LIBFTP User's guide

Oleg Orel

October 28, 1993

INTRODUCTION

The basic orientation of this library is making user's programs which transport files via TCP/IP network. It contains set of functions, starting from primitive, such as opening FTP connection to the server, and finishing by high-level functions, such as functions which retrieve files via network, making and closing channels to the server. All functions have prototypes in common header file named **FtpLibrary.h**, which must be available in standard headers directory. Those prototypes almost fully describe orientation and arguments of all functions, but common ideology and library components should be mentioned.

This library is a client and uses standard FTPD from the other side.

There are problems of errors processing in UNIX including input/output errors. The mutual mechanism of value returning of all functions is used in this library. (EXIT macros, defined in file FtpLibrary.h). This mechanism allows, after the definition of the error processing functions, write programs, considering the conditions to be ideal. Data transfer functions have possibility to preset data stream expectation timeout. When the set time expires, previously set function will be called.

The first function, which should be called for work with library is FtpConnect or FtpLogin. They make connection to FTP server and return pointer to FTP date structure.

1 The FTP data structure

int **sock**	— descriptor of a command channel to the server;
FILE ***data**	— pointer to data structure, which describes data channel to the server;
int **errno**	— last returned value. When value is lower than 1, an error occurred;
char **mode**	— type of transfer;
int ch	— help variable. Is used to convert ASCII files;

STATUS (*func)()	— pointer to an error handler. It is called when status from the server is bad;
STATUS (*debug)()	— pointer to a debug handler. Is called from functions of sending/receiving messages to/from server;
STATUS (*IO)()	— pointer to Input/Output error handler. Is called when channel to server is broken.

2 Connection/Disconnection with server

STATUS FtpConnect(FTP **, char *hostname [1])
Makes channel to the server, at the "hostname" machine. Creates FTP data structure and returns pointer to it. If the procedure **FtplibDebug**(1) was previously called, **FtpConnect** calls automatically **FtpDebug** for the **debug mode** to be turned on. (Chapter 3, page 3).

STATUS FtpUser(FTP *, char *user)
Sends the name of the user to the server. The connection must be done before it.

STATUS FtpPassword(FTP *, char *password)
Sends **password** to the server. The function **FtpUser** must be called before it.

STATUS FtpAccount(FTP *, char *acct)
Sends a name of the account to the server. The name of the account is not standard attribute for many systems, so this function is used very seldom. The function **FtpPassword** must be called before it.

STATUS FtpLogin(FTP **, char *hostname, char *user, char *password, char *account)
Executes functions **FtpConnect**, **FtpUser**, **FtpPassword**, **FtpAccount** (if necessary) consistently. If the name of the account is absent, replaces it with the **NULL** value.

STATUS FtpBye(FTP *)
Finishes work with the server and closes all channels. [2]

[1] The name of the host may be symbolic (for example **dxcern.cern.ch**) or numeric (for example **128.141.201.96**)

[2] You can see from the description of connect/disconnect functions, that you can create more than one connection to servers simultaneously.

3 The debugging

There is a possibility to predefine three functions, such as: [3]

FtpSetDebugHandler(FTP *,function)
Predefines function of protocol debugging. After the function is predefined, it is called with every sending/receiving messages from the server. The function, defined as a debug handler must do returns to the calling functions (**FtpSendMessage**/**FtpGetMessage**), but can also abort the program.

FtpSetErrorHandler(FTP *,function)
Predefines error handler. If the server's answer means, that the operation is not finished correctly, this function will be called. The result code is negative, if an error is occurs.

FtpSetIOHandler(FTP *,function)
Predefines handler of Input/Output processing. This function is called, when a connection to the server is broken. For example, when the network or the remote host is down. This handler also is called after the **timeout** of one character waiting expires.

FtpDebug(FTP *)
Turns on all standard debugging functions.

FtpDebugError	— prints a string, taken from the server, and aborts the program;
FtpDebugDebug	— prints a string, taken from the server;
FtpDebugIO	— prints string **strerror(errno)** and aborts the program.

All function for debugging have three arguments:
1. Pointer to FTP data structure;
2. Last returned value from the server. When errors occur, the value is less than 1;
3. Diagnostic string.

FtplibDebug(1 or 0)
Turns on/off autostart debug mode, when connection is established.

[3] If the **NULL** value is transferred as a parameter **"function"** to the functions, described below, the debugging will be turned off.

4 Data transfer procedures from the server

STATUS FtpRetrTimeout(FTP *, char *command, char *inp, char *out [4], long time)
Sends a command to the server, if command contains substring %s it will be replaced by string **inp**. Creates data transfer channel, and copying data from this channel to a local file **out**. If during time period **"time"** no characters are obtained from the server, this connection will be closed, and Input/Output error status will be returned. When **timeout**=0, timeout in library level will be turned off, but procedures may be aborted by the kernel of TCP/IP, when the kernel's **timeout** expires. [5]

FtpRetr(FTP *, char *command, char *inp, char *out)
Calls **FtpRetrTimeout**, with turned off **timeout**.

