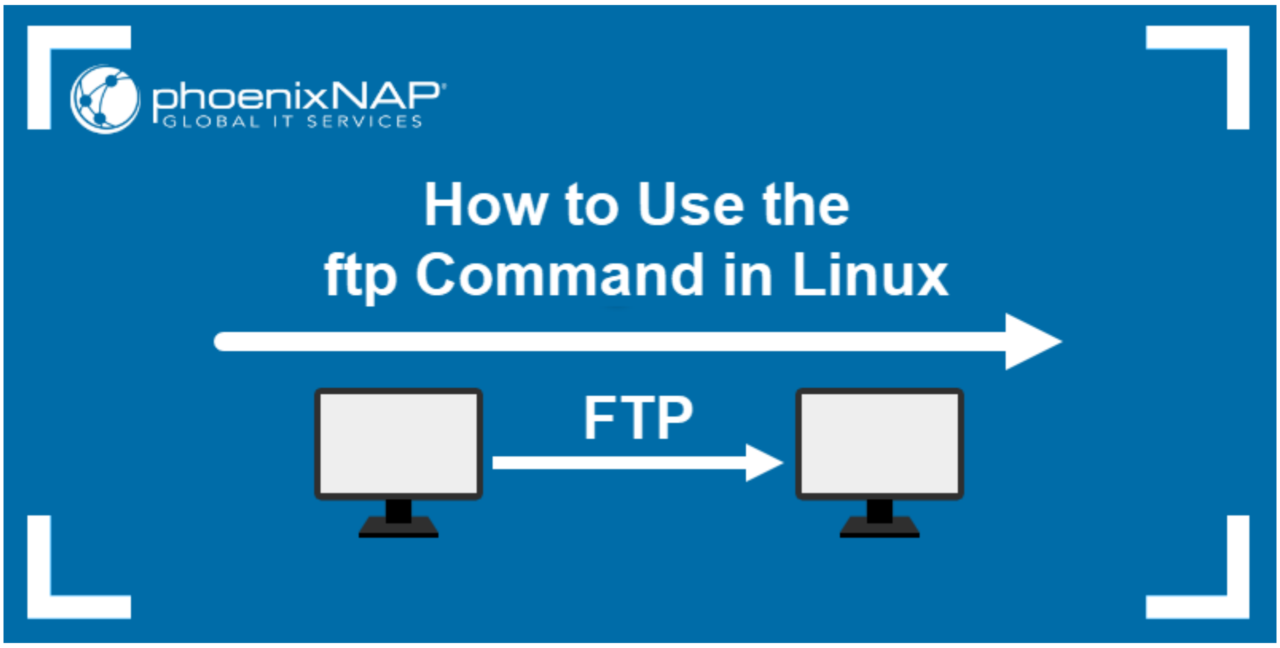


Introduction

FTP (File Transfer Protocol) is a network protocol used for transferring files from one computer system to another. Even though the safety of FTP tends to spark a lot of discussion, it is still an effective method of transferring files within a secure network.

In this tutorial, we will show you how to use the `ftp` command to connect to a remote system, transfer files, and manage files and directories.

Prerequisites

- Access to a local system and a remote FTP server.
- A working Internet connection.
- Access to the terminal window.

IMPORTANT: FTP traffic is not encrypted and is thus considered unsafe. It is not recommended to transfer files over the Internet using FTP. To learn more about secure alternatives to FTP, have a look at our articles on SFTP and TSL vs. SSL.

Linux ftp Command Syntax

The Linux `ftp` command uses the following basic syntax:

```
ftp [options] [IP]
```

The `IP` is the IP address of the system you are connecting to.

The options available for the `ftp` command are:

FTP Command Options	Description
`-4`	Use only IPv4.
`-6`	Use only IPv6.
`-e`	Disables command editing and history support.
`-p`	Uses passive mode for data transfers, allowing you to use FTP despite a firewall that might prevent it.
`-i`	Turns off interactive prompting during multiple file transfers.
`-n`	Disables auto-login attempts on initial connection.
`-g`	Disables file name globbing.
`-v`	Enables verbose output.
`-d`	Enables debugging.

The `ftp` command connects you to a remote system and initiates the FTP interface. The FTP interface uses the following commands to manage and transfer files to the remote system:

Command	Description
`!`	Temporarily escape to the local shell.
`$`	Execute a macro.
`?`	Display help text.
`account`	Supply a password for the remote system.
`append`	Append a local file to a file on the remote system.
`ascii`	Set the file transfer type to network ASCII (default type).
`bell`	Enable a sound alert after each transfer is complete.
`binary`	Set the file transfer type to binary image transfer.
`bye`	Exit the FTP interface.
`case`	Toggle upper/lower case sensitivity when ID mapping during the `mget` command.
`cd`	Change the current working directory on the remote system.
`cdup`	Change to the parent of the current working directory on the remote system.
`chmod`	Change file permissions on the remote system.
`close`	Exit the FTP interface.
`cr`	Toggle carriage return stripping on ASCII filetransfers.
`debug`	Toggle debugging mode.
`delete`	Delete a file from the remote system.
`dir`	List the contents of a directory on the remote system.
`disconnect`	Terminate the FTP session.
`exit`	Terminate the FTP session and exit the FTP interface.

Command	Description
`form`	Set the file transfer format.
`get`	Transfer a file from the remote system to the local machine.
`glob`	Toggle meta character expansion of local file names.
`hash`	Toggle displaying the hash sign ("#") for each transferred data block.
`help`	Display help text.
`idle`	Set an inactivity timer for the remote system.
`image`	Set the file transfer type to binary image transfer.
`ipany`	Allow any type of IP address.
`ipv4`	Only allow IPv4 addresses.
`ipv6`	Only allow IPv6 addresses.
`lcd`	Change the current working directory on the local machine.
`ls`	List the contents of a directory on the remote system.
`macdef`	Define a macro.
`mdelete`	Delete multiple files on the remote system.
`mdir`	List the contents of multiple directories on the remote system.
`mget`	Transfer multiple files from the remote system to the local machine.
`mkdir`	Create a directory on the remote system.
`mls`	List the contents of multiple directories on the remote system.
`mode`	Set the file transfer mode.
`modtime`	Show the last time a file on the remote system was modified.
`mput`	Transfer multiple files from the local machine to the remote system.
`newer`	Transfer a file from the remote system to the local machine only if the modification time of the remote file is more recent than that of the local file (if a local version of the file doesn't exist, the remote file is automatically considered newer).
`nlist`	List the contents of a directory on the remote system.
`nmap`	Set templates for default file name mapping.
`ntrans`	Set translation table for default file name mapping.
`open`	Establish a connection with an FTP server.
`passive`	Enable passive transfer mode.
`prompt`	Force interactive prompts when transferring multiple files.
`proxy`	Execute command on an alternate (proxy) connection.
`put`	Transfer a file from the local machine to the remote system.
`pwd`	Display the current working directory on the remote system.
`qc`	Toggle displaying a control character ("?") in the output of ASCII type commands.
`quit`	Terminate the FTP session and exit the FTP interface.
`quote`	Specify a command as an argument and send it to the FTP server.
`recv`	Transfer a file from the remote system to the local machine.
`reget`	Transfer a file from the remote system to the local machine if the local file is smaller than the remote file. The transfer starts at the end of the local file. If there is no local version of the file, the command doesn't execute.
`rename`	Rename a file on the remote system.
`reset`	Clear queued command replies.
`restart`	Restart a file transfer command at a set marker.

