

FIT 100: Lab 3

Files and Directory Structures II

Introduction

This part of lab requires you to work within your Dante file space using a command line environment and the **Tera Term** application. Lab 1 covered SFTP (secure file transfer), creating and moving directories, and understanding where we were at all times when moving and saving files. The SFTP client you used, **SSH Secure Shell**, has a Graphical User Interface (**GUI**). In a GUI you manipulate images or icons and click on buttons to use the software. A command line interface requires you to know and use a fixed set of commands in order to manipulate files and folders and to move them from location to location. Your maneuverability in this environment depends mostly on the keyboard, not on the mouse. Lab 3 explores the ways to manipulate and move files using UNIX commands. You will be doing similar tasks to the last lab, but with a different interface and you will work completely within your remote account on Dante using Tera Term.

Reading

Read the entire lab and the [Reference](#) section on directories, domains and file types.

Objectives

- To explore the difference between a command line and a GUI interface.
- To become comfortable with using the command-line prompt to navigate directory structures and access directories stored in various locations on a remote computer.
- To understand that there are many ways to access remote directories.

What we know

Your User ID and Password for your various accounts at the UW
A Directory is also called a Folder

To Do

1. Create a Directory (folder) remotely using Unix commands
2. Using PICO as a text editor
3. Moving from here to there in one step
4. Moving and copying files within the remote account

To Submit

To check out of this lab, be able to show your TA the directory you made and give a single command to move anywhere within your remote account.

Details

1. Create a Directory (folder) remotely using Unix commands

One way to move files between directories inside your account on Dante is by using a command language. To use this language, you need to be in a particular area of your remote account using

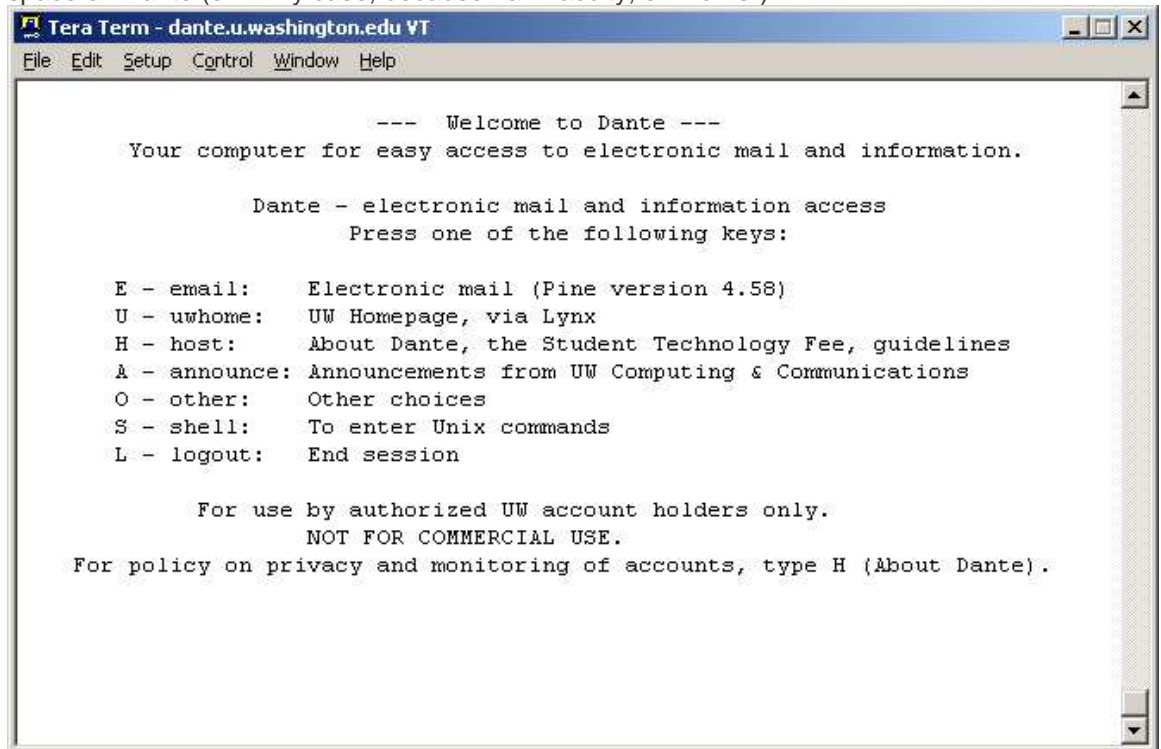
a particular piece of software. We will use the Unix Shell command line interface within the Tera Term software.

