Secure Internet Application Development

CPS 499-63/592-65, Summer 2017

## **LAB 2: A Simple TCP Client application in C**

**Code:**

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/socket.h>

#include<netdb.h>

#include<string.h>

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

{

char \*servername;

char \*path;

int port;

printf("Client program \n");

if(argc!=3)

{

printf("Usage : %s <server> <port> \n", argv[0]);

exit(0);

}

servername = argv[1];

path=argv[2];

port = atoi(argv[2]);

printf("Servername= %s , port=%d \n",servername,port);

int sockfd;

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

if(sockfd<0)

{

perror("ERROR opening socket");

}

struct hostent \*server\_he; //an host address entry

if ((server\_he = gethostbyname(servername)) == NULL)

{

perror( " error in gethostbyname");

return 2;

}

struct sockaddr\_in serveraddr; // store the server's address

*//prepare to copy: clear sockaddr\_in structure*

/\* (In network programming, we often need to initialize a field, copy the contents of one field to another \*/

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

*//set the family to AF\_INET (IPv4)*

serveraddr.sin\_family = AF\_INET;

*/\* copy the server’ address from gethostbyname(..), //return in struct hostent (stored in \*server\_he) \*/*

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

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

server\_he->h\_length);

serveraddr.sin\_port = htons(port);

if(connect(sockfd, (struct sockaddr \*) &serveraddr, sizeof(serveraddr)) < 0)

{

perror("Cannot connect to the server");

exit(0);

}

else

printf("Connected to the server");

char request[255];

sprintf(request,"GET /%s HTTP/1.0\r\nHost:%s\r\n\r\n",path,servername);

char \*msg = "GET / HTTP/1.0\r\n\r\n";

int bytes\_sent;

printf("message:%s",request);

bytes\_sent = send(sockfd, msg, strlen(msg) , 0);

char buffer[1024];

bzero(buffer,1024);

int byte\_received;

bzero(buffer , 1024);

byte\_received = recv(sockfd, buffer , 1024, 0);

if(byte\_received < 0)

error ("ERROR reading from socket");

printf("Message received : %s", buffer);

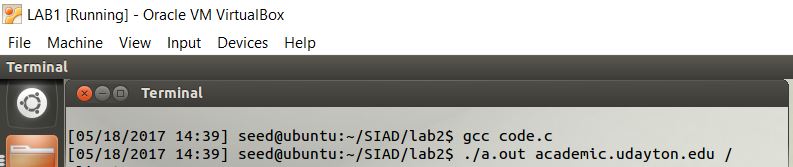
close(sockfd);

return 0;

