Remote Access to a Linux Server*

Terry Sergeant

Fall 2019

Contents

1 The Big Picture **2**
- 1.1 Preliminaries . . . 2
- 1.2 Connecting to the Server . . . 2
  - 1.2.1 Finding an ssh Client . . . 2
  - 1.2.2 Dealing with the Command-Line Interface . . . 3
  - 1.2.3 Transferring Files . . . 3
- 1.3 Summary . . . 3

2 Using the Command-Line Interface **4**
- 2.1 Managing Files . . . 4
- 2.2 Editing Text . . . 6
- 2.3 Compiling Programs . . . 6
- 2.4 Sending Email . . . 7
- 2.5 Printing Documents . . . 7

3 Graphical Interface **7**
- 3.1 Managing Files . . . 7
- 3.2 Editing Text . . . 7
- 3.3 Compiling Programs . . . 7
- 3.4 Sending Email . . . 8
- 3.5 Printing Documents . . . 8

*Online at `josephus.ser ea tser  ces.com/classes/all/remoteaccess/`

1 The Big Picture

1.1 Preliminaries

This document assumes you have completed the steps outlined in the *Linux Quickstart* document. and the *Using Linux in JB 202* document. These documents are available online at:

`https://josephus.hsutx.edu/classes/all/`.

If that is the case then you have an active account on the Linux server named `csci.hsutx.edu` and you have successfully made use of your account by connecting from JB 202. For the remainder of this document I will refer to the server as `csci`.

This document explains other ways in which you can connect to the server to perform useful work. This will be helpful because the JB 202 lab maintains limited hours. There are other computer labs on campus that have longer hours. One of those labs (JB 104) has software that will make connecting to `csci` easier.

1.2 Connecting to the Server

There are three protocols through which you can establish a connection to `csci` (from off-campus):

1. `http` (view web pages on `csci` using a browser)
2. `ssh` (log in to `csci` and type commands using an `ssh` client)
3. `sftp` (transfer files to/from `csci` using an `sftp` client)

You are undoubtedly familiar with making `http` (web) connections to a server: to view a web page you simply start your favorite browser, enter the URL of the site you want to contact, and voila, you have made a connection to the server and it provides you with information.

In the same way, to make an `ssh` connection to the server you can simply launch your favorite `ssh` client and enter the name of the server to which you want to connect. Of course, there are two problems with this advice:

1. Chances are, you do not have a favorite `ssh` client, and
2. When you connect using `ssh` instead of being greeted with friendly, graphical information, you are confronted with a command-line interface that wants to know your username and password.

Allow me to address these issues invidually.

1.2.1 Finding an ssh Client

There are a variety of Windows-based `ssh` clients. One of the free ones is Putty (available at `https://www.chiark.greenend.org.uk/~sgtatham/putty/`). The remainder of the document is written assuming that you are using Putty. I recommend that you do a full download and install so that you also get the `sftp` client. NOTE: Most folks who use MacOS just use the command-line rather than installing a separate program.

Once you download and install Putty simply launch it and enter the full Putty server name (`csci.hsutx.edu`). Upon establishing a connection, you will be prompted for your username and password. After you log in successfully you will be greeted by the Linux command-line interface. At this point it would be helpful if you knew a lot of Linux commands! If that is not the case then you should turn your attention to the next section.

1.2.2 Dealing with the Command-Line Interface

There are two primary methods of dealing with a command-line interface:

1. learn several useful commands, or
2. run graphical programs that will perform the useful commands for you

I have used the command-line interface for some time and, in fact, prefer it to a graphical interface for many uses. If you are new to Linux then this interface will be difficult to use at the beginning. One immediate advantage over using a graphical interface is that no additional software is needed.

If you want a graphical user interface to run through your `ssh` connection on a Linux server then you will have to install and run an X-Server on your Windows computer. Here are some options:

- Download and install the free X-Server called Xming. It is available at `https://sourceforge.net/projects/xming/`. Another free option is VcXsrv (`https://sourceforge.net/projects/vcxsrv/`).
- With a *lot* more hassle, another free alternative is Cygwin/X which can be downloaded at `https://x.cygwin.com/`. The installation process is somewhat painful and time-consuming because it involves creating a mini-Unix system on your computer and the X-Server is actually run within that environment. This means a big download and lots of chances to screw up the installation.
- One "commercial" X-Server is MobaXterm (`https://mobaxterm.mobatek.net/`). There is a free version with some limits and a full-featured pro version.

For Macintosh XQuartz `https://www.xquartz.org/` is an option.

1.2.3 Transferring Files

To transfer files to/from `csci` you'll need an `sftp` client. For Windows I recommend FileZilla which is a free download and can be obtained at `https://www.sourceforge.net/projects/filezilla`. Another good option is WinSCP (`https://winscp.net/eng/download.php`).

