Network Working Group
Request for Comments: 1635
FYI: 24
Category: Informational

P. Deutsch
A. Emtage
Bunyip
A. Marine
NASA NAIC

May 1994

How to Use Anonymous FTP

Status of this Memo

This memo provides information for the Internet community. This memo does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Abstract

This document provides information for the novice Internet user about using the File Transfer Protocol (FTP). It explains what FTP is, what anonymous FTP is, and what an anonymous FTP archive site is. It shows a sample anonymous FTP session. It also discusses common ways files are packaged for efficient storage and transmission.

Acknowledgements

This document is the result of work done in the Internet Anonymous FTP Archives (IAFA) working group of the IETF. Special thanks are due to Mark Baushke (Cisco), John Curran (BBN), Aydin Edguer (CWRU), Rafal Maszkowski (Onsala Space Observatory), Marsha Perrott (PREPnet), Bob Peterson (Texas Instruments), Nathan Torkington (Victoria University of Wellington), and Stephen Tihor (NYU) for excellent comments and contributions.

What is FTP?

FTP refers to the File Transfer Protocol [1], one of the protocols within the TCP/IP protocol suite used on the Internet. The File Transfer Protocol makes it possible to transfer files from one computer (or host) on the Internet to another. There are many FTP implementations built on the specification of the FTP protocol. A user of an FTP program must log in to both hosts in order to transfer a file from one to the other.

It is common for a user with files on more than one host to use the FTP program to transfer files from one host to another. In this case, the user has an account on both hosts involved, so he has passwords for both hosts.

However, Internet users may also take advantage of a wealth of information available from archive sites by using a general purpose account called "anonymous FTP".

What is an Archive Site?

An archive site is a host that acts as a repository of information, much like a conventional library. Information stored on these Internet hosts is made available for users to transfer to their local sites. Users run software to identify this information and transfer it to their own hosts. Such a transfer is done with a program that implements the File Transfer Protocol (FTP).

What is Anonymous FTP?

Anonymous FTP is a means by which archive sites allow general access to their archives of information. These sites create a special account called "anonymous". User "anonymous" has limited access rights to the archive host, as well as some operating restrictions. In fact, the only operations allowed are logging in using FTP, listing the contents of a limited set of directories, and retrieving files. Some sites limit the contents of a directory listing an anonymous user can see as well. Note that "anonymous" users are not usually allowed to transfer files TO the archive site, but can only retrieve files from such a site.

Traditionally, this special anonymous user account accepts any string as a password, although it is common to use either the password "guest" or one's electronic mail (e-mail) address. Some archive sites now explicitly ask for the user's e-mail address and will not allow login with the "guest" password. Providing an e-mail address is a courtesy that allows archive site operators to get some idea of who is using their services.

What Information Do You Need to Know?

To retrieve a specific file, a user needs to know what host it is on, and the pathname of the file. A pathname tells the directory (and possibly subdirectories) that house the file, and the name of the file. Often discussions of available files will not specifically say, "This file is available for anonymous FTP from X host with Y pathname". However, if a file is publicly announced as available and referred to as something like pub/good-stuff on nisc.sri.com, it is a good assumption that you can try to transfer it.

You may also need to know if your machine uses an ASCII, EBCDIC, or other character set to know how likely a transfer of binary information will work, or whether such a transfer will require other

keywords, such as is true for TENEX.

In the general case, you may assume that an ASCII transfer will always do the right thing for plain text files. However, more and more information is being stored in various compressed formats (which are discussed later in this document), so knowing the binary characteristics of your machine may be important.

A Sample Session

To start an FTP session on a UNIX or VMS host, you type "ftp" and the host name or host IP address of the machine to which you want to connect. For example, if you wish to access the NASA Network Applications and Information Center archive site, you would normally execute one of the following commands at the UNIX prompt:

```
        ftp naic.nasa.gov
```

or

```
        ftp 128.102.128.6
```

Observe that the first form uses the fully-qualified domain name and the second uses the Internet address for the same host.

The following is an example of connecting to the naic.nasa.gov host to retrieve STD 9, RFC 959, "File Transfer Protocol (FTP)" [1].

Note several things about the session.

1. Every response the FTP program at the archive site gives is preceded by a number. These numbers are called Reply Codes and are defined in the FTP specification, RFC 959. The text that accompanies these reply codes can vary in different FTP implementations, and usually does.

   Also note that some FTP client implementations (e.g., MVS systems) may not echo the reply codes or text as transmitted from the remote host. They may generate their own status lines or just hide the non-fatal replies from you. For the purposes of this document, the more popular UNIX interface to the FTP client will be presented.