- 1.1. Find the icon for **Tera Term (Dante)** on the desktop in front of you. If the icon is not on desktop, click **Start -> Programs -> UWICK Applications -> Student Email (Dante)**.



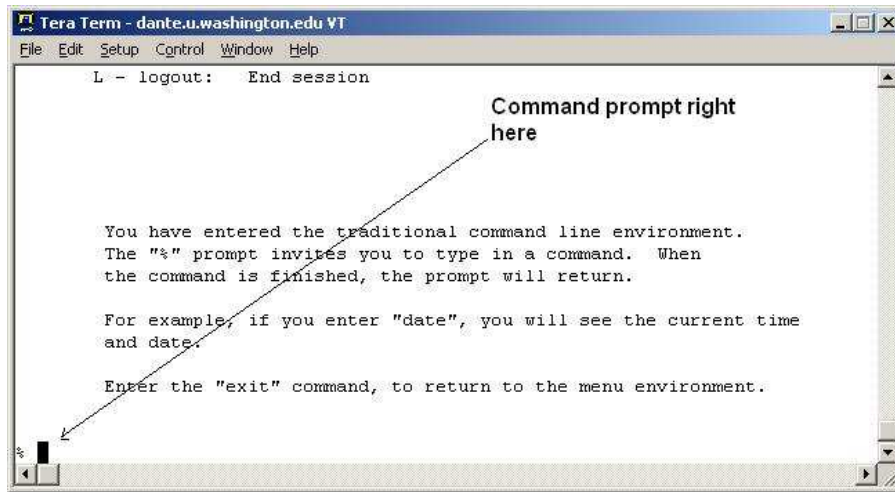
Tera Term is also listed in the Start button menu in the lower left.

- 1.2. Log in with your UW Net ID and password
- 1.3. After you log in, you will be taken to the main entry point into email, web or file server space on Dante (or in my case, because I am faculty, on Homer)

The image is a screenshot of a Tera Term window. The title bar reads "Tera Term - dante.u.washington.edu VT". The menu bar includes "File", "Edit", "Setup", "Control", "Window", and "Help". The main window area displays a text-based welcome screen for Dante. The text is as follows:

```
--- Welcome to Dante ---  
Your computer for easy access to electronic mail and information.  
  
Dante - electronic mail and information access  
Press one of the following keys:  
  
E - email:    Electronic mail (Pine version 4.58)  
U - uwhome:   UW Homepage, via Lynx  
H - host:     About Dante, the Student Technology Fee, guidelines  
A - announce: Announcements from UW Computing & Communications  
O - other:    Other choices  
S - shell:    To enter Unix commands  
L - logout:   End session  
  
For use by authorized UW account holders only.  
NOT FOR COMMERCIAL USE.  
For policy on privacy and monitoring of accounts, type H (About Dante).
```

- 1.4. Press <S> for the Unix shell.

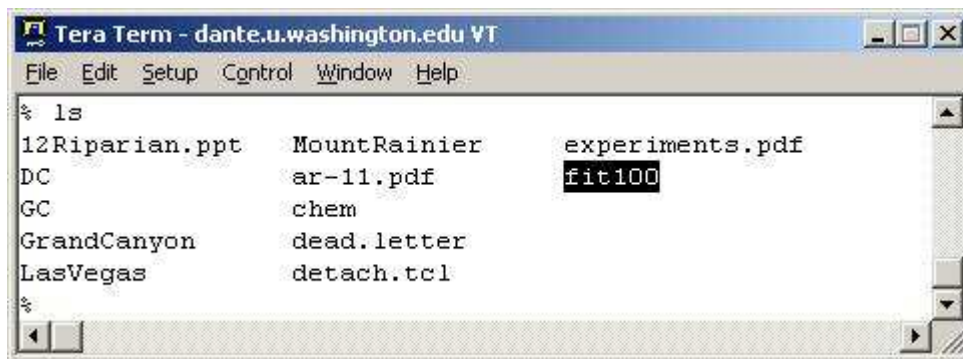


You are now located in the root (home) directory of your Dante account. The name of that directory is your UW Net ID.

