ssh Tutorial

Ssh is a program that allows you to remotely access other systems. It has two parts: the *server*, which must be active on the system you wish to connect to, and the *client*, which you run on your system to connect.

Ssh is available on Linux, MacOS, and Windows 10 systems. For other Windows systems, you will need to get a third party client. The web page `https://www.ssh.com/ssh/client` lists several of these. Most people use PuTTY.

How *ssh* Works

Ssh uses encryption to authenticate the server to the client, and to build a "tunnel". That's the term used when any information sent from one system to another is encrypted. Once the tunnel is set up, the user must authenticate himself or herself to the server to log in. This is done in one of two ways. Normally, the server will request the login name and password of the user. The user can also arrange that the client will automatically authenticate the user to the server—no password needed![1]

In what follows, we'll assume the server asks the user for the user's password.

Getting Started with *ssh*

Let's say your login name on the remote system is "robin", and you want to connect to the CSIF system "pc19.cs.ucdavis.edu". You would issue the following command:

```
ssh robin@pc19.cs.ucdavis.edu
```

The first time you do this, you will see something like:

```
The authenticity of host 'pc19.cs.ucdavis.edu (128.120.211.76)' can't be established.
ECDSA key fingerprint is SHA256:EAfDII2xwnrM58zuhxa73bjoLgzlPVmVhBmIf1klS/s.
Are you sure you want to continue connecting (yes/no)?
```

Type "yes" (the full word). This message is simply saying that you have not used *ssh* with that particular servern and asking you for permission to set up the encrypted tunnel.

You will then be prompted for your password. After you type it, you'll see something likr:

```
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-69-generic x86_64)
```

followed by other information about the system, and after that a prompt. You're now in the Linux shell on the remote system!

Copying Files Back and Forth

Now, let's retrieve a file from pc19.cs.ucdavis.edu to your client computer. On the server (pc19.cs.ucdavis.edu), you have a file called "program.c". You want to move it to your home system, the client. For this, we use the *scp* (Secure CoPy). You would type the following:

```
scp robin@pc19.cs.ucdavis.edu:program.c program-new.c
```

This copies the file "program.c" from your home directory on "pc19.cs.ucdavis.edu" to your client system. Once copied, it has the name "program-new.c" on the client system.[2]

The first argument to *scp* has the form *login_name@server:file_name*. The second is the location to which the file is to be copied. If what you give is a directory, the file will be created in that directory with the same name as it has on the server. Otherwise, the client will give the file the name of the second argument.

You can also copy a file from the client to the server. To do this, just switch the order of the last 2 arguments:

```
scp myprogram.c robin@pc19.cs.ucdavis.edu:program.c
```

This copies the file "myprogram.c" from your current working directory on the client to your home directory on "pc19.cs.ucdavis.edu" . Once copied, it has the name "program.c" on the server system.

[1] Setting this up is complicated for people who have not used *ssh* before, which is why I am not recommending it. If you want to set it up, go to `https://www.ssh.com/ssh` and follow the directions under "Automate with SSH Keys, But Manage Them."

[2] The name of the file on the server system remains the same.

A Short-Cut

If your account has the same name on the client and on the server, you can leave off the login name in the ssh and scp commands. So the above *ssh* command would be:

```
ssh pc19.cs.ucdavis.edu
```

and the first *scp* command would be:

```
scp pc19.cs.ucdavis.edu:program.c program-new.c
```

Of course, it is always safe to give the login name.

Credit

This was written for ECS 36A, Programming and Problem Solving, in Fall 2019 by Matt Bishop.