Command	Description
`rhelp`	Display help text for the remote system.
`rmdir`	Remove a directory on the remote system.
`rstatus`	Show the status of the remote system.
`runique`	Toggle storing files on the local machine with unique filenames.
`send`	Transfer a file from the local machine to the remote system.
`sendport`	Toggle the use of PORT commands.
`site`	Specify a command as an argument and send it to the FTP server as a SITE command.
`size`	Display the size of a file on the remote system.
`status`	Show the status of the FTP interface.
`struct`	Set the file transfer structure.
`sunique`	Toggle storing files on the remote system with unique filenames.
`system`	Show the operating system on the remote system.
`tenex`	Set the file transfer type to allow connecting to TENEX machines.
`tick`	Toggle printing byte counter during transfers.
`trace`	Toggle packet tracing.
`type`	Set a file transfer type.
umask	Set a default permissions mask for the local machine.
`user`	Provide username and password for the remote FTP server.
`verbose`	Toggle verbose output.

How to Use ftp Command in Linux

The `ftp` command connects a computer system to a remote server using the FTP protocol. Once connected, it also lets users transfer files between the local machine and the remote system, and manage files and directories on the remote system.

Establish an FTP Connection

To establish an FTP connection to a remote system, use the `ftp` command with the remote system's IP address:

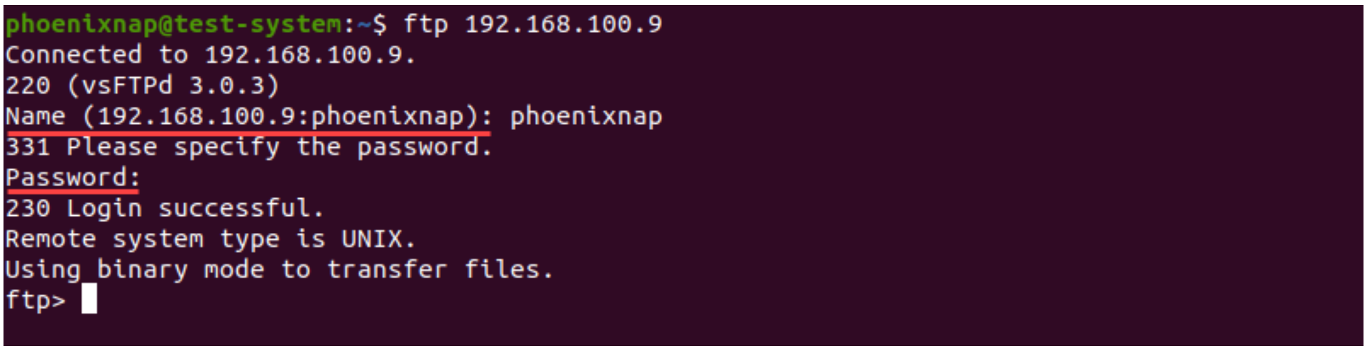
```
ftp [IP]
```

For instance, connecting to a remote server with the IP address *192.168.100.9*:

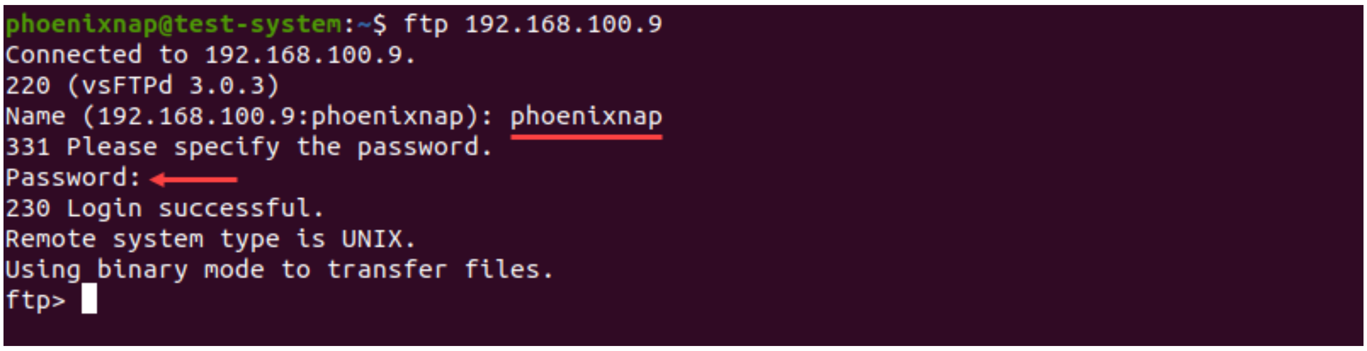
```
ftp 192.168.100.9
```

Log into the FTP Server

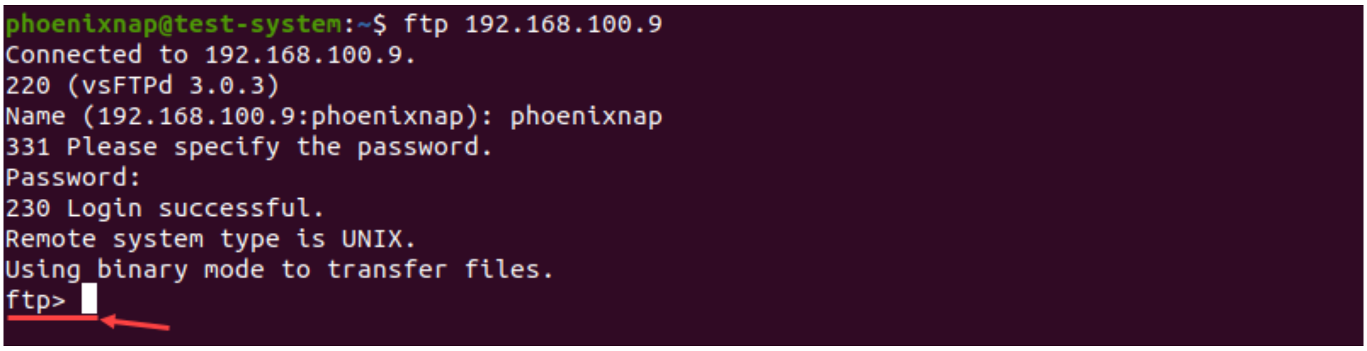
Once you initiate a connection to a remote system using the `ftp` command, the FTP interface requires you to enter a username and password to log in:


```
phoenixnap@test-system:~$ ftp 192.168.100.9
Connected to 192.168.100.9.
220 (vsFTPd 3.0.3)
Name (192.168.100.9:phoenixnap): phoenixnap
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> 
```