In Lab 1, you created a folder called **labs** inside of a folder called **fit100**. Even though that was done using a different software program and interface, you can see those folders here because you are looking into the same file space. To see if fit100 is there, list the contents of the current directory in the remote account:

- 1.5. Enter **ls** and press the Enter key. **ls** is the UNIX command to list the contents of the directory

Note that your result will be slightly different because you have different contents in your directory

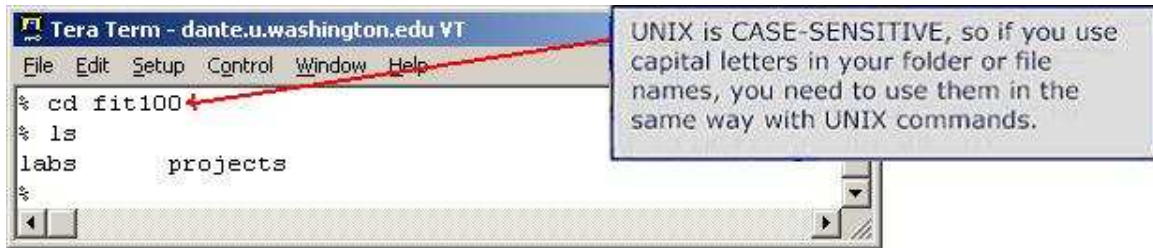


- 1.6. Change to the directory that will hold future lab work. You will need to go first to the **fit100** directory and then go to **labs** directory. Use the Unix **cd** command to change directories

```
cd fit100
```

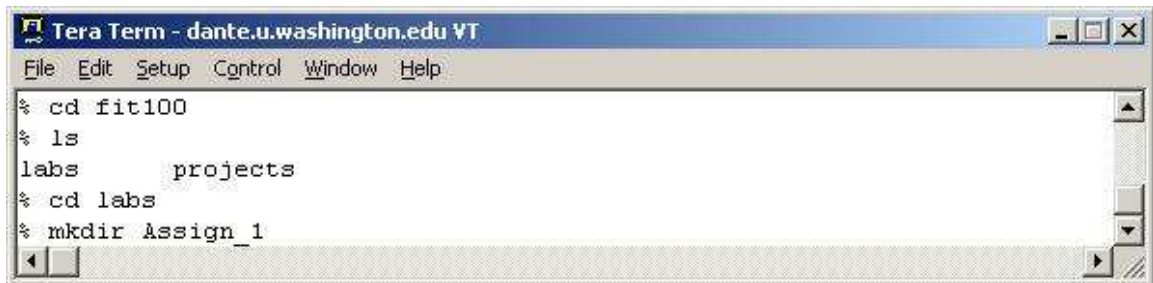
NOtice that all Unix commands are listed in the reference section of this document.

- 1.7. Use your **ls** command again to see the folders that are inside of your fit100 folder/directory.