1.3 Summary

If you read the first part of this section with some level of understanding you know how to connect to a Linux server (using `ssh`) and you also know that once you've connected you have a choice between using a command-line interface or a graphical interface. Allow me to summarize the advantages and disadvantages of the two approaches:

Command-Line Interface	Graphical Interface
no additional software needed	requires expensive (either money or time) additional software
very fast (if you know commands and tools well	will run slowly, especially when accessing the server from off-campus (because the GUI information has to travel across the network)
very time-consuming to become proficient with the commands and tools needed for assignments	issuing commands and running programs will be intuitive using standard GUI widgets

2 Using the Command-Line Interface

This section describes how to perform various tasks via the command-line interface.

2.1 Managing Files

Here are some file-manipulation commands you will likely want to use.

ls *lists files in a directory.* There are many options that can be supplied to modify the behavior of `ls`. Here are some examples of using `ls`.

`ls`	lists all files in current directory
`ls -F`	modifies file listing to differentiate between types of files (directories, including permissions
`ls ~/programs/c/`	lists all files in the subdirectory `c` in the subdirectory `programs` in your home directory
`ls -a`	shows all files, including hidden ones; files with names that begin with a period are designated "hidden" and are immune from some file operations
`ls *cpp`	shows all files in the current directory whose names end with `cpp`
`ls -al`	same as doing both `-a` and `-l`

mv *moves or renames a file.* This command can be used rename a file or to move it from one directory to another. It should be noted that mv can be applied to directories as well as files. Examples:

`mv old.txt new.dat`	renames file from `old.txt` to `new.dat`.
`mv myprogs/cool.cpp .`	moves file named `cool.cpp` in the `myprog` directory into the current directory
`mv myprogs oldprogs`	if we assume that `myprogs` is a directory (as in the previous example) then this would rename the directory to be `oldprogs`

cp *copies a file.* This works pretty much the same way as `mv` with two exceptions. First, the original file is left intact. Secondly, `cp` doesn't copy entire directories (unless you use the `-r` option).

rm *removes a file.*

chmod *changes mode (access permissions) for a file.* Every file and directory has read, write, and executable permissions for three classifications of users: the user (u), the group (g), and others (o).[1] For example, a file can be set to be readable and writeable, but not executable for the user/owner and not readable, writeable, or executable for group members or for anyone else. Likewise, you may want to allow anyone to view a particular file, but not write to it, etc. To view the current settings you can use the `ls -l` command which will produce output similar to this:

```
drwxrwxr-x   3 sergeant sergeant     1024 Aug 14 10:27 office52
-rw-------   1 sergeant sergeant    26907 Jul 30  1999 personal.xls
```

[1] That's nine permission settings if you're counting.

```
drwxrwxr-x   2 sergeant sergeant     1024 Jul 24 07:50 rpm
-rw-rw-r--   1 sergeant webcache    28492 Aug 18 18:29 stats.ps
-rw-rw-rw-   1 sergeant sergeant     5373 Feb  5  2000 questions.tex
```

Each line contains information for a single file. The first ten characters provide information about permissions. The first character has a `d` if the file is a directory and a `-` otherwise (usually). The remaining nine characters specify the read (r), write (w), and executable (x) permissions for user, group, and others. The file named `personal.xls` is readable and writeable by the user (`sergeant`) and not available to anyone else. The file `stats.ps` is available for reading and writing to the user (`sergeant`) and the group (`webcache`), but only readable to everyone else. You may notice that both directories (`office52` and `rpm`) have executable permissions turned on ... without those permissions a directory is not listable/viewable.

There are two main ways to invoke the `chmod` command. Only the mnemonic method is explained here. The command is followed by a request for new permissions and lastly the filename(s) to which the new permissions should be applied.

`chmod guo+rw public.txt`	makes the file `public.txt` readable and writeable by anyone
`chmod go-rwx *cpp`	remove all types of access from group or others for all files whose names end with `cpp`

mkdir *makes a new directory.* Simply follow the command with the desired directory name.

rmdir *removes a directory.* The main thing to be aware of here is that the directory must be empty before it can be removed with `rmdir`.

cd *changes to a new working directory.* Two notes of interest: `cd ..` moves one level up the directory tree. Also, `cd` with no directory name changes to the user's home directory (the directory you are in when you first log in).

pwd *displays present working directory.* Depending on how your prompt is displayed you may not have a reminder as to what directory your are currently working in. This command will let you know where you are.

Here are some additional commands (that don't necessarily deal directly with managing files) that you may find helpful:

man *displays manual pages for any command.* For example, if you wanted to find out *all* of the options available for the `ls` command you can type `man ls`.

apropos *displays list of commands relevant to the specified topic.* Example: `apropos clock` displays information about various commands dealing with a clock.

find *finds files.* This command has numerous options that allow you to find a file based on all manner of criteria. Example: `find -name henry.txt` will search the current directory and all its subdirectories for a file named `henry.txt`.

grep *searches files for given text.* If you want to find which files contain certain text then `grep` is what you want. For example: `grep Diana *.txt` shows all occurrences of "Diana" in all files in the current directory whose names end with `.txt`.

passwd *changes your password.* NOTE: Use the `yppasswd` command if using a lab computer.

script *captures output to a file.* This is especially useful for capturing the output of an interactive program. The command works as follows:

```
script myprog.out
ls
javac prog1.java
java prog1
exit
```

In this example all of the output generated by the `ls` command and by the program compilation and by the program that was executed are all sent to the file named `myprog.out` (as well as to the screen). The command `exit` tells the script command to stop logging the output to the specified file.