Entering the required credentials logs you in and starts the FTP interface. In this example, we are logging in as the *phoenixnap* user:


```
phoenixnap@test-system:~$ ftp 192.168.100.9
Connected to 192.168.100.9.
220 (vsFTPd 3.0.3)
Name (192.168.100.9:phoenixnap): phoenixnap
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> 
```

The FTP interface is now active and ready to execute commands:


```
phoenixnap@test-system:~$ ftp 192.168.100.9
Connected to 192.168.100.9.
220 (vsFTPd 3.0.3)
Name (192.168.100.9:phoenixnap): phoenixnap
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> 
```

Working with Directories on a Remote System

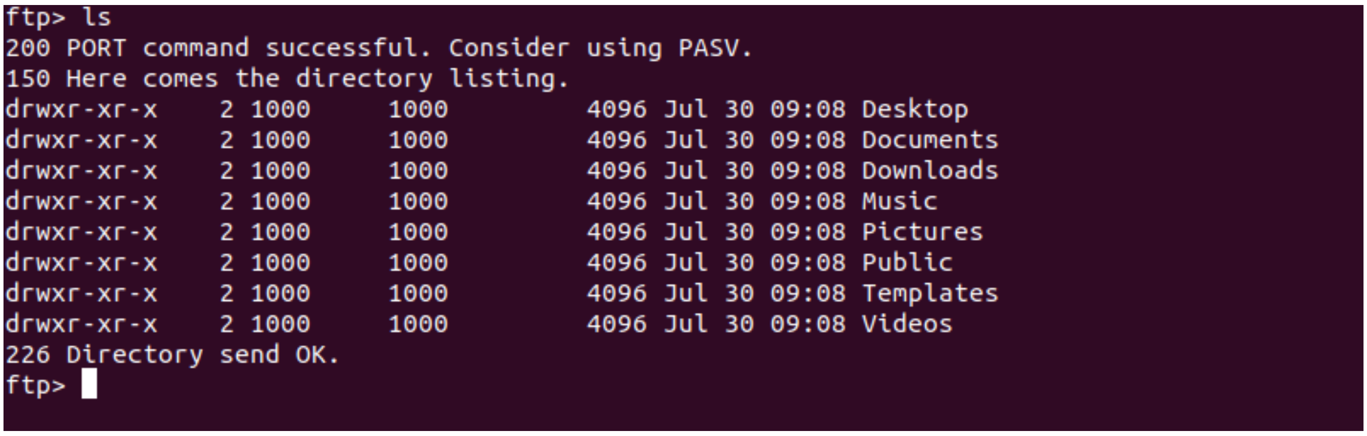
Using FTP, you can perform basic directory management on the remote system, such as creating directories, moving from one working directory to another, and listing directory contents.

List Directories

The FTP interface allows you to list the contents of a directory on a remote system using the `ls`command:

```
ls
```

Using the command without any options displays the content of the remote system's current working directory. In this example, that is the *Home* directory:


```
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxr-xr-x    2 1000     1000         4096 Jul 30 09:08 Desktop
drwxr-xr-x    2 1000     1000         4096 Jul 30 09:08 Documents
drwxr-xr-x    2 1000     1000         4096 Jul 30 09:08 Downloads
drwxr-xr-x    2 1000     1000         4096 Jul 30 09:08 Music
drwxr-xr-x    2 1000     1000         4096 Jul 30 09:08 Pictures
drwxr-xr-x    2 1000     1000         4096 Jul 30 09:08 Public
drwxr-xr-x    2 1000     1000         4096 Jul 30 09:08 Templates
drwxr-xr-x    2 1000     1000         4096 Jul 30 09:08 Videos
226 Directory send OK.
ftp> 
```

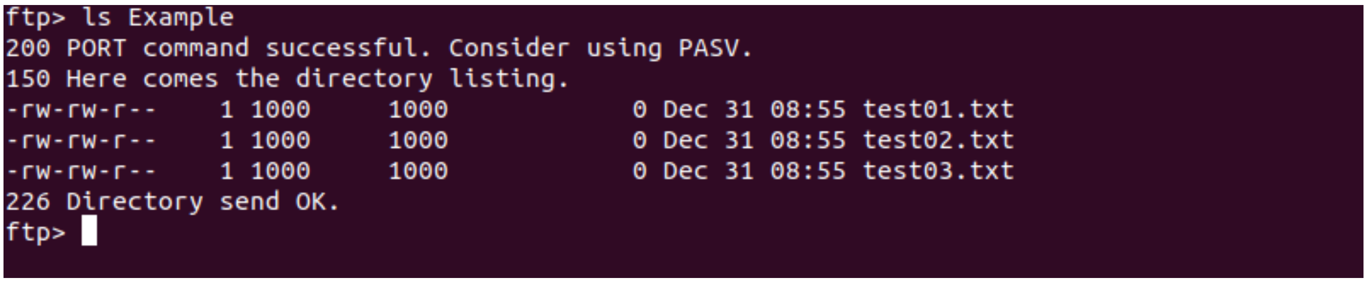
Note: The FTP interface allows you to use standard `ls` command options. Learn more in our guide to the Linux ls command.

