How to use ftp - g4poi July 12th 1993

How to use ftp on g4poi hub - g4poi July 12th 1993

A PRACTICAL GUIDE TO USING FTP

The following describes how to use ftp to send and retrieve files to and from the g4poi hub (or any other station).

FTP:

ftp means "File Transfer Protocol". ftp is used primarily to transfer large amounts of information or programs where smtp (simple mail transfer protocol) is inappropriate or cannot handle non printable characters.

CONNECTING TO AN FTP HOST:

The syntax of the command ftp is; ftp <host>
<host> is the callsign of the station you wish to act as an ftp server. (That is, as a station that you can get files from or send them to).

So at your command screen, you type; ftp g4poi

Your own computer will open a new window (Amiga) or give you a new empty screen (PC) with session information at the top.

Before you can type any commands, you will need to log into the ftp server on the remote station, the host. This is usually automatic but if it isn't, type your callsign where it asks for your name, and type your name when it asks for the password.

You can make logging on automatic by changing the contents of your NOS.RC file. If you wish to log into g4poi for instance and your callsign is g9xyz and your name is ethelbert, the entry in your NOS.RC file should read;

```
g4poi g9xyz ethelbert
```

And providing the host knows you(!) they will give you permission to move around quite freely.

On the g4poi hub, the ftp server tells you where you are in the directories before giving you a prompt. So if you are a known user on the system, it will say something like;

```
230- "/usr/g9zzz" is the current directory
ftp>
```

If you are not known on the system, you will only have access to the public areas so you would see;

```
230- Logged in as anonymous - restrictions apply
```

and;

```
230- "/public" is the current directory
ftp>
```

The "ftp>" is the prompt I mentioned a moment ago. When you see this, it means that the "host" is ready for you to type a command. If you do not see this, be patient. The "host" station is probably a little busy.

230- The funny number at the start of each message line is just a part
230- of ftp. You can tell the "host" to cut out the waffle by specifying
230- different verbose levels, but that comes later...

NAVIGATING THE DIRECTORIES:
==========

As users of the g4poi hub, you will be in your /usr/callsign area when you first connect. You do not have to be put there first. I can alter your configurations on the system so you arrive at /public if you wish.

First, let's clear up the confusion about the use of "/".

If you see a filename or a directory name with a "/" to the left of it, and nothing to the left of the "/", it means that file/directory is right at the root of the directory structure. Examples;

```
/public     \
             |
/in-tray     |-- Are all right down as low as you can get in the "tree".
             |   In other words, there is nothing beneath them to get
/usr        /    into (at least that's how it appears).
```

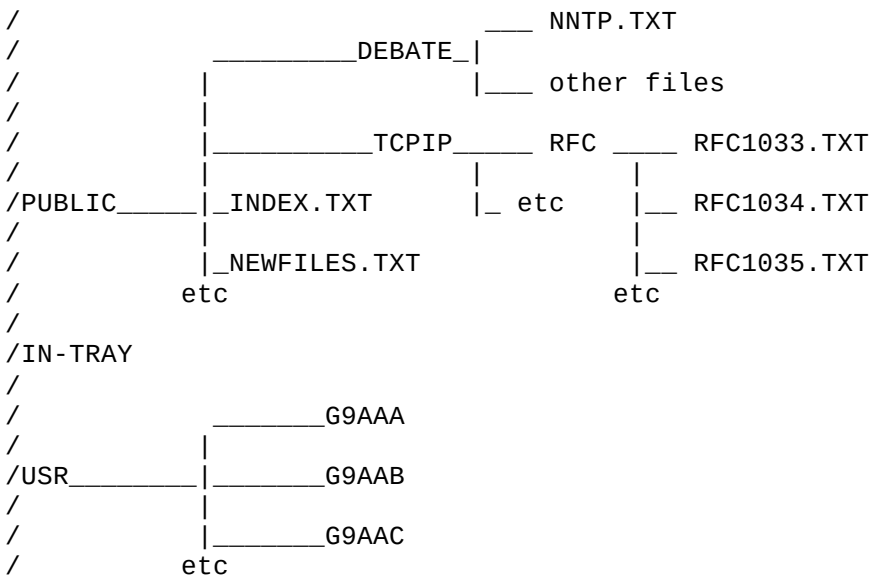
If you see another "/" separating one directory name from another or from a file, it means that it is being used to say that to get to this file (or directory), you must get to it using this particular path.

