FTP Server Technical Reference

Interniche Legacy Document

Version 1.00

Date: 17-May-2017 16:05

Table of Contents

1 Introduction

This technical reference is provided with the InterNiche portable FTP server. The purpose of this document is to describe how to build and exercise the FTP module within NicheStack.

1.1 Terms and Conventions

In this document, the term "stack", when used without other qualification, means the InterNiche 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 "user" or "porting engineer" usually refers to the engineer who is porting the server. An "end user" refers to the person who ultimately ends up using the "user'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 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()`

Samples of source code from C programs are displayed in these boxes:

```
main()
{
   printf("hello world.\n");
}
```

2 What an FTP Server Does

FTP stands for File Transfer Protocol. It is the basic mechanism for moving files between machines over TCP/IP based networks such as the Internet. FTP is a "client/server" protocol, meaning that one machine, the client, initiates a file transfer by contacting another machine, the server and making requests. The server must be operating before the client initiates his requests. Generally a client communicates with one server at a time, while most servers (including the InterNiche server) are designed to work with multiple simultaneous clients.

FTP is formally defined by the IETF document RFC959. This software is compliant with that specification, and the RFC should be consulted for any detailed questions about the protocol itself. However a brief overview of the protocol is presented here.

When an FTP client contacts a server, a TCP connection is established between the two machines. The server does a passive open (a Sockets listen) when it begins operation; thereafter clients can connect with the server via active opens. This TCP connection persists for as long as the client maintains a session with the server, (usually determined by a human user) and is used to convey commands from the client to the server, and server replies back to the client. This connection is referred to as the FTP command connection.

The FTP commands from the client to the server consist of short sets of ASCII characters, followed by optional command parameters. For example, the FTP command to display the current working directory is "XPWD". All commands are terminated by a carriage return-linefeed sequence (CRLF) (ASCII 10,13; or Ctrl-J, Ctrl-M). The servers replies consist of a 3 digit code (in ASCII) followed by some explanatory text. Generally codes in the 200s are success and 500s are failures. See the RFC for a complete guide to reply codes. Most FTP clients support a verbose mode which will allow the user to see these codes as commands progress.

If the FTP command requires the server to move a large piece of data (like a file), the server opens a second TCP connection to do this. This is referred to as the FTP data connection (as opposed to the aforementioned command connection). The data connection is opened, usually by the server back to a listening client, only for transporting the required data; and is closed as soon as all the data has been sent.

3 Master FTP Server Port File: ftpport.h

Before you compile these files you should create a version of the file ftpport.h. This file contains most of the port dependent definitions in the stack. CPU architectures (big vs. little endian), compiler idiosyncrasies, and optional features (e.g. support for disk drive letters) are controlled in this file. A single mistake in this file (such as getting big and little endian confused) will guarantee that your port won't work properly. Taking a few hours up front to implement the file line by line is time well spent. This section outlines the basic contents of ftpport.h.

4 Testing

Once your `ftpport.h` file is set up and your glue layers are coded, compiled, and linked, you are ready to test your FTP Server. Before you start, you'll need to have a valid user and password and access to at least part of the target system's file system.

Perhaps the most common test of our FTP Server is from a Windows or Linux FTP client. Just open a shell or command box on a system on the same LAN, and type:

```
ftp X.X.X.X<CR>
```

where X.X.X.X is the IP address of the FTP Server. The client will establish a TCP connection to the FTP command port (port 21) and should then ask for your user name and password, which you must type in at the keyboard. From there on, the FTP Server will respond to all the FTP commands supported by the client.

After basic connectivity is verified, you will probably want to test file transfer. To do this, we recommend you begin with binary mode, transferring a file to the target device ("PUT") and then back ("GET") making sure to store the file under a different filename. Once the file has been retrieved, perform a binary file compare between it and the original file.

Another feature of the FTP Client we recommend using during testing is hash. Enabling this causes the client to print a stream of hash marks (…#########…) to the screen as the file is transferred. The marks should appear smoothly and continuously. If they appear in bursts, with pauses in between, this means you are getting less than ideal performance. Often this is the result of packet loss and/or resource problems.

5 FTP Client Commands

The FTP module provides several CLI utilities to assist in its use as a client. Inclusion of these in your target application is dependent on whether `INCLUDE_CLI` is defined in `ipport.h`.

5.1 FTP client commands - Collection of FTP client menus

Command Name

`FTP client commands - Collection of FTP client menus`

Description

For reasons explained in the **Notes** section below, FTP client commands exist as individual menus, rather than members of an overall FTP client menu. For this reason, it is often necessary to type FTP in front of the command name in order to disambiguate it from another command. For example, you must type "FTP put", rather than simply "put".

Syntax

open	open an FTP connection to an FTP server
`open -a <ipaddr> -u <username> [-p <password>]`	

ascii	Use ASCII transfer mode
`ascii - No parameters`	

binary	Use binary transfer mode
`binary - No parameters`	

cd	Change directory on the server
`cd -d <directory path>`	

get	GET a file (transfer file from server directory)
`get -f <file name> [-d <destination name>]`	

hash	Enable/disable hash mark printing
`hash -d | -e`	

ls	List files in the server directory
`ls - No parameters`	

pasv	Set FTP server to passive mode
`pasv - No parameters`	

put	Put a file (transfer file to server directory)
`put -f <file name> [-d <destination name>]`	

pwd	Print the working directory
`pwd - No parameters`	

quit	Close the FTP session and connection
`quit - No parameters`	

verbose	Enable/disable verbose mode
`verbose -d | -e`	

state	Display FTP client state
`state - No parameters`	

log	Enable/disable logging
`log -d | -e`	

Parameters

-a	Argument of type IPADDR specifying the IP address of the server. IP4 addresses use dotted notation. IP6 addresses use colon notation
-d	hash, verbose, and log commands: disable - no argument get and put commands: name to be given to the file on the distination server - Argument of type STRING
-e	hash, verbose, and log commands: enable - no argument
-f	get and put commands - Argument of type STRING specifing an abosolute path to the file
-p	Argument of type STRING specifying a password
-u	Argument of type STRING specifying a username

Notes/Status

- The NicheStack CLI interface requires all command options to use the format -X <XXX>. Because each FTP client command is identified by a multi-character name, each requires a separate menu.
- All FTP client commands except "open" require an existing FTP connection to the FTP server.
- The log command is available only if FC_LOG is defined.

Location

These commands are provided by the FTP module module when FTP_CLIENT is defined.