(struct sockaddr \*)&serv\_addr,sizeof(serv\_addr))<0){

perror("Client: connect() error\n");

exit(1);}

if ( pipe(pipefd) ==-1) { //if the pipe is not created, then it is a pipe error

perror("pipe error");

exit(1); }

write (pipefd[1], msg, BUFSIZE);//writes msg1 from above into the pipe

read(pipefd[0], buffer, BUFSIZE);//retrieves msg1 from pipe and place it into buffer

send(sockfd,buffer,BUFSIZE,0);

printf("%s\n", buffer);

bzero(buffer, sizeof(buffer));

do{

while(flag == 'T'){

printf("\n#################################\n");

printf("Virtual Shop \n");

printf("1 to Register\n");

printf("2 to Login \n");

printf("3 Admin Access\n");

static struct sigaction act;

act.sa\_handler = SIG\_IGN;

sigfillset (&(act.sa\_mask));

sigaction (SIGINT, &act, (void \*)0);

scanf("%s", &choice);

switch(choice){

case '1':

filefd = open("register.txt", O\_WRONLY|O\_APPEND, 0755);

write(1, "User Name- Not more than 20",27);

nbyte = read(0,buffer, 20);

wait();

write(filefd, buffer, nbyte);

close(filefd);

filefd = open("register.txt", O\_RDONLY);

read(filefd, buffer, 100);

printf("%s",buffer);

bzero(buffer,sizeof(buffer));

close(filefd);

break;

case '2':

write(1, "User Name : ",12);

memset(&buffer, 0, sizeof(buffer));

scanf("%s", User);

FILE \*fp = fopen("register.txt", "r");

rewind(fp);

while(fscanf(fp, "%s", str) != EOF){

i = strlen(str);

j = strlen(User);

if(i == j){

printf("Match\n");

if ( pipe(pipefd) ==-1) { //if the pipe is not created, //then it is a pipe error

perror("pipe error");

exit(1); }

memset(&buffer, 0, sizeof(buffer));

write (pipefd[1], msgU, BUFSIZE);//writes msg1 from above //into the pipe

read(pipefd[0], buffer, BUFSIZE);//retrieves msg1 from //pipe and place it into buffer

printf("%s\n", buffer);

pID1 = fork();

if(pID1 == 0){

execl("user", "user", (char\*)0);

}

else{

wait((int \*)0);

printf("User Served\n");}

flag = 'F';

break;}

else

printf("Nope\n");

}

fclose(fp);

break;

case '3':

memset(&buffer, 0, sizeof(buffer));

write(1, "Admin Name : ",13);

scanf("%s", admin);

x = strlen(pass);

y = strlen(admin);

if(x == y){

printf("logged \n");

if ( pipe(pipefd) ==-1) { //if the pipe is not created, then it //is a pipe error

perror("pipe error");

exit(1); }

bzero(buffer,sizeof(buffer));

write (pipefd[1], msgA, BUFSIZE);//writes msg1 from above into //the pipe

read(pipefd[0], buffer, BUFSIZE);//retrieves msg1 from pipe and //place it into buffer

printf("%s\n", buffer);

pID = fork();

if(pID == 0){

execl("admin", "admin", (char\*)0);

//execl("/bin/ls", "ls", "-al", (char \*)0);

}

else{

wait((int \*)0);

printf("Admin Served\n");}

flag = 'F';}

else{

printf("wrong\n");}

break;

}

}

send(sockfd,buffer,BUFSIZE,0);

flag = 'T';

}while (strcmp(buffer,"/q"));

close(sockfd);

}

# user.c

#include <stdio.h>

#include <stdlib.h>

#include "queue.h"

#include "qoperate.h"

int main(){

char flag = 'T';

char choice;

pid\_t pid;

while(flag == 'T'){

printf("1 View Products :\n");

printf("2 Purchase Products:\n");

printf("3 Exit:\n");

scanf("%s", &choice);

switch (choice){

case '1':

switch (pid = fork()){

case 0:

serve();

break;

case -1:

warn("Fork error");

break;

default:

printf("PID for parent is %d\n", pid);

}

exit (pid != -1 ? 0 : 1);

break;

case 2:

break;

case 3:

flag = 'F';

break;

}

}

}

# admin.c

#include <stdio.h>

#include <stdlib.h>

#include "queue.h"

#include "qoperate.h"

int main(){

int priority = 1;

char product[20];

char flag = 'T';

FILE \*fp;

char choice;

char str[20];

while(flag == 'T'){

printf("1 add Products :\n");

printf("2 View All users:\n");

printf("3 Exit:\n");

scanf("%s", &choice);

switch (choice){

case '1':

if(priority == 10){

priority = 1;

}

printf("Product Name : ");

scanf("%s", product);

if(enter (product, priority) < 0){

warn("enter fails");

exit(3);}

priority += 1;

break;

case 2:

fp = fopen("register.txt", "r");

rewind(fp);

while(fscanf(fp, "%s", str) != EOF){

puts(str);

}

break;

case 3:

flag = 'F';

break;

}

}

}

# qoperate.h

int enter (char \*objname, int priority){

int len, s\_qid;

struct q\_entry s\_entry;

if((len = strlen(objname)) > MAXOBN){ //check object length

warn("invalid priority level");

return (-1);}

if(priority > MAXPRIOR || priority <0){

warn("invalid priority level");

return (-1);}

if((s\_qid = init\_queue()) == -1) //check if queue ID match

return(-1);

s\_entry.mtype = (long) priority;

strncpy (s\_entry.mtext, objname, MAXOBN);

if(msgsnd (s\_qid, &s\_entry, len, 0) == -1){ //send message

perror("Msgsnd fails");

return (-1);}

else

return (0);