Example;

```
/public/debate/nntp.txt
   |      |       |_____ to get to this file (in this instance)
   |      |_______________ you need to come via here
   |______________________ from here.
```

It may be clearer if we include some of the things that we DON'T want to make the example clearer (it sounds like double-dutch to me too!).

In the /public area, there are MANY other directories connected to it
just like the branches of a tree. Each "branch" has a name.



A few more examples may (I hope) make it clearer.

Required file	Path to it
INDEX.TXT	/public
NNTP.TXT	/public/debate
RFC1033.TXT	/public/tcpip/rfc

I have written the PINDEX (Public INDEX) program to make this navigation
easier. It gives the path to the file and the details and nature of
the file.

GETTING TO THE REQUIRED DIRECTORY

These are the commands that enable you to navigate the directories,
to see where you are, and to see what's available.

COMMAND	MEANS
cd	Change Directory
pwd	Print Working Directory (i.e. where am I?).
dir	DIRectory - lists all the files and directories available. in the working directory (i.e. where we are).
ls	List contents of this directory (same as dir but without dates and sizes, i.e. just names).

So, to go back to the previous example, let's say that when we arrive,
we find ourselves in /usr/g9xyz directory.

To get to the /public area, we type; cd /public

No matter where we are, cd /public will take us all the way down to the
bottom of the tree into the PUBLIC directory. The "/" makes sure of that.
It's like saying "take me to the public directory that is at the root
of the tree".

If you were to type; cd public instead, then the host would look for a directory called "public" in where you are at that moment. If you are in the root directory, then the host would find "public" because "public" IS in the root directory. But if you were up the tree a bit in say /public/tcpip/rfc then typing cd public would not do very much at all because there is no directory called "public" in the rfc directory.

Just think of directories as drawers. In those drawers there are items and other mini drawers. Sort of boxes within boxes. The drawers are the directories, and the odd items in the drawers are the files that are stored there.

So by typing; cd /public we can make sure we are where we want to be.

To move up into the branches of the directory tree, we use "cd" again. We can either do it all in one go or in steps.

First in steps. Our target directory is the RFC one because we are really keen to know about tcp/ip... aren't we?...

cd /public puts us right down the bottom, but we know where we are when we start here.

cd tcpip note that there isn't a "/" before "tcpip". That's because "tcpip" isn't down at the "root" of the tree.

cd rfc whoopee! We made it.

That was the slow way of doing it, but can you see the logic in it? We started by first going to the public directory. We then told the host that we wanted to go into the tcpip directory which it should look for in the public directory. Then we said go into the rfc directory which it should look for in the tcpip directory. Providing that the directories we've asked for exist, everything will go to plan.

And now the quick way;

cd /public/tcpip/rfc

Whammo, kappow! We're there in a twinkling of an eye. This is like saying "Go to rfc which is in tcpip which is in public. You know where it is now so take me straight there." And it does!

There are two ways of climbing back down the tree form /public/tcpip/rfc.

The hard way;

cd /public/tcpip takes us back down by one directory to the "parent" directory. i.e. tcpip is the "parent" of rfc and rfc is the "child" of tcpip. I'm sneaking the terminology used in the industry into this explanation so you'll understand what other people are talking about.

The easy way;

cd .. (gasp!) Wasn't that easy? ".." means PARENT directory, so saying cd .. means, "change directory from the directory where I am to it's parent (i.e. the one immediately below this one)".

Not wishing to confuse you but you should know that you can jump more than one directory backwards by using the / again, like this;

```
cd ../..            would take us back to the parent of the parent!!!
                    but that's just for information.
```

So now we know how to climb the tree and how to climb down again, let's see how to examine what is available by way of files at each "branch".

```
cd /public          takes us all the way to the bottom of the tree into
                    the public directory.

dir                 will list all of the files and child directories
                    available to us. (so will typing ls).
```

So we can see what is available at each step by using dir or ls.

Some systems don't tell you where you are, so if you aren't sure, just type; pwd at the ftp> prompt. To remind you, it means "Print the Working Directory" or in plain english, "Tell me where the heck I am!" (yes I know that's not what you say but it's my license!).