Specifying the path to a directory as an argument to the `ls` command displays the content of that directory:

```
ls [path to directory]
```

For example, listing the content of the *Example* directory:

```
ls Example
```


```
ftp> ls Example
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-rw-r--    1 1000     1000            0 Dec 31 08:55 test01.txt
-rw-rw-r--    1 1000     1000            0 Dec 31 08:55 test02.txt
-rw-rw-r--    1 1000     1000            0 Dec 31 08:55 test03.txt
226 Directory send OK.
ftp> 
```

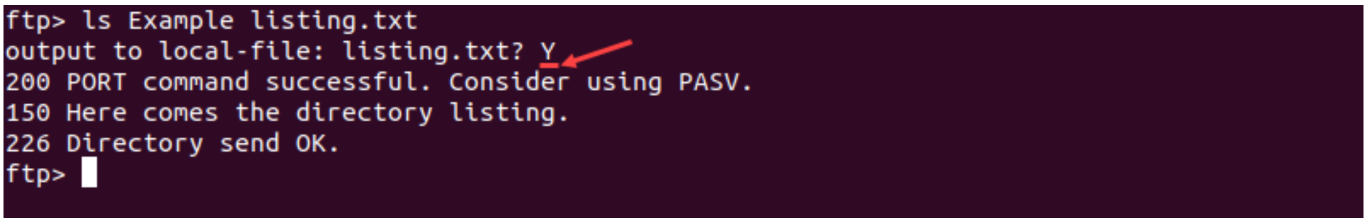
Appending the name of a text file to the end of the `ls` command saves the content of a directory to that file:

```
ls [path to directory] [file name]
```

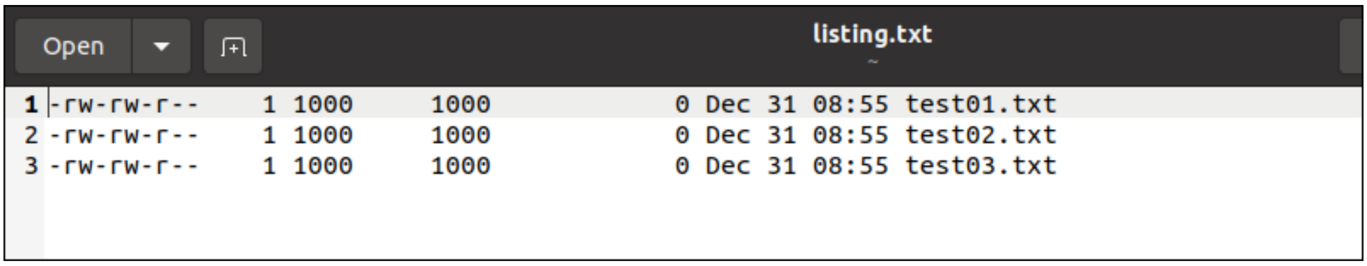
For instance:

```
ls Example listing.txt
```

This command syntax requires you to type `Y` and press **Enter** to confirm saving the text file:


```
ftp> ls Example listing.txt
output to local-file: listing.txt? Y
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
226 Directory send OK.
ftp> 
```

Opening the text file reveals the content of the directory:



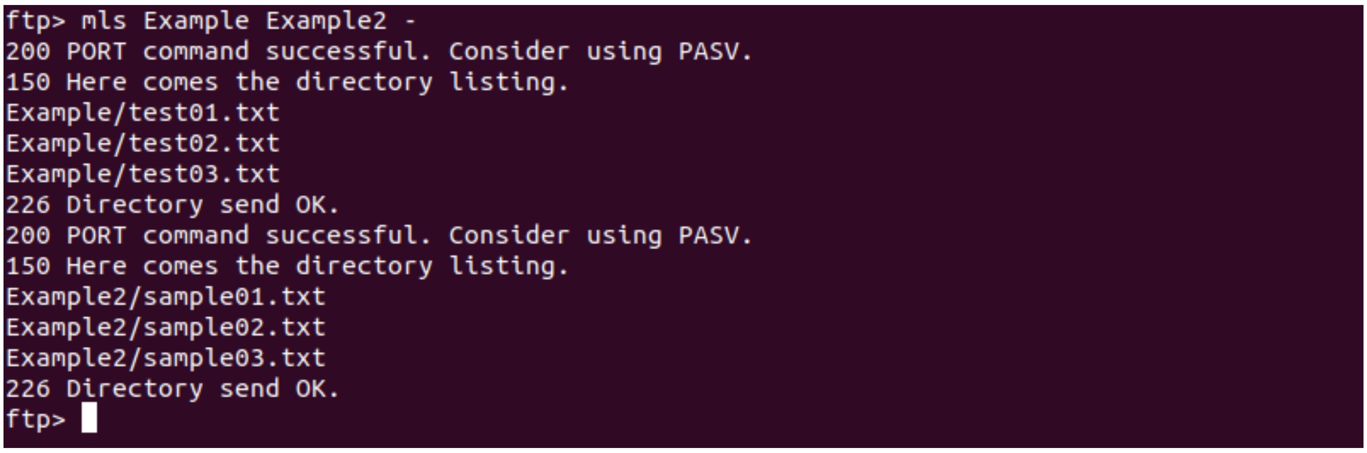
```
-rw-rw-r--    1 1000      1000            0 Dec 31 08:55 test01.txt
-rw-rw-r--    1 1000      1000            0 Dec 31 08:55 test02.txt
-rw-rw-r--    1 1000      1000            0 Dec 31 08:55 test03.txt
```

The **`dir`** and **`nlist`** commands are alternatives to the **`ls`** command and work the same way. The FTP interface also allows you to list the contents of multiple directories using the **`mls`** command:

```
mls [directory 1] [directory 2] .. [directory n]
```

For instance, the example below lists the contents of *Example* and *Example2*:

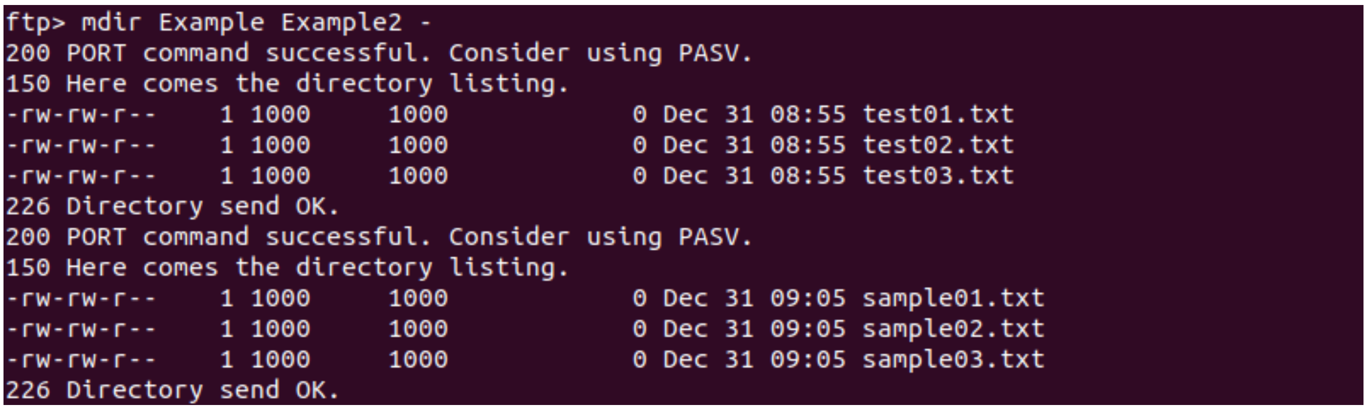
```
mls Example Example2 -
```



```
ftp> mls Example Example2 -
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
Example/test01.txt
Example/test02.txt
Example/test03.txt
226 Directory send OK.
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
Example2/sample01.txt
Example2/sample02.txt
Example2/sample03.txt
226 Directory send OK.
ftp> 
```