2. The password you type is never shown on your screen.

3. It is possible to "browse" in archives, but most often users already know the pathname of the file they want. The pathname for RFC 959 on this host is files/rfc/rfc959.txt. In the

example, we first connect to the 'files/rfc' directory (cd files/rfc), then get the specific file we know we want. If you do not know the name of the file you want, a file called README or something similar (00README.1ST, AAREAD.ME, INDEX, etc.) is probably the one to retrieve first.

```
atlas.arc.nasa.gov% ftp naic.nasa.gov
Connected to naic.nasa.gov.
220 naic.nasa.gov FTP server (Wed May 4 12:15:15 PDT 1994) ready.
Name (naic.nasa.gov:amarine): anonymous
331 Guest login ok, send your complete e-mail address as password.
Password:
230-----------------------------------------------------------------
230-Welcome to the NASA Network Applications and Info Center Archive
230-
230-     Access to NAIC's online services is also available through:
230-
230-        Gopher          - naic.nasa.gov (port 70)
230-    World-Wide-Web - http://naic.nasa.gov/naic/naic-home.html
230-
230-        If you experience any problems please send email to
230-
230-                     naic@nasa.gov
230-
230-                 or call +1 (800) 858-9947
230-----------------------------------------------------------------
230-
230-Please read the file README
230-  it was last modified on Fri Dec 10 13:06:33 1993 - 165 days ago
230 Guest login ok, access restrictions apply.
ftp> cd files/rfc
250-Please read the file README.rfc
250-  it was last modified on Fri Jul 30 16:47:29 1993 - 298 days ago
250 CWD command successful.
ftp> get rfc959.txt
200 PORT command successful.
150 Opening ASCII mode data connection for rfc959.txt (147316 bytes).
226 Transfer complete.
local: rfc959.txt remote: rfc959.txt
151249 bytes received in 0.9 seconds (1.6e+02 Kbytes/s)
ftp> quit
221 Goodbye.
atlas.arc.nasa.gov%
```

Variations

The above example is of the FTP program available on UNIX systems. Other operating systems also make FTP programs available. The actual commands you type may vary somewhat with other programs. However, in general, you will do the following with every FTP program:

- Log in to your local host, and invoke the FTP program.
- Open a connection to the host (using either the host name or its IP address)
- Once connected to the remote host, log in with username "anonymous".
- Provide either the password "guest" or whatever the password the site requests.
- Issue whatever FTP commands you require, such as those to change directories or to retrieve a file.
- When finished, exit the FTP program, which will close your connection to the archive host.

Friendly Servers

These days, many sites are using a form of FTP that allows them to display several lines of explanatory text that help direct users through their archive. The listing of alternative services on naic.nasa.gov is an example. If these effusive servers confuse the client you are using, try typing a hyphen (-) before your password when you log in. That should disable the verbose mode of the server.

Other FTP Commands

We have demonstrated some of the commands available with FTP programs. Many others are possible. For example, once you have logged in to a remote host:

- You may ask the FTP program to display a list of available commands, typically by invoking the FTP program without arguments and typing "help".
- You may view the contents of the directory to which you are connected. Type "dir" or "ls" to do so.
- You may rename a file by using the "get" command’s optional local file name, which follows the remote file

name on the command line. You probably should rename a file when the remote file name exceeds your local file system's naming constraints, e.g., if the remote file name is too long. An example of using the "get" command to rename a file when transferring it might be "get really-long-named-file.txt short.txt".

- You may set BINARY mode to transfer executable programs or files of data. Type "binary" to do so. Usually FTP programs assume files use only 7 bits per byte, the norm for standard ASCII-encoded files. The BINARY command allows you to transfer files that use the full 8 bits per byte without error, but this may have implications on how the file is transferred to your local system.

  If you are not sure what format a file is in, you may need to transfer it a second time in the other mode (BINARY or ASCII) if your first guess is wrong. The extension at the end of the file name may give you a clue. File name extensions are described below.

  Because some machines store text files differently than others, you may have to try your luck if you're not sure what format a file is in. A good guess is to try ASCII mode first, if you have grounds to suspect the file is a text file. Otherwise, try BINARY mode. Try TENEX mode as a last resort.

- You may transfer multiple files at the same time. To set this mode, type "mget". You then supply a file name pattern that the remote system understands and it tries to transfer each file in turn. If your local FTP user agent cannot transform the remote file names into legal local file names, or if there are some files that must be transferred in ASCII mode and others that must be transferred in BINARY mode, you may not be able to take advantage of this facility.

Full details on the commands and options available are in the FTP documentation that comes with your system. You can also type "help" at the FTP command prompt for a list of command options.