GETTING A FILE:

Getting a file is easy. Deciding where to put it isn't so I recommend that you do the following.

On your computer, make a directory off /nos (PC) or tcpip: (amiga) called TRANSIT (or some other type of van... just a joke) for files in transit.

Now the rest is easy.

"cd" TO THE DIRECTORY WHERE THE FILE IS you want to "get". Let's start with getting the index.txt file since that will open the rest up for you.

Amiga

```
cd /public
get index.txt tcpip:transit/index.txt
```

PC

```
cd /public
get index.txt /transit/index.txt
```

Do you see that in both cases we have to tell our own system where to put the file that it receives?

I definitely recommend that you don't download files directly into your NOS or TCPIP: directories as you will end up overwriting some essential system file sooner rather than later!!! (Guess how I know...).

The heart of this is the "get" command.

Explained:

```
          get index.txt /transit/index.txt
           |      |          |        |______ What it's to be called on
           |      |          |                our own system.
           |      |          |
           |      |          |________________ The path on our own system
           |      |                           where the file is to be put
           |      |
           |      |_____________________________ What the file is called on
           |                                     the host
           |
           |_____________________________________ ftp command "get" this file


          get <hostfile> <put_it_here_and>/<call_it_this>
```

PUTTING A FILE:
==========

Exactly the same principle as "get", put behaves the same.

```
          put /transit/myfile.txt g9xxx.txt
           |      |          |        |______ What it's to be called on
           |      |          |                the host system.
           |      |          |
           |      |          |________________ What the file is called on
           |      |                           your system
           |      |
           |      |_____________________________ The path to your file on
           |                                     your system
           |
           |_____________________________________ ftp command "get" this file
```

Note that with the "put" command, we have to tell our own system where to find the file that we wish to send!

A QUICK WORD ABOUT FILE NAMES:
==========

Different computers can use different filename conventions. For instance the Amiga and Unix machines allow 32 character filenames. The Amiga allows you to have spaces within the filenames too! And clearly, the lack of conformity will cause grief somewhere along the line which is why I recommend you use the PC filename convention. All other machines (apart from the BBC micro if I remember rightly) will accept the PC format of filenames so that's the one I recommend you use.

The PC allows you to have 8 characters followed by a full stop followed by 3 characters with no spaces in-between. Also, certain characters are not allowed on the PC where they might be on other machines.

Here are some examples of valid PC filenames;

MY_FILE	ANOTHER.FIL	AN-OTHER.1	ABCDEFGH.XYZ
1.BAT	NOS.EXE	COMMAND.COM	THIS_IS.OK2

And here are some definite no-no's;

$.USD	BRIGHT*.1	MY.FILE	QUESTION.???
MY .FIL	.POI	>ANOTHER.CHR	AND<THIS.TOO

The problems (in order of appearance) are;

- dollar is a special character
- asterisk is a special character
- my.file has four letters after the '.' which is illegal
- question mark is a special character
- spaces in names aren't allowed
- there must be something before the '.' character
- the > character is a special character
- the < character is a special character

What I mean by "special character" is that DOS (the Disk Operating System) on the PC treats these characters differently to others. For instance, the ? character in PC DOS is used to represent a "wild card", that is, it can be substituted by any single character. Likewise, the * is also a wild card that is used to say "any combination of characters". The Amiga likewise uses ? to say any character but instead uses the two character sequence #? to say any combination of any characters. The $ is a very special case on the PC and is used when you do not wish a user to be able to modify or delete a file without a particular program. The > and < characters are for "redirection", which means, sending something somewhere other than where it was going originally. For instance, one could get a file to go to a printer instead of the screen by redirecting the output to LPT1: or PRN: e.g. DIR > LPT1: sends a directory of the files to the printer instead of the screen. The Amiga uses slight variations of the same principle. By the way, don't use the ':' character within the name. Just about every computer uses this character to say that you are referring to some device. e.g. on the PC, C: means the first hard disk drive, and on the Amiga, DF0: means the first floppy disk drive. So only use it in reference to your own computers disks or devices, not in the file names themselves. e.g. (on the Amiga); get afile.txt df0:afile.txt is perfectly fine, but; get afile.txt df0:afile:txt is not.
I'm sure you get the picture. ^

For simplicity, I recommend the following; When naming files, use characters A-Z and numbers 0-9. Where appropriate, use the _ (underscore character) and - (minus) to make names clearer. e.g. FTP_INFO.TXT is a sensible name because it is clear that the file is an information file about ftp that is plain text (.TXT) and not a program.