Like the **`ls`** command, the **`mls`** command allows users to save the contents in a text file. This command treats the last argument as the name of the text file. If you want to list directory contents without saving them to a text file, replace the file name with a dash symbol (**`-`**).
The **`mdir`** command works the same as the **`mls`** command but offers a more detailed output:

```
mdir Example Example2 -
```



```
ftp> mdir Example Example2 -
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-rw-r--    1 1000     1000            0 Dec 31 08:55 test01.txt
-rw-rw-r--    1 1000     1000            0 Dec 31 08:55 test02.txt
-rw-rw-r--    1 1000     1000            0 Dec 31 08:55 test03.txt
226 Directory send OK.
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-rw-r--    1 1000     1000            0 Dec 31 09:05 sample01.txt
-rw-rw-r--    1 1000     1000            0 Dec 31 09:05 sample02.txt
-rw-rw-r--    1 1000     1000            0 Dec 31 09:05 sample03.txt
226 Directory send OK.
```

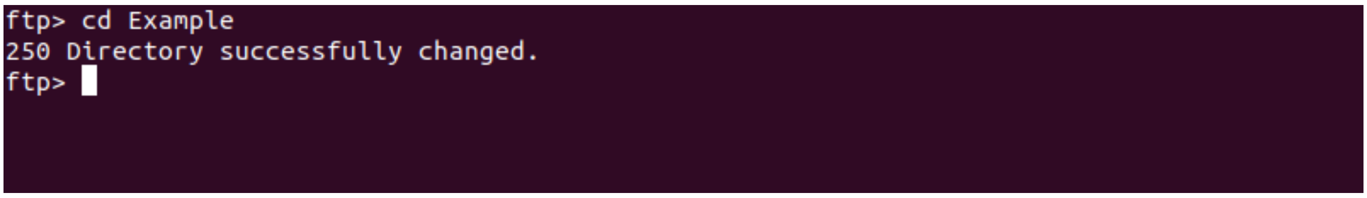
Change Directories

Use the `cd` command to change the current working directory on the remote system:

```
cd [path to directory]
```

For instance, moving to the *Example* directory:

```
cd Example
```

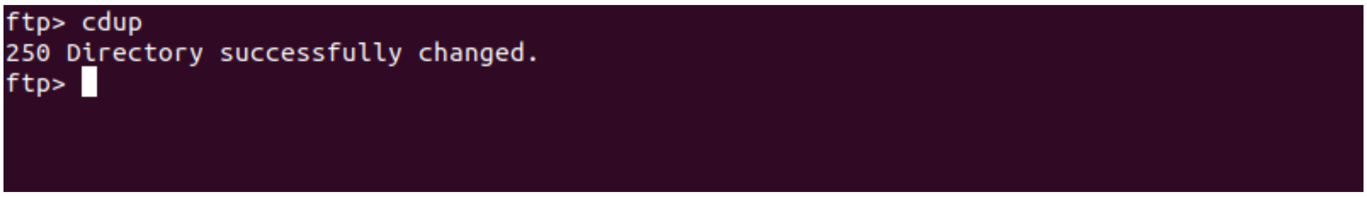


```
ftp> cd Example
250 Directory successfully changed.
ftp>
```

Note: Learn more about using the cd command to change the directory in Linux.

Use the `cdup` command to move to the parent of the current working directory. In this example, we are moving from the *Example* directory to the *Home* directory:

```
cdup
```



```
ftp> cdup
250 Directory successfully changed.
ftp>
```

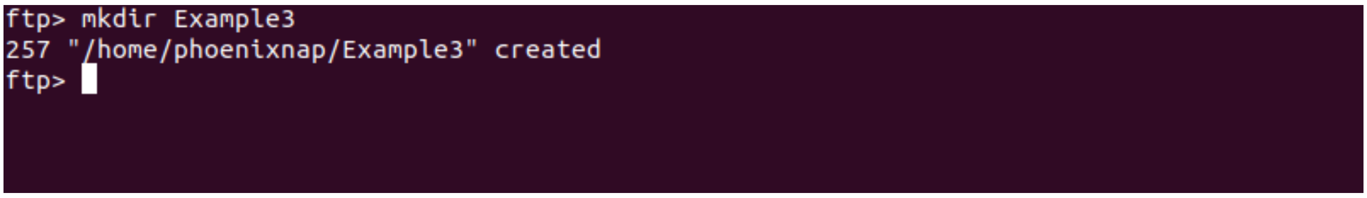
Create Directories

Using the `mkdir` command allows you to create a directory on the remote system:

```
mkdir [directory name]
```

In the example below, we create a directory named *Example3*:

```
mkdir Example3
```



```
ftp> mkdir Example3
257 "/home/phoenixnap/Example3" created
ftp> 
```

Download Files via FTP

To transfer a file from a remote system to the local machine, use the **`get`** or **`recv`** command.

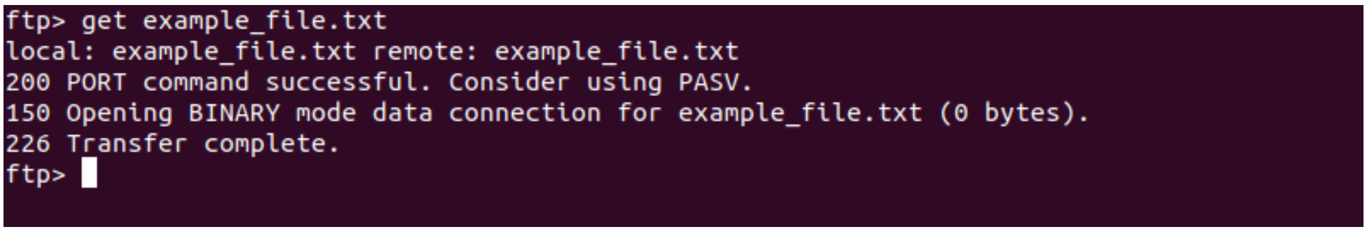
```
get [remote file name]
```

OR

```
recv [remote file name]
```

In the example below, we transfer *example_file.txt* to the local machine.

