Telnet Server Technical Reference

Interniche Legacy Document

Version 1.00

Date: 05-May-2017 16:54

Table of Contents

1 Overview

This Technical reference is provided with InterNiche's Telnet Server software . The purpose of this document is to provide enough information so that a moderately experienced "C" programmer with a reasonable understanding of TCP/IP protocols can port the InterNiche's Telnet Server software to a new environment.

If the Telnet code was delivered as part of an InterNiche TCP/IP stack, there is little or nothing to do - the Telnet layers were compiled, linked and tested with the IP stack. This manual is intended primarily as an aid to programmers porting InterNiche's Telnet Server software.

1.1 Terms and Conventions

In this document, the term "stack", when used without other qualification, means the TCP/IP and related code as ported to an embedded system. "System" refers to your embedded system. "Sockets" refers to the TCP API developed for UNIX at U.C. Berkeley. A "porting engineer" refers to the engineer who is porting the Telnet code. An "end user" refers to the person who ultimately ends up using the engineer's product. "FCS" is an acronym for "First Customer Ship", the point in the software development cycle when the product is declared ready to ship. A "packet" is a sequence of bytes sent on network hardware, also known as a "frame" or a datagram".

Names of files, C structures and C routines are displayed as follows: `c_routine()`.

Small samples of source code from C programs is displayed in these boxes:

```
/* C source file - yet another hello program. */
main()
{
   printf("hello world.\n");
}
```

1.2 What Telnet Does

Introduction

Telnet is a protocol based on a TCP (Transmission Control Protocol) connection. It is used to transmit data interspersed with Telnet control information. The original Telnet protocol specification can be found in RFC 854.

The purpose of the Telnet protocol is to provide a fairly general, bi-directional, eight-bit byte oriented communications facility. Its primary goal is to allow a standard method of interfacing terminal devices and terminal-oriented processes to each other. It is envisioned that the protocol may also be used for terminal-terminal communications ("linking") and process-process communication (distributed computing).

General Considerations

The Telnet protocol is built upon three main ideas: first, the concept of "Network Virtual Terminal"; second, the principle of negotiated options; and third, a symmetric view of terminals and processes.

When a Telnet connection is first established, each end is assumed to originate and terminate at a "Network Virtual Terminal", or NVT. An NVT is an imaginary device which provides a standard, network-wide, intermediate representation of a canonical terminal. This eliminates the need for "server" and "user" hosts to keep information about the characteristics of each other's terminals and terminal handling conventions.

The principle of negotiated options takes cognizance of the fact that many hosts will wish to provide additional services over and above those available within an NVT. Independent of, but structured within the Telnet protocol are various "options" that will be sanctioned and may be used to allow a user and server to agree to use a more elaborate set of conventions for their Telnet connection. The symmetry of the negotiation syntax can potentially lead to non-terminating acknowledgment loops. The Telnet standard has certain rules to avoid such loops.

2 Features and Options

Following is a description of all the #define options available for Telnet.

TELNET_SRV	Controls the inclusion of all Telnet sources in the build. It is defined in `ipport.h`
TEL_INICHE_IP	Use InterNiche's TCP/IP stack.
TEL_SHOW_MSG	Show description for error messages and Telnet negotiation options.
TEL_SESS_TIMEOUT	If defined the Telnet Server will close a Telnet session if there has not been any activity for TEL_IDLE_TIME seconds.
TEL_MENU	Enable the menu commands. The default implementation provides use of menu commands on InterNiche's command prompt. It is done in tel_menu_init(). If menus are to be used with some other architecture, then customization can be done in tel_menu_init().
TEL_USERAUTH	Support for user authentication (login, password).
TEL_SB_COM_PORT	Enbles recognition of the Telnet options defined in RFC2217 for supporting modems and serial ports.

2.1 Interface Provided by Telnet Server

When a Telnet session is established, the Telnet Server creates a GIO context through which the server can communicate with the Telnet client. The Telnet Server with NicheStack extends the console menu commands to Telnet. When a Telnet client connects to the Server, it will be connected to an interactive session through which the user can enter and execute commands.

To illustrate this, consider the following senario:

1. When a new Telnet session is opened, the Telnet Server sends a welcome message and a prompt.
2. It processes characters from the client until a complete command has been entered (that is, when the user presses the <Enter> key).
3. It then calls tel_exec_cmd() to execute the command. For InterNiche's TCP/IP, tel_exec_cmd() maps to cli_command().