As a matter of interest, the system often uses the phrase "Permission denied" to cover a multitude of sins, it can mean that you are using a file name that is illegal, and it can also mean that a file already exists of that name (if you are putting one on the host or getting one to your own system). So try changing the name if you see the above message, it may be as simple as that.

There we are, a DOS course thrown in too!... Now back to ftp...

DIFFERENT TYPES OF FILE TRANSFER:

There are what are know as ASCII and IMAGE transfers. IMAGE is usually the default (that's why you will often see "TYPE I" appear when you are getting or putting a file).

IMAGE is for programs and data. Everything remains EXACTLY as it was in the original. This is how you can store Amiga programs on a PC host and vice versa.

ASCII is very useful since it converts line ending of text files to be compatible with your own computer during a get and to that of the host during a put.

So to grab a PC text file and read it on an Amiga, or vice versa, at the ftp> prompt, before you put or get a file, you enter the following;

```
ftp> type a
     ------.
           `- you type this exactly as shown "type a" which means
              "type ascii".
```

And the computer will convert the file for you as you send/receive it.

You can then go ahead and get or put the file, but don't forget to enter;
type i if you are going to get a program next to tell it you want the unconverted image of the file!!!

HOW TO QUIT:

At the ftp> prompt, type; quit

Or you can try punching the boss in the eye. That works too...

OTHER AVAILABLE COMMANDS:

Not all of them. This is a "How to get started" tutorial, not a "So you want to be an expert".

abort

If you are transferring a file and want to abort it, press [ESC] to take you to the command screen and enter; abort
Press enter again to take you back to the ftp session. In a few seconds or so, the ftp transfer will stop and you'll get the ftp> prompt back.

Abort is valid only when a transfer is in progress. When a 'get' or 'put' operation is aborted, a partial copy of the transferred file will be left on the destination machine. This copy must be removed manually if it is unwanted.

dele <remote_filename>

Deletes a file on the host system. On the poi hub, you only have permission to delete files in your own private /usr/callsign areas. This is to make the system safe against mishaps and to make you feel

confident that you can't hurt anything vital while playing with ftp.

hash
====
This is the equivalent to the verbose 3 command (later in this file)

mkdir <remote_directory_name>
=============================
Creates a directory on the host. Have a play with this in your own areas on the hub. It's useful if you are sending stuff to another station and want your files to be kept separate.

An example:

cd /public	puts you in their public directory
mkdir de-g9zzz	creates a new directory called g9zzz
cd de-g9zzz	puts you up in the new directory called /public/de-g9zzz

Note: there is an 8 character limit to directory and file names on PC's (if you don't include a suffix - I'll explain that if you ask nicely).

rmdir <remote_directory_name>
=============================
Means ReMoveDIRectory. You can only do this if you, a) have permissions for it, and, 2) if the directory is empty.

verbose n (where n = 0, 1, 2, or 3)
===================================
i.e. how verbose you want the system messages to be. The four ranges are;

0 - Error messages only.

1 - Error messages, plus a one-line summary after each transfer giving the name of the file, its size, and the transfer time and rate.

2 - Error and summary messages, plus the progress messages generated by the remote FTP server.

3 - Display all messages. In addition, a "hash mark" (#) is displayed for every 1,000 bytes sent or received. "verbose 3" has an alias, typing "hash" has the same effect (see above).

SUMMARY:
========
ftp is a really useful tool. You should practice with it until you are happy transferring files too and from each other and the local hubs since it is a marvellous source of new programs and information.

Have fun, and, Share and Enjoy!

73 de Dave

g4poi.ampr.org. IP 44.131.19.160 Mail to g4poi@g4poi
Sysop to the g4poi hub located in Northaw village, South Herts

Note from G4APL.the above addresses are now invalid.
This document is an example of how to use FTP 21.11.2002