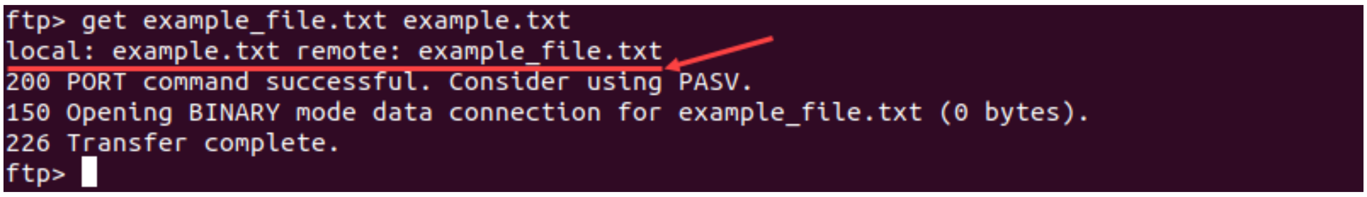
```
get example_file.txt
```



```
ftp> get example_file.txt
local: example_file.txt remote: example_file.txt
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for example_file.txt (0 bytes).
226 Transfer complete.
ftp> 
```

To transfer *example_file.txt* and save it as *example.txt* on the local machine, use:

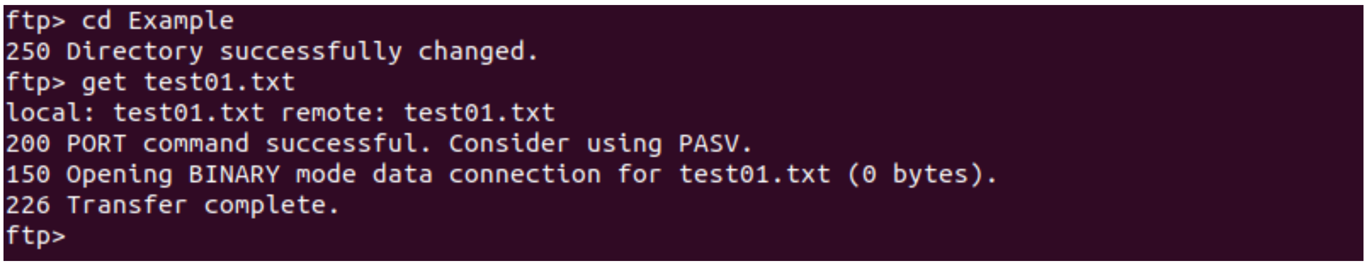
```
get example_file.txt example.txt
```



```
ftp> get example_file.txt example.txt
local: example.txt remote: example_file.txt
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for example_file.txt (0 bytes).
226 Transfer complete.
ftp> 
```

Transferring a file from a specific directory requires you to move into that directory:

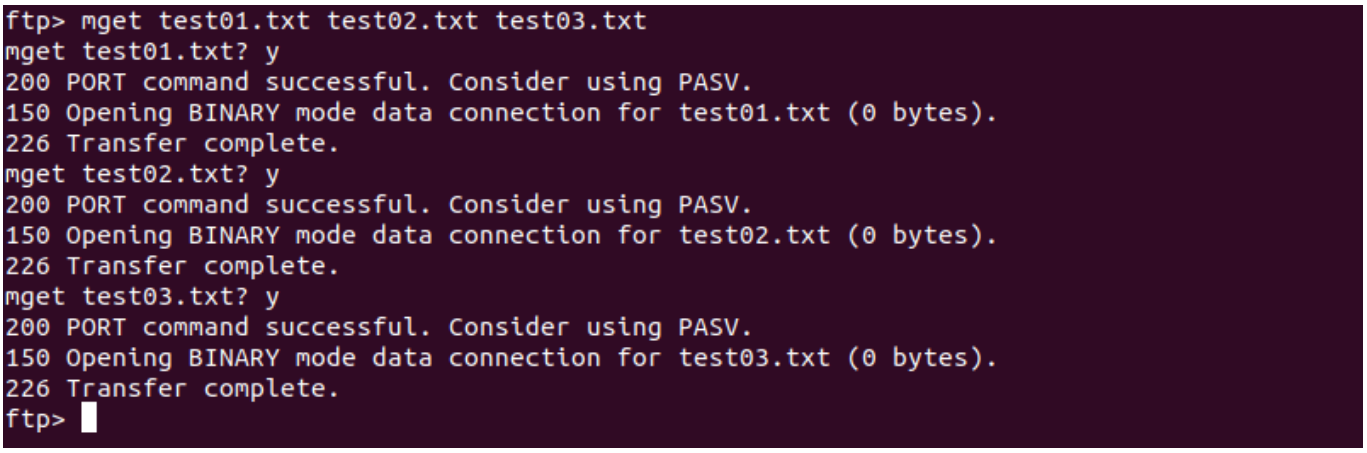
```
cd Example

get test01.txt
```



```
ftp> cd Example
250 Directory successfully changed.
ftp> get test01.txt
local: test01.txt remote: test01.txt
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for test01.txt (0 bytes).
226 Transfer complete.
ftp>
```

The **`mget`** command allows you to transfer multiple files at the same time. For example, transferring *test01.txt*, *test02.txt*, and *test03.txt* from the *Example* directory:

```
mget test01.txt test02.txt test03.txt
```



```
ftp> mget test01.txt test02.txt test03.txt
mget test01.txt? y
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for test01.txt (0 bytes).
226 Transfer complete.
mget test02.txt? y
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for test02.txt (0 bytes).
226 Transfer complete.
mget test03.txt? y
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for test03.txt (0 bytes).
226 Transfer complete.
ftp>
```

Note: By default, the **`mget`** command displays an interactive prompt asking users to confirm each file transfer. Use the **`prompt`** command to toggle this feature on and off.

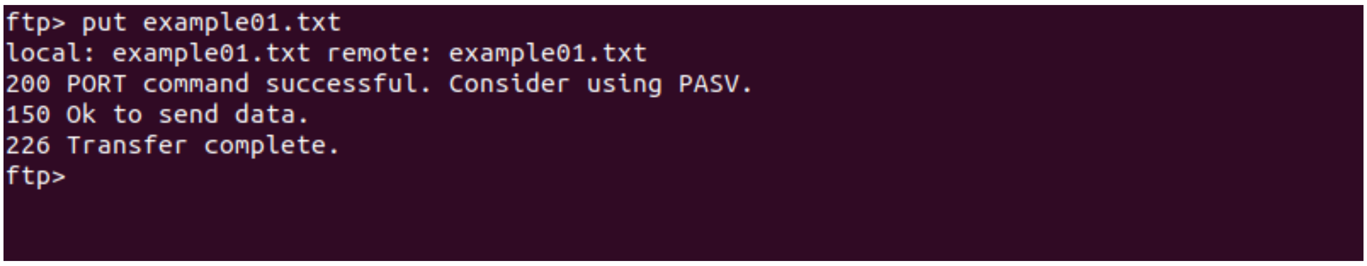
Upload Files via FTP

Use the **`put`** or **`send`** command to transfer a file from the local machine to a remote system. Both commands use the same basic syntax:

```
put [local file name]

send [local file name]
```