2.2 Editing Text

There are *many* fine text-based editors available on `csci`. Most folks end up using either `vim` or `nano`. A benefit of `nano` is that it is easier to learn and you are welcome to use it instead of `vim`. I know and use `vim` so that is what I document here.

Bad news: `vim` has a steep learning curve. Plan on spending quite a bit of time learning to use the editor before you attempt to do something substantial with it.

Here are some resources that should help your transition to `vim`:

Advice Don't try to use `vim` to do anything useful until you've gone through an introductory tutorial and you've taken some time to play around with it. Then, and only then, should you try to use it for an assignment.

Also, in the early stages don't try to use `vim` when you are in a hurry. The idea is that you want to be proficient before you add the stress of "hurry" to the stress of learning something new.

Tutorial On `csci` there is a `vim` tutorial called `vimtutor`. Spend time to go through that tutorial. You may even want to go through it several times until you become comfortable with the various commands.

Quick Reference Look over the quick reference cards here: `http://www.freetechie.com/blog/5-helpful-vi-and-vim-vi-improved-cheat-sheets-and-quick-reference-cards/`

I have used the one by Don Bindner.

On-Line References Of course, there are many online references.

2.3 Compiling Programs

Writing and compiling programs using a text-based interface in Linux involves writing/editing the source code using an editor of your choice following by compiling the program by issuing the appropriate command. Many editors provide commands within them to allow compilation without leaving the editor. I sometimes work with two windows open: one for issuing compilation commands and one for editing the source code.

To compile Java programs use:	`javac MyProg.java`	compiles Java program found in `MyProg.java` and, if successful, produces a class file named `MyProg.class`
	`java MyProg`	runs a compiled Java program

2.4 Sending Email

For sending and receiving (text-based) email I use a client called `mutt`. I like it an use it because it is fast, configurable, and simple to use. But, as with many text-based programs it may take a while to become comfortable using it. To invoke this program simply type `mutt` at the command-line.

2.5 Printing Documents

To print virtually any document I recommend that you use the `a2ps` print formatter rather than sending your document straight to the printer. To use `a2ps` to print a document named `myprog.cpp` in the JB 105 computer lab, issue the following command:

```
a2ps --printer=jb105 myprog.java
```

Of course, you can change `jb105` to `jb104` to print to the JB 104 printer.

3 Graphical Interface

You may find that while you prefer to do most tasks via a GUI, that some tasks (such as sending email or printing documents) are quite simple when invoked from the command-line. Feel free to consult various subsections from section 2 as needed.

The irony of having an `ssh` connection through which you forward GUI information is that the graphical programs have to be invoked from the command line. To run any of the programs discussed in this section simply type the name of the program at the command-line. I recommend that you follow the command with an ampersand (&) which will cause the program to be launched in the background and will leave your command-line open to invoke other programs as well. For example to invoke the `gedit` editor you could type: `gedit&`

3.1 Managing Files

To load a file managing utility (similar to right-clicking on the Windows `Start` menu and selecting `Explore ...`) you can use `nautilus`.

3.2 Editing Text

If you only need an editor for creating source code you will want to read the discussion in the following section before reading this recommendation. For general text editing needs `gedit` works fine. This editor has a simple interface but performs syntax highlighting for some programming languages (including Java).

3.3 Compiling Programs

a remote connection to a Linux server:

- use a full-featured editor try VSCode (type `code`)

- use a scaled-back integrated development environment (IDE); (try `jgrasp` or `bluej`)
- use a graphical text editor and compile from the command-line; (try `gedit` along with compiler commands given in section 2.3)

The advantage of the IDE is that you may be generally familiar with that style of program development. On the down side, it can be somewhat of a pain to create a simple stand-alone program. Also, the hefty interface may be exceptionally slow over a network. Lastly, if you have trouble using it or understanding it, I will be of no help because I haven't ever used it!

The advantages of using a graphical editor and invoking the compiler manually are that it will work more quickly over the network and I will be able to be of some assistance if things don't go well. Also, you don't have to create an entire directory (folder) filled with various files just to compile a program. On the down side you will need to learn to use the text-based compiler commands.

A scaled-back IDE falls in the middle of the other two options in terms of required network capacity and ease-of-use.

3.4 Sending Email

Just use web mail.

3.5 Printing Documents

Any GUI program will provide a menu option for printing. Rather than printing to the default printer you will want to specify an alternate printer (assuming you want to print in one of the labs).