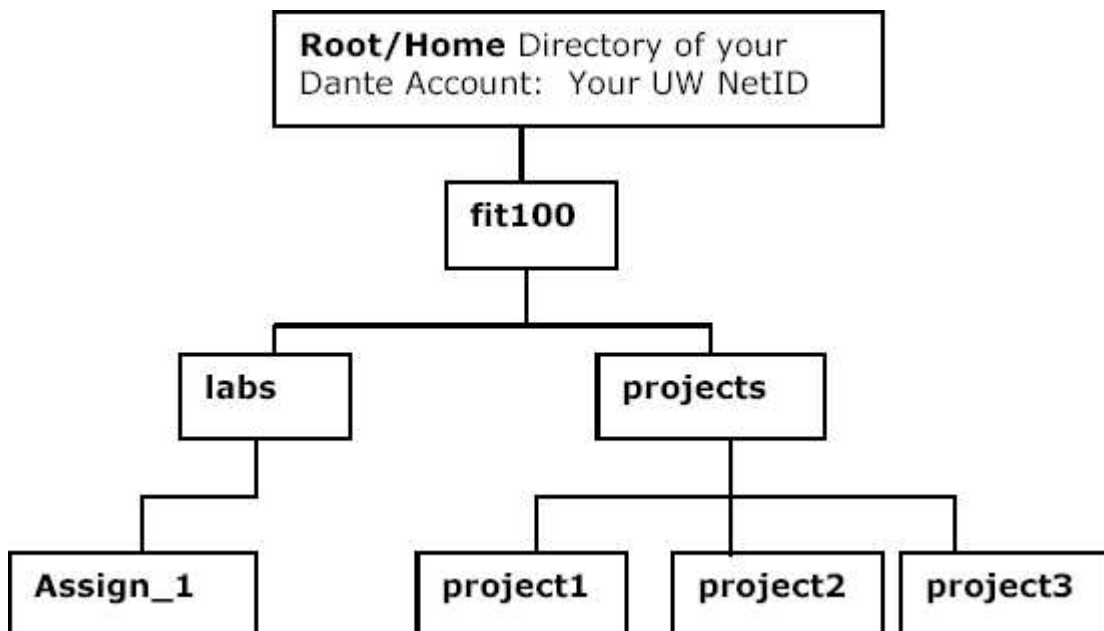


1.8. Change directories (**cd**) again to your **labs** directory.
cd labs

1.9. Create a directory called **Assign_1** in **labs** with Unix's make directory command:
mkdir Assign_1



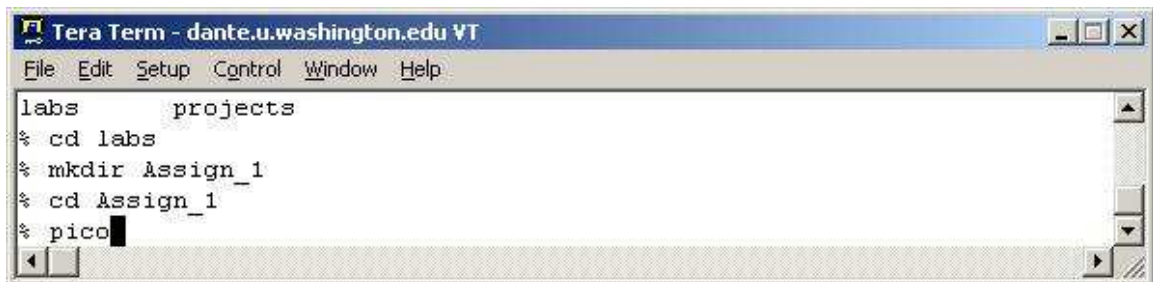
You've created a directory named **Assign_1** inside **labs** which sits inside of **fit100** which sits at the root of your Dante account. The structure of your folders for FIT100 in your remote account should be as follows:



1.10. Using the change directory command again go into the **Assign_1** directory by typing **cd Assign_1**. We will now create and save a text file within this folder.

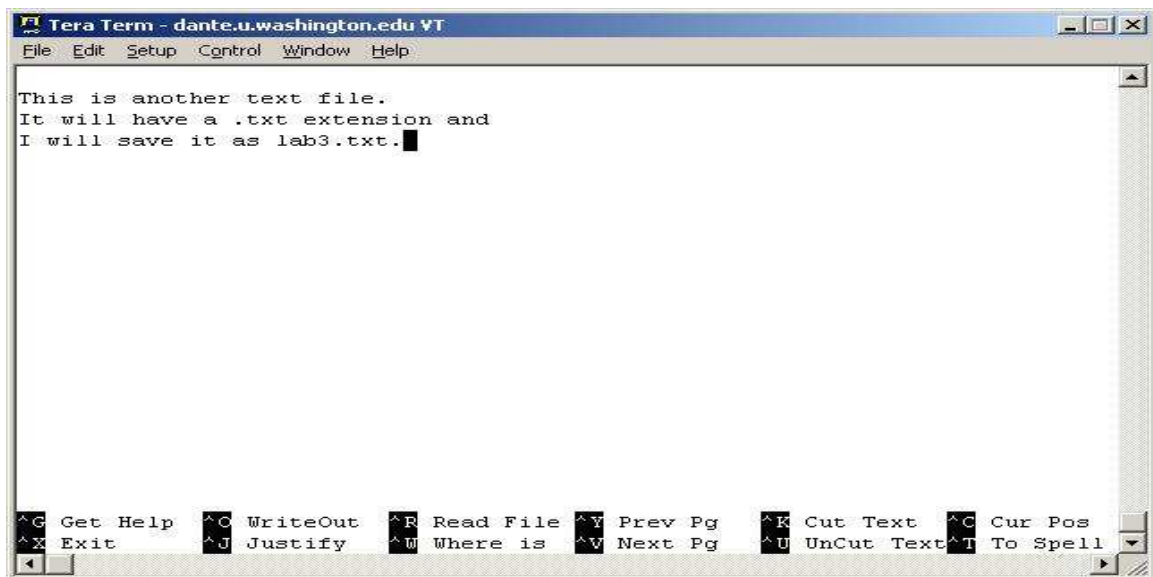
2. Using Pico as a text editor

2.1. The Unix shell has a text editor that you can use (it is just like using Notepad). Call up the editor by typing **pico** in the command line and pressing the Enter key.



```
Tera Term - dante.u.washington.edu VT
File Edit Setup Control Window Help
labs    projects
% cd labs
% mkdir Assign_1
% cd Assign_1
% pico
```