To transfer *example01.txt* to the remote system, use:

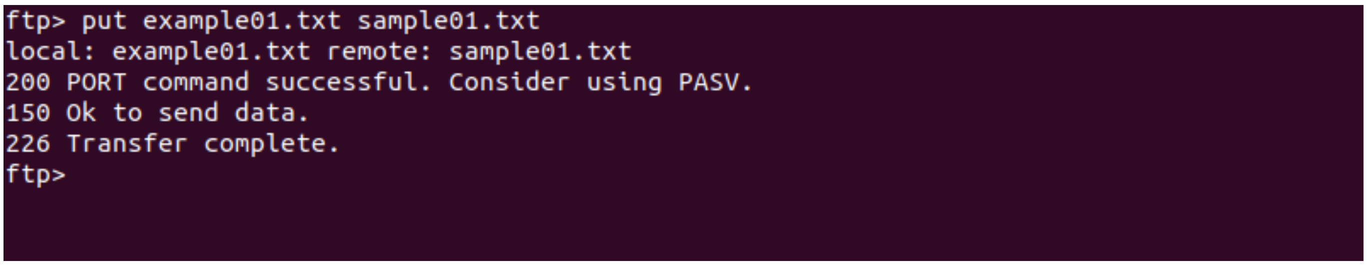
```
put example01.txt
```



```
ftp> put example01.txt
local: example01.txt remote: example01.txt
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
ftp>
```

To upload *example01.txt* to the remote system as *sample01.txt*, use:

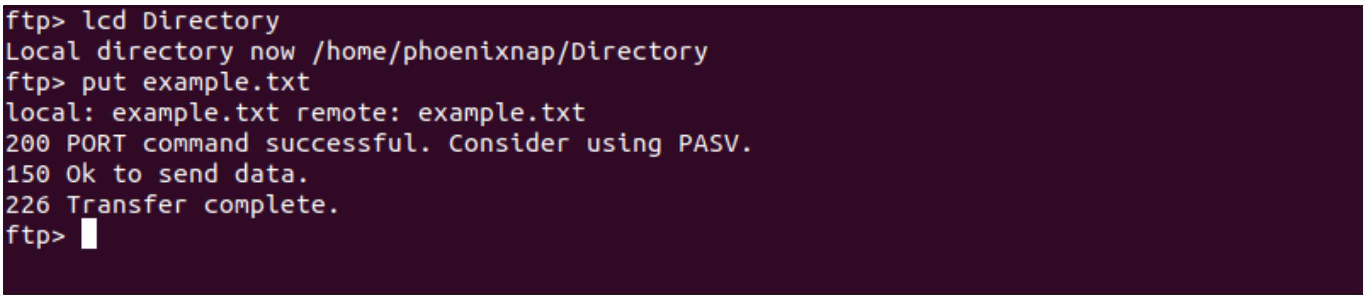
```
put example01.txt sample01.txt
```



```
ftp> put example01.txt sample01.txt
local: example01.txt remote: sample01.txt
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
ftp>
```

Moving into a specific directory allows you to transfer files from that directory:

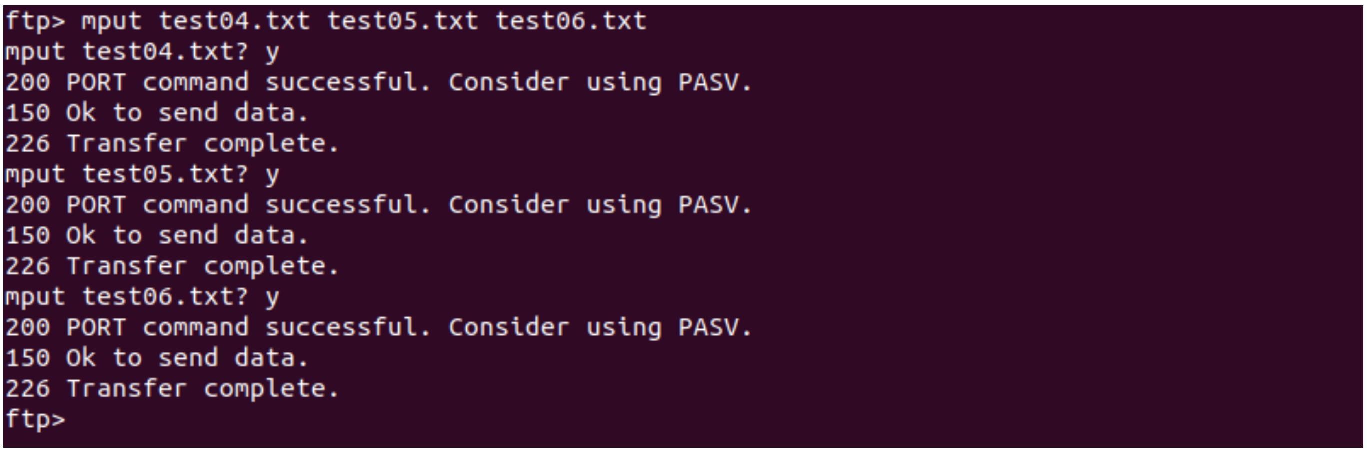
```
cd Directory

put example.txt
```



```
ftp> lcd Directory
Local directory now /home/phoenixnap/Directory
ftp> put example.txt
local: example.txt remote: example.txt
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
ftp>
```

Use the `mput` command to transfer multiple files to the remote system. For example, transfer *test04.txt*, *test05.txt*, and *test06.txt* with:

```
mput test04.txt test05.txt test06.txt
```



```
ftp> mput test04.txt test05.txt test06.txt
mput test04.txt? y
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
mput test05.txt? y
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
mput test06.txt? y
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
ftp>
```

Note: Using the `mput` command creates an interactive prompt asking the user to confirm each file transfer. Use the `prompt` command to toggle this feature on and off.

