NAME

libnetFAQ - libnet Frequently Asked Questions

DESCRIPTION

Where to get this document

This document is distributed with the libnet distribution, and is also available on the libnet web page at

http://search.cpan.org/dist/libnet/

How to contribute to this document

You may report corrections, additions, and suggestions on the CPAN request tracker at

http://rt.cpan.org/Dist/Display.html?Name=libnet

Author and Copyright Information

Currently maintained by Steve Hay <shay@cpan.org>.

Disclaimer

This information is offered in good faith and in the hope that it may be of use, but is not guaranteed to be correct, up to date, or suitable for any particular purpose whatsoever. The authors accept no liability in respect of this information or its use.

Obtaining and installing libnet

What is libnet ?

libnet is a collection of perl5(1) modules which all related to network programming. The majority of the modules available provided the client side of popular server-client protocols that are used in the internet community.

Which version of perl do I need ?

libnet has been know to work with versions of perl from 5.002 onwards. However if your release of perl is prior to perl5.004 then you will need to obtain and install the IO distribution from CPAN. If you have perl5.004 or later then you will have the IO modules in your installation already, but CPAN may contain updates.

What other modules do I need ?

The only modules you will need installed are the modules from the IO distribution. If you have perl5.004 or later you will already have these modules.

What machines support libnet ?

libnet itself is an entirely perl-code distribution so it should work on any machine that perl runs on. However IO may not work with some machines and earlier releases of perl. But this should not be the case with perl version 5.004 or later.

Where can I get the latest libnet release

The latest libnet release is always on CPAN, you will find it in

http://search.cpan.org/dist/libnet/

Using Net::FTP

How do I download files from an FTP server ?

An example taken from an article posted to comp.lang.perl.misc

```
#!/your/path/to/perl

# a module making life easier

use Net::FTP;

# for debugging: $ftp = Net::FTP->new('site','Debug',10);
```

```
# open a connection and log in!

$ftp = Net::FTP->new('target_site.somewhere.xxx');
$ftp->login('username','password');

# set transfer mode to binary

$ftp->binary();

# change the directory on the ftp site

$ftp->cwd('/some/path/to/somewhere/');

foreach $name ('file1', 'file2', 'file3') {

# get's arguments are in the following order:
# ftp server's filename
# filename to save the transfer to on the local machine
# can be simply used as get($name) if you want the same name

$ftp->get($name,$name);
}

# ftp done!

$ftp->quit;
```

How do I transfer files in binary mode ?

To transfer files without <LF><CR> translation Net::FTP provides the `binary` method

```
$ftp->binary;
```

How can I get the size of a file on a remote FTP server ?
How can I get the modification time of a file on a remote FTP server ?
How can I change the permissions of a file on a remote server ?

The FTP protocol does not have a command for changing the permissions of a file on the remote server. But some ftp servers may allow a chmod command to be issued via a SITE command, eg

```
$ftp->quot('site chmod 0777',$filename);
```

But this is not guaranteed to work.

Can I do a reget operation like the ftp command ?
How do I get a directory listing from an FTP server ?
Changing directory to "" does not fail ?

Passing an argument of to *->cwd()* has the same affect of calling *->cwd()* without any arguments. Turn on Debug (*See below*) and you will see what is happening

```
$ftp = Net::FTP->new($host, Debug => 1);
$ftp->login;
$ftp->cwd("");
```

gives

```
Net::FTP=GLOB(0x82196d8)>>> CWD /
Net::FTP=GLOB(0x82196d8)<<< 250 CWD command successful.
```

I am behind a SOCKS firewall, but the Firewall option does not work ?

The Firewall option is only for support of one type of firewall. The type supported is an ftp proxy.

To use Net::FTP, or any other module in the libnet distribution, through a SOCKS firewall you

must create a socks-ified perl executable by compiling perl with the socks library.

I am behind an FTP proxy firewall, but cannot access machines outside ?

Net::FTP implements the most popular ftp proxy firewall approach. The scheme implemented is that where you log in to the firewall with `user@hostname`

I have heard of one other type of firewall which requires a login to the firewall with an account, then a second login with `user@hostname`. You can still use Net::FTP to traverse these firewalls, but a more manual approach must be taken, eg

```
$ftp = Net::FTP->new($firewall) or die $@;
$ftp->login($firewall_user, $firewall_passwd) or die $ftp->message;
$ftp->login($ext_user . '@' . $ext_host, $ext_passwd) or die $ftp->message.
```

My ftp proxy firewall does not listen on port 21

FTP servers usually listen on the same port number, port 21, as any other FTP server. But there is no reason why this has to be the case.