A copy of the UNIX version of the FTP documentation is available from the online manual. If your UNIX site has the manuals installed, type the following at the UNIX prompt:

```
% man ftp
```

The Packaging and Naming of Files

Several widely used conventions allow for efficient storage and transmission of information stored at archive sites.

Information stored on archive sites is often "transformed" in three common ways. "Compressing" (reducing the size of) the stored information makes more space available on the archive, and reduces the amount of data actually transferred across the network. "Bundling" several files into one larger file maintains the internal directory structure of the components, and allows users to transfer only one larger object rather than several (sometimes hundreds) of smaller files.

In addition, binary data is often converted into an ASCII format for transmission, a process referred to in this document as "transformation". Traditionally, Internet RFC 822-based electronic mail and USENET protocols did not allow the transmission of "binary" (8-bit) data; therefore, files in binary format had to be transformed into printable 7-bit ASCII before being transmission.

On many systems, various file naming conventions are used to help the remote user to determine the format of the stored information without first having to retrieve the files. Below we list the more common compression, bundling, and transformation conventions used on the Internet. This list is not intended to be exhaustive. In all cases public domain or freely-available implementations of the programs associated with these mechanisms are available on the network.

1) compress/uncompress

Filenames terminating in ".Z" normally signify files that have been compressed by the standard UNIX Lempel-Ziv "compress" utility. There is an equivalent program called "uncompress" to reverse the process and return the file to its original state. No bundling mechanism is provided, and the resulting files are always in binary format, regardless of the original format of the input data.

2) atob/btoa

Performs a transformation of ASCII to binary (atob) and the reverse (btoa) in a standard format. Files so transformed often have filenames terminated with ".atob". No bundling or compression mechanisms are used.

3) atox/xtoa

A data transformation standard used to convert binary files to transferable ASCII format. Sometimes used in preference to other similar mechanisms because it is more space efficient; however, it is not a compression mechanism per se. It is just more efficient in the transformation from one format to the other. Filenames of files in this format often have the ".atox" extension.

4) uuencode/uudecode

Transforms binary to ASCII ("uuencode") and the reverse ("uudecode") transformation in a standard manner. Originally used in the UUCP ("Unix to Unix CoPy") mail/USENET system. No bundling or compression mechanisms are used. Naming conventions often add a .uu at the end of the file name.

5) tar/untar

Originally a UNIX based utility for bundling (and unbundling) several files and directories into (and from) a single file (the acronym stands for "Tape ARchive"). Standard format provides no compression mechanism. The resulting bundled file is always in binary format regardless of whether the constituent files are binary or not. Naming conventions usually hold that the filename of a "tarfile" contain the sequence ".tar" or "-tar".

6) zip/unzip

Often used in IBM PC environments, these complementary programs provide both bundling and compression mechanisms. The resulting files are always in binary format. Files resulting from the "zip" program are by convention terminated with the ".zip" filename extension.

7) arc/unarc

Often used in IBM PC environments, these complementary programs provide both bundling and compression mechanisms. The resulting files are always in binary format. Files stored in this format often have a ".arc" filename extension.

8) binhex

Used in the Apple MacIntosh environment, the binhex process provides bundling as well as binary to ASCII data transformations. Files in this format by convention have a filename extension of ".hqx".

9) shar

Bourse shell archives package text or binary files into a single longer file which, when executed, will create the component files. Because this format is vulnerable to misuse, most users use a special tool called unshar to decode these archives. By convention, files in this format have a filename extension of ".shar".

10) VMS_SHARE

DCL archives package text or binary files into a single longer file which, when executed, will created the component files. Because this format is vulnerable to misuse, care must be take to examine such an archive before executing it. By convention, files in this format have a filename extension of ".shar".

11) Multipart shar/vms_share files

Sometimes these shell archive files are broken into multiple small parts to simplify their transfer over other forms of fileservers that share the same archive tree. In such cases, the parts of the files are usually suffixed with a part number (e.g., xyz.01 xyz.02 xyz.03 ... or even .01-of-05). Collect all the parts, concatenate them on your local system, and then apply the procedure listed above for a simple shar or vms_share file to the concatenated file you just made.

12) zoo

The zoo program implements compression/decompression and bundling/unbundling in a single program. Utilities supporting the zoo format exist on a wide variety of systems, including Unix, MS-DOS, Macintosh, OS/2, Atari ST, and VAX VMS. Files created by the "zoo" programs by convention end with the ".zoo" filename extension. Zoo is a popular distribution format due to the availability of free implementations (both source and executable code) on a wide variety of operating systems.