2.2. Enter in the following text:



```
Tera Term - dante.u.washington.edu VT
File Edit Setup Control Window Help
This is another text file.
It will have a .txt extension and
I will save it as lab3.txt.
```

^G Get Help ^O WriteOut ^R Read File ^Y Prev Pg ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where is ^V Next Pg ^U UnCut Text ^T To Spell

2.3. Save the file as lab3.txt

2.3.1. Use **^O** (^ stands for the Control key) to WriteOut the contents of the text file to a space on the hard disk. This is the Unix way of **SAVING** a file!

2.3.1.1. *Save modified buffer? Yes*

2.3.1.2. *File Name to write: lab3.txt*

This environment doesn't easily recognize spaces in file names. It is best not to include spaces in your file or folder names anyway.

2.3.2. Press the Enter key to save the file as lab3.txt

Because you were located in the Assign_1 directory when you opened Pico and created the file, the default location for saving the file is to the current directory.

2.4. Exit the Pico text editor by pressing **^X** (Ctrl + x)

2.5. From the command prompts, list the contents of the folder **Assign_1**. using Unix's **ls** There should now be a file called **lab3.txt**.



```
Tera Term - dante.u.washington.edu VT
File Edit Setup Control Window Help
% ls
lab3.txt
%
```

2.6. Go back into the file and edit it. From the command prompt, call the Pico editor and give the file name to open:

```
pico lab3.txt
```

2.7. When you have changed a few words in the file, save it again and exit.

3. Moving from here to there in one step

A benefit of using the command line interface is that you can go from the root of your directory structure all the way to the deepest folder with one command if you choose. Up until now, you have moved up and down your account one level at a time using the **cd** command. Now we will move from inside the **Assign_1** folder and go into one of our Project folders in one step. If I want to move from the **Assign_1** folder location to the **projects** folder that is sitting inside of **fit100**, I could enter either one of the following commands:

```
cd ~/fit100/projects
```

or

```
cd ../../projects
```

Read the Reference section below on *Absolute and Relative Paths* and see if you can figure out why both commands work.

3.1. Use the **cd** command to move from your current location in folder **Assign_1** to the new location of the **projects** folder.

3.1.1. To see if you successfully changed to the **projects** folder, type **ls**. Are the 3 folders that you created inside of projects listed?

3.2. Now, change folders again to go back to **Assign_1**

4. Moving and copying files from one directory to another

4.1. Look at the Unix commands at the end of this assignment.

4.1.1. Which command will allow you to MOVE a file to another location?

4.1.2. Which command will allow you to COPY a file to another location?

4.2. Copy the **lab2.txt** file in **Assign_1** to your **project2** folder

4.3. Change directories over to the project2 folder and list the contents of project2 to see if lab2.txt is there.

4.4. Move **lab2.txt** from **project2** folder to the **project1** folder

4.5. Now list the contents of project2 again. Is the lab2.txt file gone from project2?

4.6. Go to the project1 folder. Is the lab2.txt file there?

4.7. Log off the machine before leaving.

Congratulations!

You have made directories and moved files using the command line interface!! You will encounter this alternative to GUIs occasionally, and now you understand how it works. You can now create folders and move content between folders in your account on Dante using the command line. This is a very helpful resource when you need to be able to access and work on documents at a variety of locations but may not have a disk to transport the work. You want to become comfortable in the environment introduced in this lab. It will save you headaches in the weeks to come!!!

HELPFUL HINTS: When you get lost

As we go through all of this new material it is very easy to get confused and often times to get lost. When that happens in the command line environment the easiest thing to do is go back to your root directory. From there you can navigate back through your directories to find the place you need to be again. If you get lost, or take a wrong turn as we go through this lab, simply get yourself back to the home directory, the root, of your account with the following step:

Type in the change directory command with no directory listed and press <enter>:

```
cd
```

You are now in your root directory.