Rename Files

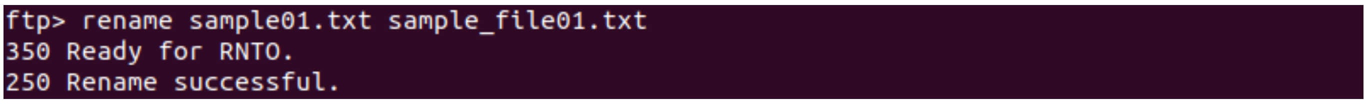
Use the `rename` command to rename files on the remote server. The `rename` command uses the following syntax:

```
rename [old file name] [new file name]
```

For instance, renaming *sample01.txt* to *sample_file01.txt*:

```
rename sample01.txt sample_file01.txt
```

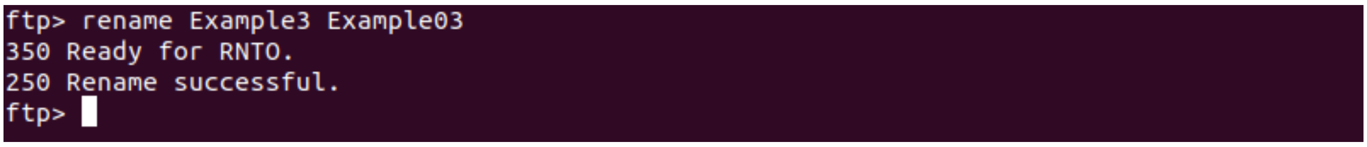
Executing the command successfully produces the following output:



```
ftp> rename sample01.txt sample_file01.txt
350 Ready for RNTO.
250 Rename successful.
```

Use the `rename` command to change directory names too.
In the example below, the *Example3* directory is renamed to *Example03*:

```
rename Example3 Example03
```



```
ftp> rename Example3 Example03
350 Ready for RNTO.
250 Rename successful.
ftp>
```

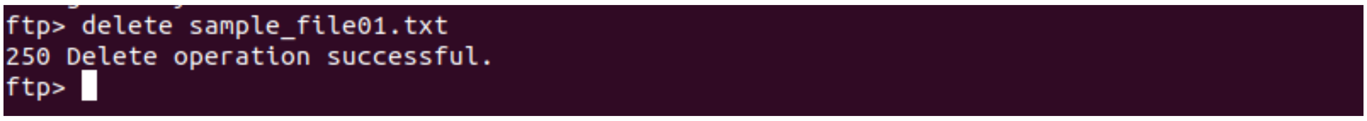
Delete Files

The `delete` command allows you to delete a file on the remote system. It uses the following syntax:

```
delete [file name]
```

For instance, deleting *sample_file01.txt*:

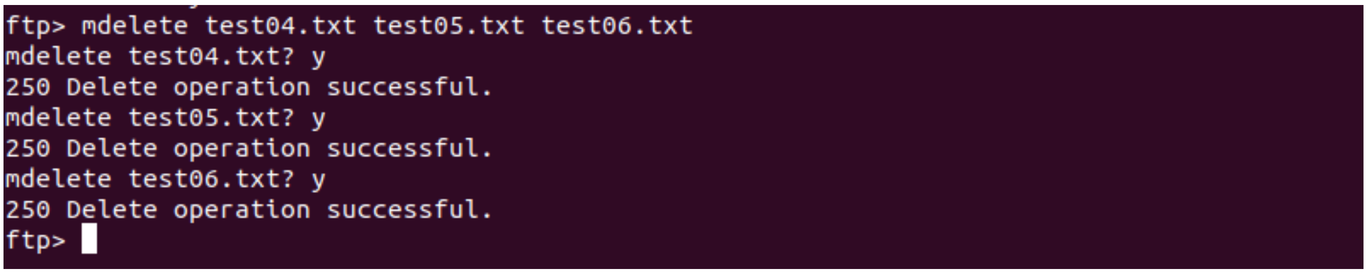
```
delete sample_file01.txt
```



```
ftp> delete sample_file01.txt
250 Delete operation successful.
ftp>
```

Using the `mdelete` command allows you to delete multiple files at the same time by adding the file names after the command:

```
mdelete test04.txt test05.txt test06.txt
```

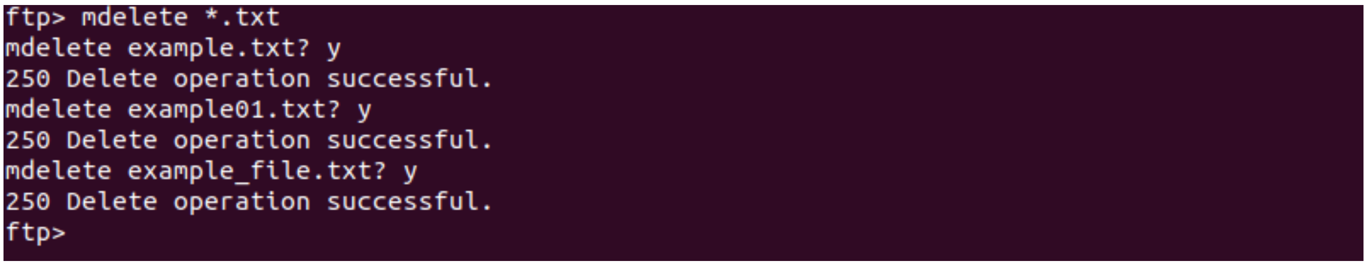


```
ftp> mdelete test04.txt test05.txt test06.txt
mdelete test04.txt? y
250 Delete operation successful.
mdelete test05.txt? y
250 Delete operation successful.
mdelete test06.txt? y
250 Delete operation successful.
ftp>
```

Note: When using the `mdelete` command, an interactive prompt asks you to confirm each file deletion. Use the `prompt` command to toggle this feature on and off.

Another method is to use the `mdelete` command with a wildcard character. For instance, to delete all *.txt* files, use:

```
mdelete *.txt
```

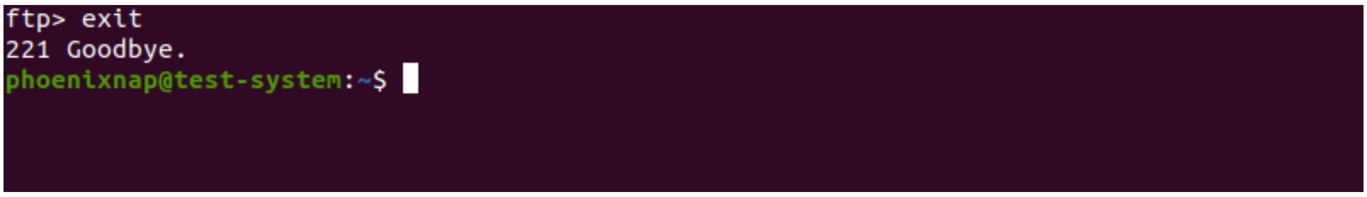


```
ftp> mdelete *.txt
mdelete example.txt? y
250 Delete operation successful.
mdelete example01.txt? y
250 Delete operation successful.
mdelete example_file.txt? y
250 Delete operation successful.
ftp>
```

Close the FTP Connection

Use the `bye`, `exit`, or `quit` command to terminate the FTP connection and exit the interface.

Using the `disconnect` command closes the connection without exiting the interface.



```
ftp> exit
221 Goodbye.
phoenixnap@test-system:~$
```

Conclusion

After reading this article, you should be able to establish an FTP connection between a local system and a remote server, and use it to transfer files and perform basic file and directory management.