gFTP and More

By Jerry Peek

Linux often gives you lots of ways to do the same thing and file transfer is no exception. We covered file transfer in three columns from March to May 2003 (available online at http://www.linux-mag.com/depts/power.html). This month let's look at yet another variation using *gFTP* and some not-so-obvious details that can help you navigate all of the file transfer protocols.

gFTP: More Than Meets the Eye

gFTP (http://www.gftp.org/) is a graphical file transfer client that runs under the *X Window System*. Its name might make you think that it only handles the *File Transfer Protocol* (FTP), but gFTP does more. It can get files from a web server with *Hypertext Transfer Protocol* (HTTP) and *Secure HTTP* (HTTPS), and also does *Foreign Exchange Protocol* (FXP) file transfers between two remote FTP servers. gFTP even handles *SFTP*, a secure, FTP-like protocol that actually uses *SSH* to transfer files.

One gFTP interface is a command line, which is something like the standard FTP client that comes with virtually every operating system. The other interface is a flexible, two-window GUI with nice touches, such as bookmarks and a "Compare Windows" tool to show what's different between the two ends of the connection.

There's not a lot of gFTP documentation on the web. If gFTP is installed on your system, look for a complete user guide for the GUI interface (with just a few paragraphs about the command-line interface) in a file such as */usr/share/doc/gftp-common/USERS-GUIDE.gz*. You might want to keep that document handy as you read this. Here, we'll touch on some lesser-known features, like the FTP `SITE` commands, that also work in other file transfer programs.

One quick disclaimer before we dig in: gFTP changes fairly often and the version that you have installed may have bugs. It's a good idea to read the gFTP user group forums (linked from the home page) for information on new releases and outstanding issues.

The Command-Line Interface

The gFTP command-line interface is fairly limited. You'll get it automatically if you run *gftp* in a terminal that's not under the X Window System, like a Linux virtual console. To get the command-line interface in an X terminal window, run the command *gftp-text*.

The standard FTP utility is more powerful in most ways than gFTP's, with a few exceptions:

- *gftp-text* has a nice multicolor display that can help you sort out the verbose text you see while doing FTP transfers.
- Another difference is the "local" commands: use *lls* to list the local directory; *lmkdir* to make a local directory; and so on. (In the standard FTP client, you use the shell-escape operator, `!`, to run a command on the local system, such as `!ls` or `!mkdir`. *gftp-text* can't do a shell escape.)
- You can download directories *recursively* (a directory and all of its files), or just a single file, by specifying a URL when you start *gftp-text*, like this:

```
$ gftp-text -d ftp://ftp.abc.xyz/somedir/
```

The `-d` option downloads the specified URL, and since the URL is a directory, its entire contents are fetched.

In general, though, the *wget* utility (described in the April 2003 "Power Tools" column, available online at http://www.linux-mag.com/2003-04/power_02.html) is better for non-interactive transfers.

From here on, we'll cover gFTP's graphical interface.

The GUI, Overall

The gFTP graphical interface has five parts (you can see it by looking ahead to *Figure One*):

1. A menu bar. (The first menu, "FTP," is actually for all protocols — not just FTP).
2. An area to control the current connection: the remote hostname, port, and so on. You can type values in here or choose them from the dropdowns, although bookmarks (which we'll cover later) are much handier.
3. Two windows for the two sides of the connection: one for the local host (except during an FXP transfer) and the other for the remote host. Click an arrow between the windows to transfer selected items from one host to the other.

You'll find more operations on the menu bar. One of them is "Tools, Compare Windows," which highlights any filename that's in only one of the windows. (See *Figure One*.) NEED ANOTHER LINE HERE

4. A progress window that shows the queue of files to trans-

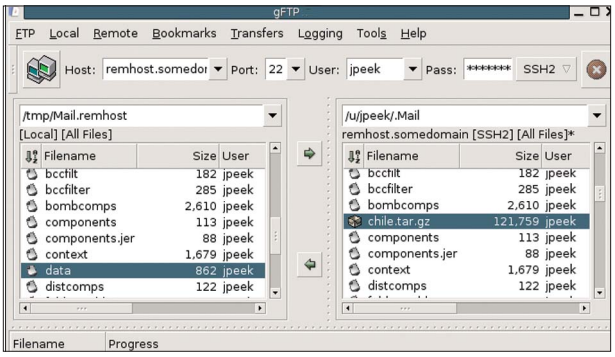

FIGURE ONE: *gFTP's* Compare Windows function

fer and what's happening during a file transfer. You can control transfers by clicking here and using menu entries.

5. A scrollable log window with verbose details.

In this article, we'll be transferring files between a local host and a remote host.