}

int warn(char \*s){

fprintf(stderr, "Warning:%s\n",s);}

init\_queue(void){

int queue\_id;

if((queue\_id = msgget(ftok("\temp",9), IPC\_CREAT| QPERM)) == -1)//pass pathname and ID

perror("Msgget fails");

return(queue\_id);}

int serve(void){

int mlen, r\_qid;

struct q\_entry r\_entry;

if( (r\_qid = init\_queue()) == -1) //check if queue ID match

return (-1);

for(;;)

{

if((mlen = msgrcv(r\_qid, &r\_entry,MAXOBN, (-1\* MAXPRIOR\* 0),MSG\_NOERROR))==-1){

perror("msgrcv faild");

return(-1);}

else{

r\_entry.mtext[mlen] = '\0';

proc\_obj (&r\_entry);} }}

int proc\_obj(struct q\_entry \*msg){

printf("\n Priority: %ld message %s\n", msg->mtype, msg->mtext);

}

# queue.h

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <string.h>

#include <errno.h>

#define Qkey ((key\_t)0105) //ipc key for queue

#define QPERM 0660 //access permission

#define MAXOBN 50 //maximum length for object

#define MAXPRIOR 10 //maximum priority value

struct q\_entry {//message structure for message queue

long mtype; //type of message

char mtext[MAXOBN+1]; //message

};

# inet.h

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <string.h>

#include <signal.h>

#include <fcntl.h>

#include <unistd.h>

#include <setjmp.h>

#define SERV\_TCP\_PORT 25000

#define SERV\_UDP\_PORT 35001

#define CLI\_UDP\_PORT 35002

# Server.c

#include "inet.h"

#include <signal.h>

#define BUFSIZE 1024

int main(int argc, char\*\* argv){

fd\_set master;

fd\_set read\_fds;

struct sockaddr\_in myaddr;

struct sockaddr\_in remoteaddr;

int fdmax;

int sockfd;

int new\_sockfd;

char buffer[BUFSIZE];

int nbytes;

int yes=1;

int addrlen;

int i,j;

sigset\_t set1;

sigemptyset(&set1);

sigaddset(&set1, SIGTSTP);

sigprocmask(SIG\_BLOCK, &set1, NULL);

FD\_ZERO(&master); //clear the thing in master

FD\_ZERO(&read\_fds);//clear the thing in read\_fds

if((sockfd = socket(AF\_INET, SOCK\_STREAM, 0))==-1) {

printf("\nsocket() error!!!\n");

exit(1);}

if(setsockopt(sockfd, SOL\_SOCKET, SO\_REUSEADDR, &yes, sizeof(int))==-1) {

printf("\nSetsockopt() error!!!\n");

exit(1);}

bzero((char \*)&myaddr, sizeof(myaddr));

myaddr.sin\_family = AF\_INET;

myaddr.sin\_addr.s\_addr = INADDR\_ANY;

myaddr.sin\_port = htons(SERV\_TCP\_PORT);

bzero(&(myaddr.sin\_zero), 8);

if( bind(sockfd, (struct sockaddr \*)&myaddr, sizeof(myaddr))==-1){

printf("\nbind() error!!!\n");

exit(1);}

if(listen(sockfd, 10) == -1){

printf("\nlisten() error!!!\n");

exit(1);}

FD\_SET(sockfd, &master); //add sockfd into master

fdmax = sockfd; //only sockfd at this moment

for(;;){

read\_fds = master;

if(pselect(fdmax+1, &read\_fds, NULL, NULL,NULL, &set1) == -1){ //to prevent the signal race condition

printf("select() error\n");

exit(1); }

for(i=0; i<=fdmax; i++) {

if( FD\_ISSET(i,&read\_fds)){

if(i == sockfd) {

addrlen = sizeof(remoteaddr);

if((new\_sockfd = accept(sockfd, &remoteaddr, &addrlen)) == -1) {

printf("\naccept() error!!!\n"); }

else {

FD\_SET(new\_sockfd, &master); //add new\_sockfd into master

if(new\_sockfd > fdmax) {

fdmax = new\_sockfd;}

printf("selectserver: new connection from %s on socket %d \n",

inet\_ntoa(remoteaddr.sin\_addr), new\_sockfd);

} //else for accept()

}//else for i == sockfd

else{ /\*if i != sockfd\*/

if((nbytes = recv(i, buffer, sizeof(buffer), 0))<=0){

if(nbytes == 0) {

printf("selectserver: socket %d hung up\n", i);}

else {

printf("\nrecv() error!!!\n"); }//else for nbytes ==0

close(i);

FD\_CLR(i, &master);//remove i from master

}//for recv

else {

printf("Message [%s] from soket [%d] client [%s] \n", buffer, i,inet\_ntoa(remoteaddr.sin\_addr));

/\*

for(j=0; j<= fdmax; j++){

if(FD\_ISSET(j, &master)) {//test to check whether j is in master or not

if(j==new\_sockfd) {

if(send(j, buffer, nbytes, 0)==-1){

perror("\nsend() error!!!\n");}

else

printf("\nSend[%s] via soket [%d]\n",buffer,j);

}// for j!=sockfd

}}}\*/}}}}}

return 0;

}