**Server.c**

#include <netinet/in.h>

#include <string.h>

#include <resolv.h>

#include <stdio.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <netdb.h>

#include <time.h>

void error(char \*msg)

{

perror(msg);

}

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

{

struct tm \*timecut;

int socketid, newsocketid, portno, i=0;

socklen\_t addrlength;

char output[256], temp;

struct sockaddr\_in serv\_addr, cli\_addr;

addrlength = sizeof(cli\_addr);

time\_t currtime;

char str[10000];

int n;

if (argc < 2) { /\*must have port specified\*/

fprintf(stderr,"You must have the port number as a command line arguement");

}

socketid = socket(AF\_INET, SOCK\_STREAM, 0); /\*creating socket with id type AF\_INET\*/

if (socketid < 0)

error("ERROR opening socket");

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

portno = atoi(argv[1]); /\*convert commandline port number to portno\*/

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

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

serv\_addr.sin\_port = htons(portno); /\*conver port number to network byte order that sin\_port can use\*/

if (bind(socketid, (struct sockaddr \*) &serv\_addr,

sizeof(serv\_addr)) < 0)

error("ERROR on binding");

bind(socketid, (struct sockaddr \*) &serv\_addr,

sizeof(serv\_addr));

listen(socketid,4); /\*waiting for client\*/

newsocketid = accept(socketid, (struct sockaddr \*) &cli\_addr, &addrlength); /\*accepts connection to client\*/

while(1)

{

if (newsocketid < 0)

error("ERROR on accept");

bzero(output,256); /\*nullify output before reading\*/

n = read(newsocketid,output,255); /\*read input into output array\*/

while(output[i]!='\0')

{

if(output[i]<='z' && output[i]>='a') /\*if lowercase then convert to upper\*/

{

temp=output[i];

output[i]=(temp-32);

}

else

{

output[i]=output[i];

}

i++;

}

output[i]='\0';

i=0;

write(newsocketid,output,100); /\*send back output to the client with "write"\*/

}

}

**Client.c**

#include <stdio.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <netdb.h>

#include <stdlib.h>

#include <string.h>

void error(char \*msg)

{

perror(msg);

exit(0);

}

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

{

int sockfd, portno, n;

struct sockaddr\_in serv\_addr;

struct hostent \*server;

char output[256]; /\*buffer i will write to\*/

if (argc < 3) {

fprintf(stderr,"usage %s hostname port\n", argv[0]);

exit(0);

}

portno = atoi(argv[2]); /\*assigns port arguement to portno\*/

sockfd = socket(AF\_INET, SOCK\_STREAM, 0); /\*create socket\*/

if (sockfd < 0)

error("ERROR opening socket");

server = gethostbyname(argv[1]); /\*serer=host\*/

if (server == NULL) {

fprintf(stderr,"ERROR, no such host\n");

exit(0);

}

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

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

bcopy((char \*)server->h\_addr,

(char \*)&serv\_addr.sin\_addr.s\_addr,

server->h\_length); /\*initializes host information\*/

serv\_addr.sin\_port = htons(portno); /\*convert port number into network byte type\*/

if (connect(sockfd,(struct sockaddr \*)&serv\_addr,sizeof(serv\_addr)) < 0) /\*connect/if connection to server fails\*/

error("ERROR connecting");

while(1) {

printf("Please enter the message: ");

bzero(output,256);

fgets(output,255,stdin); /\*takes input of client\*/

n = write(sockfd,output,strlen(output)); /\*sends input to server\*/

if (n < 0)

error("ERROR writing to socket");

bzero(output,256);

n = read(sockfd,output,255); /\*read server output into output variable\*/

if (n < 0)

error("ERROR reading from socket");

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

}

return 0;

}

**Output**

Please enter the message: Hello

HELLO

Please enter the message: am I in all CAPS?

AM I IN ALL CAPS?

Please enter the message: cool

COOL

Please enter the message:

**Program log**

Time to implement 3 hours

Time testing: 45 minutes

This program was to configure a server and client and make a successful connection. Upon the successful connection the server was to convert client input into all caps and send it back to the client. Luckily there was sample code of server/client code in C from the web. I still learned a lot from it such as the “create/bind/connect/listen”. I learned that server/client code is not magic and it is simple as finding a place for the two to communicate and that is a socket. One very important lesson I learned from this was not just about sockets, but on a Linux command. After finished with my client program, and having it run flawlessly, I decided to try “gcc –o” to name it something other than “a.out”. Unfortunately I had my arguments mixed so it overwrote my “client.c” file with “client”. I had to re-make the client program and learned the potential danger of the –o option and the importance of backing up.