Protocol Overview

gFTP handles several file transfer protocols. Let's look at each of them.

Every protocol initiates a connection to a particular well-known port number on a remote host. FTP uses port 21, SSH (and SFTP) port 22, and HTTP port 80. A server can run on other ports, though, and gFTP lets you choose a different port other than the default. If you leave the port number blank, gFTP uses the default for that protocol.

FTP is an old standard that's been around for decades. There have been some recent changes, like a passive transfer mode, but the basic idea is the same: send commands and transfer data "in the clear" with no encryption. Transferring data to a remote server means logging in with a username and password. FTP is simple and effective, but anyone sniffing packets from the network can snoop.

The most popular use for FTP is probably *anonymous FTP*, where you log in as the user *anonymous* with any password (it's courteous to use your email address so the remote host knows who you are, but this isn't always required). Some FTP servers accept uploaded files — typically only to a directory named *incoming*, where files can be written but often can't be read (except by a local user on that system).

FTP generally supports two *transfer types*: *ASCII* and *binary*. The ASCII type translates plain text between the different line-end conventions on different operating systems. For instance, the lines of *DOS* text files end with a carriage return character followed by a linefeed character, but the lines of Linux text files end with only a linefeed. Binary mode sends a file as-is, with no translation.

Binary mode is what you usually want, and it's almost always the right choice between two hosts running the same operating system. (In gFTP, the transfer type is easiest to set on the "FTP" tab of each bookmark where it applies. If you transfer both text and binary files, you may want two bookmarks per host: one that sets binary type and another for ASCII.)

HTTP is typically used to send web pages to browsers, but it also can transfer arbitrary data like programs and pictures. There's no username or password needed, and data is sent in the clear. On the other hand, HTTPS encrypts data. A web server may support one or both of those protocols.

SFTP uses the SSH protocol to effectuate secure, encrypted transfers. It's not supported by all SSH servers, because the server's administrator has to enable it. You'll need a username and password on the remote host. The setup can vary from server to server, which makes configuration a little tricky. This is where gFTP's bookmarks come in handy.

gFTP Bookmarks

As you've seen, there are lots of ways to transfer files and lots of details to keep track of. gFTP's bookmarks are a handy way to manage all of those details.

The "Bookmarks" menu has a tree of folders and bookmark entries. If you choose "Edit bookmarks," a simple window opens up. You can drag entries around, and you can also click on an entry and choose "File, Properties" to get a properties dialog. The dialog has a series of tabs for the main protocols and also for the overall transfer — things like which directory to open on the remote and local sides of the connection.

Although gFTP has overall option settings (in "FTP, Options"), and you can also specify some settings from near the top of the main window, you'll probably find that most settings are easiest to make in individual bookmarks.

Bookmarks are kept in a file named, oddly enough, *bookmarks*. It's in your *~/.gftp* subdirectory (the name starts with a dot to "hide" it). *Listing One* shows two of the bookmarks.

Each bookmark entry starts with the folder name and subentry name in square brackets. (These two bookmarks are from the "General Sites" folder.) The second bookmark, which is for the SFTP protocol, has an `ssh2_sftp_path` entry. This comes from the `SSH2 sftp-server path` box on the "SSH" bookmark tab, and we'll discuss it later.

Notice the FTP password `@EMAIL@`. This actually stands for the email address set in the "Email address" field on the "FTP" options tab. The password is shown on the "General" tab in the "Password" field as a row of asterisks (`*******`). This setup is done automatically if you check the box "Log in as ANONYMOUS" on a bookmark's FTP options tab, then save the settings and return later.

You can store plain-text passwords (not just your email

address) in the *bookmarks* file, but this is insecure. gFTP has an option "Store passwords in scrambled form" on the "FTP, Options, General" tab, but this still isn't very secure. If you leave the password field blank, gFTP prompts for your password as you connect. This is the most secure choice.

You can use a text editor like *vi* or *emacs* to quickly make global bookmark changes. For example, you can change the `local directory` setting in many of the entries. Be careful not to corrupt the file — by leaving extra spaces, for instance.

Most of gFTP's other features are described in its user guide, and you can read that later. Let's see some features of file transfers that aren't obvious.

Saving Download Time

If you have a fast network connection, uncheck the gFTP option "Do one transfer at a time." If you've selected more than one file to download (possibly from several sites, as described in the user guide), gFTP starts multiple simultaneous downloads.

FTP SITE Commands

When you use the command-line FTP command `put` or you double-click on a file in the gFTP window, the FTP client actually sends a standard FTP command like `STOR` to the remote server. (There may be a list of these protocol commands in your *ftpd man* page — try typing `man in.ftpd`.)