FtpGetTimeout(FTP *, char *inp, char *out, long time)
Transfers file **inp** from the server to the local file **out**, with **timeout=time**.

FtpGet(FTP *, char *in, char *out)
Calls **FtpGetTimeout**, with turned off **timeout**.

FtpDirectory(FTP *, char *pat[6], char *out)
Transfers files listing from the server, described by **pat**, to the local file **out**.

FtpDir(FTP *, char *out)
Transfers files listing of the current directory from the server to the local file **out**.

5 Data transfer procedures to the server

FtpStorTimeout(FTP *, char *command, char *inp, char *out, long time)
Sends body of the local file **inp** to the server, and stores it in the file **out**. The "time", is maximum time needed to transfer one character to the server.

FtpStor(FTP *, char *command, char *inp, char* out)

[4] When the name of the local file is ***STDIN***, ***STDOUT***, ***STDERR***, then the current stream is redirected to these channels

[5] In different kernels timeout is different

[6] This is the first argument for **"ls"** command

Calls **FtpStorTimeout**, without **timeout**.

FtpPutTimeout(FTP *, char *in, char *out, long time)
Transfers data from the local file **inp** to the server and stores it in the file **out**.

FtpPut(FTP *, char *in, char *out)
Calls FtpPutTimeout with turned off timeout.

6 Server's files read/write procedures

This library contains special functions for remote files reading and writing, without precopying them to local files. The functions, which are described below, do it. After the data channel to a remote file is created, it becomes possible to read and write characters using standard Input/Output functions, or using special functions **FtpRead**/**FtpWrite**, which reorganize stream for standard text file, under condition that the **ASCII** mode is set. [7]

FtpData(FTP *, char *command, char *param, char *mode)
Makes data transfer channel, with presending command composed from **command** and **param**. The mode must be **"r"** or **"w"**

FtpOpenRead(FTP *,char *filename)
Opens file named **filename** for reading on server

FtpOpenWrite(FTP *,char *filename)
Creats and opens file named **filename** for writing on server

FtpOpenAppend(FTP *,char *filename)
Creats and opens file named **filename** for appending on server

FtpOpenDir(FTP *, char *files)
Creats channel for directory list reading, described by argument **files**.

int FtpRead(FTP *)
Reads character from data stream. If **ASCII** mode is set[8] converts new line markers. When the end of file is detected or channel is broken, returns **EOF**

FtpGetString(FTP *, char *str)
Reads one string from data stream using **FtpRead**

[7] Of course, such functions as **seek**, **ioctl**, ... can not be used.
[8] By default

FtpWrite(FTP *, char c)
Writes single character to stream, if **ASCII** mode is set converts new line markers. When channel is broken, returns **EOF**

FtpClose(FTP *)
Closes opened channel to server

7 Other commands for server

FtpCommand(FTP *, char *command, char *param, int ok1, ok2, ok3, ..., okN, EOF)
Sends a command, composed from **command** and **param** using **sprintf** function. Reads an answer from the server. When return code from the server is not included to **ok-list**(**ok1**,**ok2**...) the sign of code will be inverted.

FtpType(FTP *,char *mode)
Sets transfer mode, such as **"A"**,**"I"**,**"S"**

FtpBinary(FTP *)
Sets binary mode

FtpAscii(FTP *)
Sets **ASCII** mode

FtpMkdir(FTP *,char *dirname)
Makes directory on server

FtpChdir(FTP *,char *dirname)
Changes working directory on server

FtpRm(FTP *,char *filename)
Removes file on server

char *FtpPwd(FTP *)
Returns the name of working directory on server

FtpMove(FTP *,char *oldfilename, char *newfilename)
Renames file from **oldfilename** to **newfilename**

FtpGetFile(FTP *,char *filename)
Sends start transfer file command to server. Does not make data channel

FtpPutFile(FTP *,char *filename)

Sends start transfer file command to the server. Does not make data channel

FtpPort(FTP *, int a, int b, int c, int d, int e, int f)
A command for the server for making a new data channel. **a.b.c.d** is an IP address of a client(i.e. your IP address), **e*256+f** is a port number

8 Subprograms for sending/receiving control messages to/from server

FtpSendMessage(FTP *, char *message)
Sends a message to the server

int FtpGetMessage(FTP *)
Receives a message from the server.

FtpMessage(int Number)
Gets a message by code.

9 High-level functions

FILE *FtpFullOpen(char *filename,char *mode)
Parses string **filename**, which must contain a string in format or **host/user/password:filename** or **filename**, what corresponds to remote or local file. The second argument is the type of opening, divided into two characters: first — the mode of opening **"r"**, **"w"** or **"a"**, second is the transfer type , if contains character **"b"**, then the mode is binary.

STATUS FtpFullClose(FILE *stream)
Closes an opened file

Index

Contents