13) gzip/gunzip

The Free Software Foundation GNU project adopted a variant of the zip compression mechanism as a substitute for the compress/uncompress commands. The resulting files are always in binary format. Files resulting from the "gzip" program are by convention terminated with the ".z" or ".gz" filename extensions. The gunzip program also recognizes ".tgz" and ".taz" as shorthands for ".tar.z" or ".tar.Z". Also, gunzip can recognize and decompress files created by the gzip, zip, compress, or pack commands.

The GNU project recently began distributing and using the gzip/gunzip utilities. Even more recently they changed the default suffix from .z to .gz, in an attempt to (1) reduce confusion with .Z, and (2) eliminate a problem with case-insensitive file systems such as MS-DOS. The gzip software is freely redistributable and has been ported to most UNIX systems, as well as Amiga, Atari, MSDOS, OS2, and VMS systems.

In some cases, a series of the above processes are performed to produce the final file as stored on the archive. In cases where multiple transformation processes have been used, tradition holds that the original (base) filename be changed to reflect these processes, and that the associated filename extensions be added in the order in which the processes were performed. For example, a common procedure is first to bundle the original files and directories using the "tar" process, then to "compress" the bundled file. Starting with a base file name of "foobar", the file name in the archive would become "foobar.tar.Z". As this is a binary file, it would require a further transformation into printable ASCII by a program such as "uuencode" in order to be transmitted over traditional email or USENET facilities, so it might finally be called "foobar.tar.Z.uu."

Some operating systems can not handle multiple periods; in such cases they are often replaced by hyphen (-), underscore (_), or by detailed instructions in the "read me" files in the directories.

Compress and Tar

Here is an example of the use of the "compress/uncompress" and "tar/untar" programs.

Suppose "patch" is a useful public domain program for applying program patches and updates. You find this file at an archive site as "patch.tar.Z". Now you know that the ".Z" indicates that the file

was compressed with the UNIX "compress" command, and the ".tar" indicates that it was tar'ed using the UNIX "tar" tape archive command.

First retrieve the file onto your machine using anonymous FTP. To unpack this program, you would first uncompress it by typing:

```
uncompress patch.tar.Z
```

This will uncompress the file, and in the process, rename it to "patch.tar". You can then execute the "tar" command to extract the individual files.

In the example of patch.tar, you could invoke the command as:

```
%tar xvf patch.tar
```

The files would be extracted (that's the 'x' argument to tar) from the file patch.tar (that's the 'f' argument). Because we use the 'v' (for verbose) argument, the name of each file is printed as it is extracted. When tar is complete you should have all the files that make up the "patch" program in your working directory.

Etiquette

Not every site that supports FTP permits anonymous tranfers. It is wrong to try to get files from systems that have not advertised the availability of such a service.

Remember that Internet site administrators for archive sites have made their systems available out of a sense of community. Rarely are they fully compensated for the time and effort it takes to administer such a site. There are some things users can do to make their jobs somewhat easier, such as checking with local support personnel first if problems occur before asking the archive administrator for help.

Most archive machines perform other functions as well. Please respect the needs of their primary users and restrict your FTP access to non-prime hours (generally between 1900 and 0600 hours local time for that site) whenever possible. It is especially important to remember this for sites located on another continent or across a significant body of water because most such links are relatively slow and heavily loaded.

In addition, some sites offering anonymous FTP limit the number of concurrent anonymous FTP logins. If your attempt to log onto such a site results in an error message to the effect that too many anonymous FTP users are online, you should wait a while before attempting another connection rather than retrying immediately.

To reduce redundant storage, you should find out how to make useful the files you fetch using FTP available to your entire organization. If you retrieve and test a program that turns out to be useful, you should probably ask your administrator to consider making the program generally available, which will reduce the redundant effort and disk space resulting from multiple individuals installing the same package in their personal directories.

If you find an interesting file or program on an archive site, tell others about it. You should not copy the file or program to your own archive unless you are willing to keep your copy current.

References

[1] Postel, J., and J. Reynolds, "File Transfer Protocol (FTP)", STD 9, RFC 959, USC/Information Sciences Institute, October 1985.

Security Considerations

Security issues are not discussed in this memo.

Authors' Addresses

Peter Deutsch
Bunyip Information Systems
266 Blvd. Neptune
Dorval, Quebec, H9S 2L4
Canada

Phone: (514) 398-3709
EMail: peterd@bunyip.com

Alan Emtage
Bunyip Information Systems
266 Blvd. Neptune
Dorval, Quebec, H9S 2L4
Canada

Phone: (514) 398-3709
EMail: bajan@bunyip.com

April N. Marine
NASA NAIC
M/S 204-14
Ames Research Center
Moffett Field, CA 94035-1000

Phone: (415) 604-0762
EMail: amarine@atlas.arc.nasa.gov