The remote FTP server may support some extra, site-dependent commands. These commands have at least two words: the first word is always `SITE`. You can type these commands from a command-line FTP client. For instance, type `site help` to get a listing of the SITE commands that the remote server supports. (A common answer is `500 Permission denied`, which means you can't use SITE commands on that server.)

In gFTP, choose "Send SITE Command..." from the "Remote" menu; you can type everything that goes after SITE. For example, *Listing Two* shows the gFTP log window while using "Send SITE Command..." to send `help`, `help checkmethod`, and `checkmethod`. We've shown the commands that gFTP sends in boldface (on your screen, the same text is green).

Apparently, at least on this site, you can ask for *MD5* checksums of files you retrieve to be sure you've gotten them correctly. (This might be especially handy if a transfer is interrupted and you have to use gFTP's resume transfer feature, appending the new data to the existing partial file.)

SFTP Setup

As we said, the SFTP protocol actually starts the *ssh* program to connect to the remote host, which then needs to start the program called *sftp-server*. *ssh* can start any remote program,

LISTING ONE: Two gFTP bookmarks stored in *~/.gftp/bookmarks*

```
[General Sites/Mozilla]
hostname=ftp.mozilla.org
port=21
protocol=FTP
remote directory=/pub
local directory=
username=anonymous
password=@EMAIL@
account=

[General Sites/remhost]
hostname=remhost.somedomain
port=0
protocol=SSH2
remote directory=/u/jpeek
local directory=
username=jpeek
password=
account=
ssh2_sftp_path=/usr/libexec
```

LISTING TWO: Sample FTP SITE commands

```
SITE help
214-The following SITE commands are recognized
  (* =>'s unimplemented).
    EXEC              CHECKMETHOD
    CHMOD             NEWER
    ALIAS             CHECKSUM
    HELP              MINFO
214 Direct comments to ftpadm@abc.xyz.

SITE help checkmethod
214 Syntax: SITE CHECKMETHOD [ <sp> method ]

SITE checkmethod
200 Current checksum method: MD5 (RFC1321)
```

but there's no required location for *sftp-server* in a filesystem. There are at least two ways to handle this problem with gFTP.

If the remote server's SSH configuration file (like *sshd_config*) includes a `subsystem` line that defines *sftp-server*, then gFTP can simply use the command `ssh -s sftp` to connect to the remote server, and "it should work." Enable this by clicking "Use SSH2 SFTP subsys" on the `SSH` options tab (typically in a host's bookmark entry).

If a remote server's SFTP subsystem isn't set up, you can also give the absolute pathname of the *sftp-server* program on the remote host. Put this in the "SSH2 sftp-server path" box under the SSH options tab, and also *un*-check "Use SSH2 SFTP subsys."

For instance, if the remote server is in the file */usr/libex-*

ec/sftp-server, then put `/usr/libexec` (with no trailing slash!) in the "SSH2 sftp-server path" box.

The gFTP log messages can show you what's going on. If you need more help, try adding `-v` (for "verbose") in the box "SSH Extra Params" under the SSH options tab. (Note that, on at least one version of gFTP, you need to include a space before the `-v`!)

Tracking HTTP Troubles

Getting the contents of remote directories by HTTP can be confusing with gFTP. One thing to remember about HTTP transfers is how web servers typically work: if a remote directory has an index file (like *index.html*, *home.htm*, or others) the HTTP server won't send a listing of the remote directory. Instead, it will send the contents of the index file inself, which is typically HTML text.

When this happens in gFTP, the remote directory window will stay blank — and, instead, you'll see the HTML page (or a server error message) in the gFTP log window.

gFTP can be very handy for HTTP and (especially) secure HTTPS transfers from a remote server that doesn't have index files in the data directories you want to download. But, on other servers, you may want to use *wget* instead. It can parse HTML index pages to try to find what's in the remote directories.

POWER TIP: Set Passive FTP as Default

If you use the standard Linux *ftp* program, it may seem that you can't run FTP commands on hosts outside your local network. This can be because your *ftp* client is trying to use the FTP *active mode*, which firewalls may block. There may be a command-line option like `-p` to fix this by using *passive mode*. You can try running *pftp* instead of *ftp*. Or you can type `passive` from an `ftp>` prompt. You can also make passive mode the default by adding three lines (the last line is empty) to the file *.netrc* in your home directory:

```
default macdef init
passive

```

A `default macdef init` line defines a macro named `init` used on all machines that don't have a specific entry in your *.netrc* file. Be sure that *.netrc* is readable only by you (run `chmod 600 .netrc`).

Jerry Peek is a freelance writer and instructor who has used Unix and Linux for over 20 years. He's happy to hear from readers; see http://www.jpeek.com/contact.html.