PROJECT REPORT

TELNET SERVER

SUBMITTED IN PARTIAL FULFILMENT OF THE DEGREE OF

BACHELOR OF TECHNOLOGY

by

Jubin Jose
Y1030 S6 CSE

Under the guidance of

Mrs. Priya Chandran

2004

Department of Computer Engineering
National Institute of Technology, Calicut

National Institute of Technology, Calicut
Department of Computer Engineering

Certified that this Project entitled

TELNET SERVER

is a bonafide work carried out by

Jubin Jose
Y1030 S6 CSE

in partial fulfilment of his
Bachelor of Technology Degree
under our guidance

Mrs. Priya Chandran
Faculty
Dept. of Computer Engineering

Dr. V.K. Govindan
Professor and Head
Dept. of Computer Engineering

Acknowledgement

I thank my guide Mrs.Priya Chandran for her guidance and help for the successful completion of my project .

Jubin Jose
Y1030 S6 CSE

Abstract

Telnet Server is a program running on a host machine and allow remote clients to connect to the server.This server program checks the authenticity of the remote clients.Autherized clients can work on the remote machine .Server on the destination machine accepts the characters sent to it by the client, and passes them to a terminal server. A "terminal server" is just some facility provided by the operating system for entering keystrokes from a user's keyboard. The terminal server treats the remote user as it would any other user logged in to the system, including relaying commands to other applications. The terminal server passes outputs back to the TELNET server, which relays them to the client, which displays them on the user's screen. The connection is achieved using the TELNET protocol.

Contents

Chapter 1

Introduction

This Telnet Server program helps the user to login to a remote machine,if we run this program on that machine.The user at the remote client can use a standard Telnet Client program to communicate to the server.In general, a TELNET server is implemented as a master server with some number of slave servers. The master server listens for service requests from clients. When it hears one, it spawns a slave server to handle that specific request, while the master goes back to listening for more requests.

Chapter 2

The TELNET Protocol

The TELNET protocol is based on three ideas:
The Network Virtual Terminal (NVT) concept. An NVT is an imaginary device having a basic structure common to a wide range of real terminals. Each host maps its own terminal characteristics to those of an NVT, and assumes that every other host will do the same.
A symmetric view of terminals and processes .
Negotiation of terminal options. The principle of negotiated options is used by the TELNET protocol, because many hosts wish to provide additional services, beyond those available with the NVT. Various options may be negotiated. Server and client use a set of conventions to establish the operational characteristics of their TELNET connection via the "DO, DON'T, WILL, WON'T" mechanism discussed later in this document.

The two hosts begin by verifying their mutual understanding. Once this initial negotiation is complete, they are capable of working on the minimum level implemented by the NVT. After this minimum understanding is achieved, they can negotiate additional options to extend the capabilities of the NVT to reflect more accurately the capabilities of the real hardware in use. Because of the symmetric model used by TELNET, both the host and the client may propose additional options to be used. The set of options is not part of the TELNET protocol, so that new terminal features can be incorporated without changing the TELNET protocol (mouse?). All TELNET commands and data flow through the same TCP connection. Commands start with a special character called the Interpret as Command escape character (IAC). The IAC code is 255. If a 255 is sent as data - it must be followed by another 255 Each receiver must look at each byte that arrives and look for IAC. If IAC is found and the next byte is IAC - a single byte is presented to the application/terminal. If IAC is followed by any other code - the TELNET layer interprets this as a command.

Chapter 3

Design

First of all We grab a socket for the master server .Then bind the serer to that socket.The sever listens to that port .When a client program connects , accepts the connection and do the service according to whether the client is authorized or not.The server takes the commands fromthe client side and remove its protocol part.After this the server gives this commands to a terminal server .The terminal server evaluates the command and gives the output to a server.Server then send the results to the client.These are done by read , write function .We use the programming language C to implement our server .C provides various libraries to suffice our requirements

3.1 Grabbing the socket

We use the function int socket(int domain,int type, int protocol) to get the socket.This function returns a socket decriptor of the granted socket.

3.2 Binding to the socket

Then we bind our server to the grabbed socket using the function int bind(int sockfd, struct sockaddr *my addr, socklen t addrlen);

3.3 Listening to the socket

The server listens to the socket for connection fromthe client by using the function int listen(int socket, int backlog), where backlog is the max length of queue waiting for a connection.

3.4 Accepting the conecction

It extracts the first connection request on the queue of pending connections, creates a new connected socket with mostly the same properties as s, and allocates a new file descriptor for the socket, which is returned. The newly created socket is no longer in the listening state. The original socket s is unaffected by this call. int accept(int s, struct sockaddr *addr, socklen t *addrlen);

3.5 Sending and Recieving of commands and results

Sending and recieving is done by the system functions write,read respectively. ssizet read(int fd, void *buf, sizet count); ssizet write(int fd, const void *buf, sizet count);These function are used to read and write to file descrptors.Commands are written to the terminals and the result is read from it .Then write this to the correspoding socket.

Chapter 4

Authenticating the request

This deals with checking whether the client is authorized or not.This uses a loginpath like /user/login.We execute this program to verify the authenticaion.If the user is valid we will continue otherwise we reject and close the connecion.

Chapter 5

Handling multiple clients

To serve multiple clients we maintain a linked list of clients .The node is given below
struct tsession
struct tsession *next;
int sockfd, ptyfd;
int shell pid;
/* two buffers */
char *buf1, *buf2;
int rdidx1, wridx1, size1;
int rdidx2, wridx2, size2;
;

We have got a virtual terminal associated with a client and there is two buffers also.One buffer is to write to the terminal ,ie. the commands .The other is to take the result of command execution and write it to the socket . The server continuosly check the status of the file descriptors by using the function
int select(int n, fdset *readfds, fdset *writefds, fdset *exceptfds, struct timeval *timeout);
The functions select wait for a number of file descriptors to change status.If a change occures then server traverse the list and work out read , write operations from and to the buffers and the sockets.

Chapter 6

Appendix

6.1 Pseudo code

while(ifAuthenticated)
(accept new connection
check for the set of file descriptors to change the state.
service each client according to the change in its descrptors.
recieve the commands and give it to corresponding virtual terminal
take the output and send it to the corresponding socket.
)

6.2 Implementation details

1.Language used is C
2.Operating System: Linux
3.Root permission in the host is needed to check login.

6.3 Usage

At server
Usage: ./a.out -p port -l loginprogram -d (daemonize)
At client
telnet hostname port

6.4 Comments

Servers are usually implemented in C , I used C.This is because of efficiency constrains and C provides enough libraries.This forced me to do this in C.I wrote this to work in linux environment since linux is the most famous operating system for networking.

Bibliography

[1] Linux man pages

[2] Beej's Network Programming tutorials

[3] Thomas H. Corman, Charles E. Leiserson, Ronald L. Rivest, "Introduction to Algorithm", MIT Press, Cambridge, MA, USA, 1990

[4] Larry L . Peterson , Bruce S. Davie,"Computer Networks A systems approach"