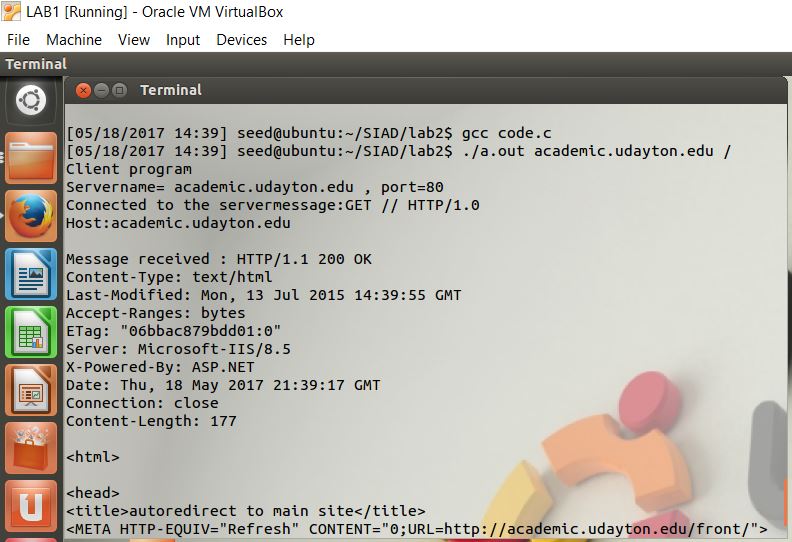
}

**Screenshots:**

**Input URL 1:**



**Output 1:**

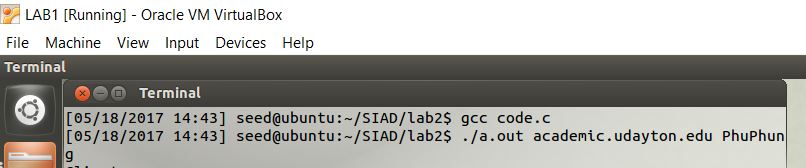


**Explanation:**

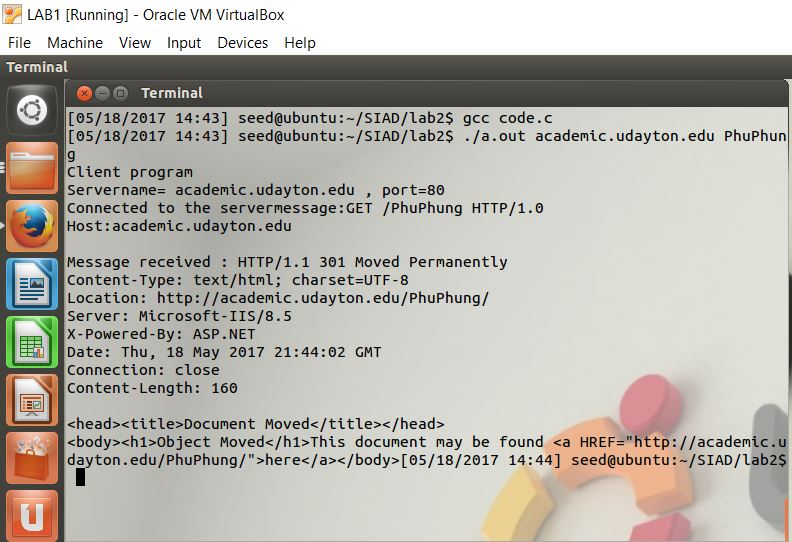
In this example, an URL is the input : the call will be sent to the server and a HTTP Response will be displayed.

HTTP/1.1 200 OK – Successful HTTP Response

**Input URL 2:**



**Output 2:**

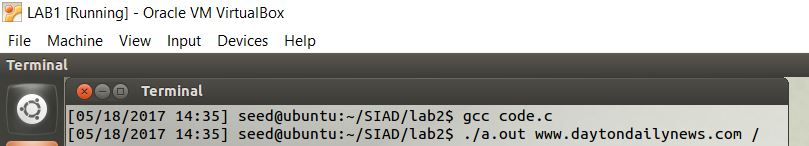


**Explanation:**

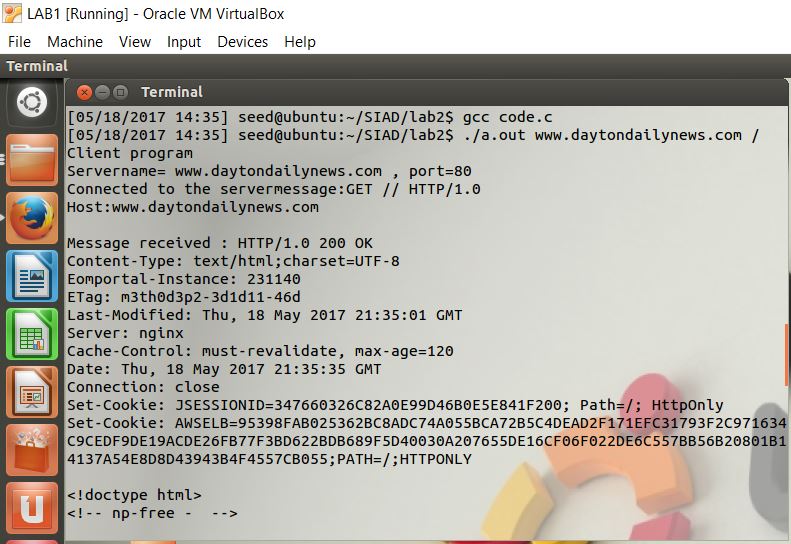
In this example, an URL is the input : the call will be sent to the server and a HTTP Response will be displayed.

HTTP/1.1 301 Moved Permanently – HTTP Response which indicates that site is moved to another location permanently

**Input URL 3:**



**Output 3:**



**Explanation:**

In this example, an URL is the input : the call will be sent to the server and a HTTP Response will be displayed.

HTTP/1.1 200 OK – Successful HTTP Response