To move up just one directory at a time, use **cd ..**

Reference

Common Unix Commands

[See more commands at: <http://www.washington.edu/computing/unix/unixqr.html>]

Exit the shell back to the main Tera Term menu:

exit
logout

Change directories (folders):

cd *path*

The path can be either relative or absolute. To find out what each of these means, read the section below on Paths.

cd

This command by itself will take you to your root directory

cd ..

Command to move back one level at a time ... those are dot, dots

List contents of directory / directories:

ls

Create a new directory (folder) on the remote computer:

mkdir *path*

Make a directory on the remote machine. The path can be relative or absolute.

Moving and renaming files:

mv *file directory*

Moves a file to a different directory

mv *directory1 directory2*

Moves a directory to a second directory

mv *file1 file2*

Renames file1 as file2

mv *directory1 directory2*

Renames directory1 as directory2

mv -i *file1 file2*

Inquires before overwriting an existing file

Copying files:

cp *file directory*

Copies a file into a different directory

cp *file1 file2*

Makes a copy of a file with a different name

cp *file1 file2*

Inquires before overwriting an existing file

Delete a remote file / files:

rm *file*

Absolute versus Relative Paths when pointing (referring) to files or folders

```
A:\INFO100\MyFirstLab\myfile.txt  
C:\My Documents\My Webs\images\
```

Consider the two lines just above. Both are path names that refer to the location of a file or folder. Now look at the two lines below. They refer to the same file and folder above, but are considered relative path names.

```
myfile.txt  
images\
```

It's all LOCATION, LOCATION, LOCATION!!! An absolute path name shows the exact path to the file. Starting from the drive it is located on, right down to the name of the folder or file itself. It always gives the exact location. The second set of path names are known as relative path names. They are named relative to the current folder. The name "myfile.txt" is only useful to me if it is a file name I am looking for within the current directory. It would not be as helpful to someone just sitting down at the computer who didn't know where to start looking. It is ESSENTIAL that you understand the difference between these 2 as you work in your remote folder structure using the command line environment. [There is more explanation in the FIT textbook.] Since you do not have the option to "see" where you are at with a graphical interface, you need to use commands to show you what folder you are in and to look up or down your folder hierarchy.

Local vs. Remote Directories: "What's the difference?"

A directory, or folder, is a container for files. Just like the folders in an office file cabinet hold documents, a directory on your computer or any other, can hold files of all types. A directory is a way to organize related files in a manner that allows for quick retrieval. If you have your work divided into the various areas they pertain to, you can use folders to hold them in a logical manner. There are many ways to define **local directories**. For this lab, local directories are referred to as the folders that are located on the C drive on the PC physically in front of you.

Documents placed in folders at this location cannot be accessed at other campus computers. They are local to the machine in front of you. **Remote directories** are folders stored on remote computers (called servers) that can be accessed from various computer labs on campus and even from home, if you have a way to connect. To access remote files within folders (directories) from home, use the UWICK software kit. (UWICK) UW Internet Connectivity Kit Information <http://www.washington.edu/computing/software/uwick/>

Servers and Clients on the Web

You have probably already heard the terms "web server" and "e-mail client," but you might not realize that "server" and "client" are general terms describing roles that computers can play on a network, depending on what software they are running. In general, a *server* is a computer that provides some kind of data (e.g., web pages, database entries) or service (e.g., e-mail, printing), and a *client* is a computer that requests and receives the data or service. To illustrate this difference, we will begin by discussing an example with web pages. When you view a web page on your computer, your computer is acting as a *client* to a *web server* somewhere on the Internet, the computer where the web pages are stored. Each time you click on a link, your web browser sends a web page request through the network to the web server, and the server responds by sending a copy of the requested page back to your computer, where the browser displays it for you.