If you pass a port number to Net::FTP then it assumes this is the port number of the final destination. By default Net::FTP will always try to connect to the firewall on port 21.

Net::FTP uses IO::Socket to open the connection and IO::Socket allows the port number to be specified as part of the hostname. So this problem can be resolved by either passing a Firewall option like `"hostname:1234"` or by setting the `ftp_firewall` option in Net::Config to be a string in the same form.

Is it possible to change the file permissions of a file on an FTP server ?

The answer to this is ‘‘maybe’’. The FTP protocol does not specify a command to change file permissions on a remote host. However many servers do allow you to run the chmod command via the `SITE` command. This can be done with

```
$ftp->site('chmod','0775',$file);
```

I have seen scripts call a method message, but cannot find it documented ?

Net::FTP, like several other packages in libnet, inherits from Net::Cmd, so all the methods described in Net::Cmd are also available on Net::FTP objects.

Why does Net::FTP not implement mput and mget methods

Net::FTP not implement mput and mget methods The quick answer is because they are easy to implement yourself. The long answer is that to write these in such a way that multiple platforms are supported correctly would just require too much code. Below are some examples how you can implement these yourself.

sub mput { my($ftp,$pattern) = `@_`; foreach my `$file` (glob($pattern)) { `$ftp`->put($file) or warn `$ftp`->message; } }

sub mget { my($ftp,$pattern) = `@_`; foreach my `$file` ($ftp->ls($pattern)) { `$ftp`->get($file) or warn `$ftp`->message; } }

Using Net::SMTP

Why can’t the part of an Email address after the @ be used as the hostname ?

The part of an Email address which follows the @ is not necessarily a hostname, it is a mail domain. To find the name of a host to connect for a mail domain you need to do a DNS MX lookup

Why does Net::SMTP not do DNS MX lookups ?

Net::SMTP not do DNS MX lookups ? Net::SMTP implements the SMTP protocol. The DNS MX lookup is not part of this protocol.

The verify method always returns true ?

Well it may seem that way, but it does not. The verify method returns true if the command succeeded. If you pass verify an address which the server would normally have to forward to another machine, the command will succeed with something like

```
252 Couldn't verify <someone@there> but will attempt delivery anyway
```

This command will fail only if you pass it an address in a domain the server directly delivers for, and that address does not exist.

Debugging scripts

How can I debug my scripts that use Net::* modules ?

Most of the libnet client classes allow options to be passed to the constructor, in most cases one option is called `Debug`. Passing this option with a non-zero value will turn on a protocol trace, which will be sent to STDERR. This trace can be useful to see what commands are being sent to the remote server and what responses are being received back.

```
#!/your/path/to/perl

use Net::FTP;

my $ftp = new Net::FTP($host, Debug => 1);
$ftp->login('gbarr','password');
$ftp->quit;
```

this script would output something like

```
Net::FTP: Net::FTP(2.22)
Net::FTP: Exporter
Net::FTP: Net::Cmd(2.0801)
Net::FTP: IO::Socket::INET
Net::FTP: IO::Socket(1.1603)
Net::FTP: IO::Handle(1.1504)

Net::FTP=GLOB(0x8152974)<<< 220 imagine FTP server (Version wu-2.4(5) Tue Jul 29 11:17:18 CD
Net::FTP=GLOB(0x8152974)>>> user gbarr
Net::FTP=GLOB(0x8152974)<<< 331 Password required for gbarr.
Net::FTP=GLOB(0x8152974)>>> PASS ....
Net::FTP=GLOB(0x8152974)<<< 230 User gbarr logged in. Access restrictions apply.
Net::FTP=GLOB(0x8152974)>>> QUIT
Net::FTP=GLOB(0x8152974)<<< 221 Goodbye.
```

The first few lines tell you the modules that Net::FTP uses and their versions, this is useful data to me when a user reports a bug. The last seven lines show the communication with the server. Each line has three parts. The first part is the object itself, this is useful for separating the output if you are using multiple objects. The second part is either `<<<<` to show data coming from the server or `&gt&gt&gt&gt` to show data going to the server. The remainder of the line is the command being sent or response being received.

AUTHOR AND COPYRIGHT