At the heart of this mechanism is the GIO Context, which is described in the NicheStack Technical Reference Manual. One such structure is used for each Telnet session. Here is how the information flows:

1. Telnet Client completes typing a command.
2. Telnet Server, when processing the corresponding session, finds that a complete command has been entered. So it calls cli_command() with a pointer to the session's CLI context which in turn includes the GIO Context.
3. cli_command() calls the appropriate function (example: tel_show_stats) with the GIO pointer
4. Using the GIO structure, tel_show_stats() sends all its output to the Telnet Client.

To provide an alternate menu interface via the Telnet Server, the following changes would be required:

1. tel_exec_cmd() should be mapped to a function which processes menu commands.
2. All of the functions hooked to a menu should use the GIO structure in combination with `gio_printf()` to output to the Telnet Client.

2.2 Setting Telnet Options

When a new Telnet connection is established, the Telnet Server negotiates for certain options with the Telnet Client. InterNiche's Telnet Server supports a few of them like echo and suppress Go Ahead. To support a new option, the following needs to be done.

1. For example if you want to enter support for status option: Add an extra member to the structure TelOptionList (end of list). Namely struct TelnetOption status.
2. Add the default values for this option to global array tel_opt_list.

The following things will now happen automatically.

1. When a new Telnet session is established, it starts with default values of all the options. It negotiates the options and the values are set appropriately.
2. If during a Telnet session, the Telnet Client renegotiates an option, then the values of that option are properly updated.

Using the above, you can implement the processing related to this option.

2.3 Telnet Sub-Negotiation (Expanded Negotiation)

If some particular option requires a richer negotiation structure, it can perform sub-negotiation. InterNiche's Telnet Server provides an entry point for such needs. So if sub-negotiation is desired for an option, then the changes can be done in tel_proc_subcmd(). The comport options, enabled via `TEL_SB_COM_PORT`, are an example of how to do sub-negotiation processing.

3 Telnet User Authentication

InterNiche's Telnet Server provides the option of user authentication. It works as follows:

1. A call is made to `change_usrtab()` to add a Telnet username and password entry to the user table.
2. When a new Telnet session is initiated, the user is asked for login and password.
3. The `TEL_CHECK_PERMIT()` function is called to verify them. If they are correct, a normal Telnet session is started. If they are incorrect, the user is asked to re-enter the login and password strings.
4. Telnet Server allows `TEL_MAX_LOGIN_TRIES` (5 by default) tries. After `TEL_MAX_LOGIN_TRIES`, the Telnet connection is closed.

In NicheStack, `TEL_CHECK_PERMIT` maps to the `check_permit()` function. The `change_usrtab()` and `check_permit()` functions are implemented in `misclib/userpass.c`.

Alternatively, if `INCLUDE_CLI` is defined, entries can be added to the user table by an init-time script that contains a "user" command. For example:

```
user -a -u guest -p guest -m telnet
```

4 Telnet User Menu

The Telnet Server comes with portable C code to implement a few simple diagnostic commands on a command line interface. The commands can be invaluable both during debugging of the server and to the end user during configuration and runtime. If you do not implement these menu commands as provided, we strongly suggest that some alternative method (i.e. a GUI) be provided to the end user for accessing the same data.

The menu commands are summarized below:

exit	Logout from the Telnet session
logout	Logout from the Telnet session
netstat	Shows OPTION values and/or statistics for all Telnet sessions

4.1 telnet exit

Command exit

`exit - Terminate a Telnet session`

Syntax

`exit`

Parameters

none

Description

This command terminates a Telnet session.

Notes/Status

- The `exit` and `logout` commands are functionally equivalent.
- This command is only valid within a Telnet session.

Location

This command is provided by the `Telnet` module when `TELNET_SVR` is defined.

4.2 telnet logout

Command logout

`logout - Terminate a Telnet session`

Syntax

`logout`

Parameters

none

Description

This command terminates a Telnet session.

Notes/Status

- The `exit` and `logout` commands are functionally equivalent.
- This command is only valid within a Telnet session.

Location

This command is provided by the `Telnet` module when `TELNET_SVR` is defined.

4.3 telnet netstat

Command netstats

`netstat - Display Telnet options and/or statistics`

Syntax

`netstat [-o] [-s]`

Parameters

`-o`	Display the option settings for each Telnet session
`-s`	Display the session statistics for each Telnet session

Description

This command displays information about each open Telnet session. If `'-o'` is specified, the state of the negotiated options are displayed. If `'-s'` is specified, session statistics are displayed. Specifying no parameters is equivalent to `'-o -s'`.

Notes/Status

- If no parameters are specified, both options and statistics are displayed.

Location

This command is provided by the `Telnet` module when `TELNET_SVR` is defined.