Structure of a URL

At the heart of each of these requests is a *URL* (short for Uniform Resource Locator), which is a standard way of specifying a file or resource on a particular computer on the network. Although URLs can be used to specify many kinds of network requests, since they are most commonly used for web pages, URLs are also called “web addresses” or “links.” URLs seem to appear everywhere now, from advertisements to local television news broadcasts and even boxes of breakfast cereal, as companies and other organizations set up web sites to accompany traditional media materials. Every URL includes three important pieces of information: what kind of request it is (usually for a web page), the hostname of the server the request is going to, and the location and name of the file being requested. Suppose you are trying to view the web page at this URL on your computer: http://www.pcwebopedia.com/quick_ref/servers.html

The first part of the URL is before the `://` and, in this case, `http` indicates that this is a request for a web page. *HTTP* stands for Hypertext Transfer Protocol, where “hypertext” is a technical term for text that includes links to other documents, and is the standard method of transferring web files through the Internet. One useful way of thinking about the last two parts of the URL is to interpret them together as a pathname for a web page file. One important difference between URLs and pathnames is that a URL must specify which computer the file is on, something which is just assumed in the case of a pathname. The hostname is specified between the `://` and the next `/`, and the remainder of the URL is just like a pathname—it specifies the location of the file on the given host, with slashes separating folder names. In some cases, the filename at the end of the URL can be omitted (e.g., `http://nature.org`), and the web server assumes that a file with a standard name like `index.html` or `home.html` is being requested.

To view the page at the URL given above, your web browser sends a web page (HTTP) request to the host `www.pcwebopedia.com`. If this computer is properly set up as a web server, it is running software that listens on the network for these requests and will respond by sending back the appropriate web page—in this case, `servers.html` in the `quick_ref` folder. A single computer can act as more than one kind of server by running more than one kind of server software at once. In fact, it is common for most hosts to be playing the role of at least a few different servers, e.g., web, mail, printing, and file storage. In the next part of the lab, we will see how another kind of server can be used to copy files between computers over the network.

Is the server the computer or the software?

The term “web server” can be confusing, because it is often used in two different ways. The term commonly refers to a computer that is running software that enables it to send web pages on request, as in, “My home page is on the web server `students.washington.edu`.” However, some people also use the term “web server” to refer to this software, rather than the computer, as in, “If you’re running a Microsoft web server on your computer, you should regularly check for security problems.” Context usually disambiguates, but not in some common cases. In this lab, we use the term in the sense of a computer or a role that it plays, rather than the software, which we call “server software.”






Copying Files Across the Network with SFTP

SFTP (short for Secure File Transfer Protocol) is a standard method of sending files between computers through a network. The primary difference between SFTP and HTTP is that using SFTP requires that you identify yourself with a *user name* (also known as a *login*) and a *password*. (Imagine if HTTP were like this. You would have to type in a user name and password every time you clicked a link!) In this lab, you will use SFTP to put your first file on the web, i.e., use SFTP to make a file publicly available via HTTP. An SFTP server is a lot like a bank for files, i.e., a computer on the network with lots of hard disk space that offers individual users access to private storage. Just like with a hard disk on your own computer, you can copy, delete and rename files on an SFTP server, as well as maintain folders to keep your files organized. You use SFTP client software to connect with an SFTP server, just as you use a web browser (web client software) to connect with an HTTP server (web server). We learned that web pages are stored and delivered from web servers, but how do web pages get to a web server in the first place?

Sometimes, web authors create and edit them on the web server directly, but this is not the usual case. Typically, a web author edits a page on their own computer, makes sure the page looks right then transfers a copy to the web server. As soon as a file is transferred to a web server, it generally becomes publicly available, so working on their own computer helps ensure that there are no mistakes in the pages that actually go on the web, or “go live,” as some people say, borrowing the expression from broadcast media. In this lab, you will follow the same process to put your first file on the web.

File Formats

Although all programs save their data on disks as files, they often store them in a special *format*, a way of encoding information. (File format is also known as file *type* and is shown under this name in the Windows file Properties dialog.) The simplest file format is plain text, the format that Notepad and other text editors use, and files of this type are conventionally named with the *txt filename extension*. In Windows, every file format has a corresponding filename extension. Each file is displayed with the *icon* (small image) associated with the program used for the file format, like the little spiral-bound notebook for Notepad. Notepad cannot open word processor document files (for example, those in Microsoft Word format), because they are in a format different from plain text that includes data for text style, such as italicization and underlining, in addition to the text itself. The table below lists some common file formats and their corresponding icons, as displayed on most standard Windows PCs.

<i>extension</i>	<i>description</i>	<i>icon</i>
.txt	Plain Text	
.doc	Microsoft Word document	
.htm, .html	Web page source	
.pdf	Portable Document Format	
.gif	Graphics Interchange Format	
.bmp	Bitmapped image	