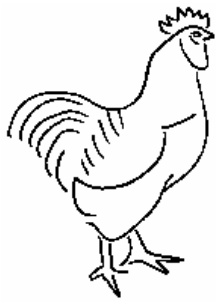
CLTC Project Information Sheet 2:

Remote Login to Another Linux System



Developed by The 'Free Range' Community Linux Training Centre Project −
Version 1.0, August 2002. http://www.fraw.org.uk/cltc/

This briefing looks at remotely logging into another server − either on you local network, or somewhere else on the Internet. For most Internet users this is not an important issues. But for those who hire space on commercial servers it is often necessary.

Remote login

Most people are used to sitting at the computer they are using. But when you network computers, that is not necessary. It is possible to login into another Unix/Linux system as if you were at the keyboard, entering the commands yourself. This has a variety of uses.

Originally, in the early days of network computing, there was little effort to protect systems connected to the Internet. This was because if you were on the 'Net, you must be a pillar of the establishment. That changed in the 1980s when 'crackers' (people who covertly break into computers) became a problem. Today the systems that allow people to login to an online system are more secure.

Remotely logging in to a server is much the same process as starting up your Linux system and logging on from the keyboard. You have an account on the system, with a name and a password. The purpose of logging in is to authenticate your access to the system. This is the basic principle of remote login, except that you can use a 'secure shell login' system that additionally encrypts the data moving between you and the server so that eavesdroppers can't get hold of your passwords and other important information.

Remote login to a computer has a variety of uses. If you work on a server, you can login to work, check your mail, or get information on the system. Some people also use remote servers as a means of secure data backing−up. Instead of backing−up to a floppy disk or CD ROM, you send your important files to another server where they are kept.

Mostly, people remotely login to another server in order to maintain Internet servers. If you rent a commercial server where you run anything more than just a web service, then at some point you might have to login to the server. Many of the cheap commercial 'server farms' use Linux on their systems because it's cheap to run, stable, and easy to configure. Therefore using Linux at home can be a helpful step to learning how to undertake commercial Internet or server configuration.

One last point. Remote login is entirely 'console' − text only. You can't run an X server − Linux's graphical interface − over a remote connection. There are ways of connecting up an X server over a network, but there are many security problems implicit in doing this − so it isn't done very often.

The basics − `rlogin`

Once people got security conscious about people remotely logging in, new systems were developed to provide better control of how people connected to another system remotely. One of the most well used is `rlogin` − or 'remote login'.

`rlogin` provides an almost identical process of logging in over a network to the ordinary Linux login. However, there are some security problems with `rlogin`, and so it has been slowly phased out in favour of `telnet` (see below).

Most Linux distributions still support `rlogin` − although you usually have to install it because it's not automatically installed. Like `telnet`, `rlogin` requires a 'daemon' working on the server to control connections (see below).

The standard − `telnet`

`telnet` has taken over from `rlogin` because it provides more secure control over connections. It is also able to handle more complex exchanges of information, such as 'control characters'. `telnet` also enforces conditions on what those making the connection are able to connect to.

Internet services like `telnet` require a 'daemon' program to work. A daemon is a program that stays in the server's memory, automatically monitoring the system to

respond to requests for services. The telnet daemon is called *telnetd* – some distributions install it automatically, and some do not. But on most systems you have to manually enable it (it doesn't work automatically because it has security implications).

`telnet` is a usually console program (although there are some graphical versions, which is pretty pointless as it's a text–only system). So when you run telnet from KDE, what you actually end up with is KDE's console program that automatically runs the telnet program (see right, top).

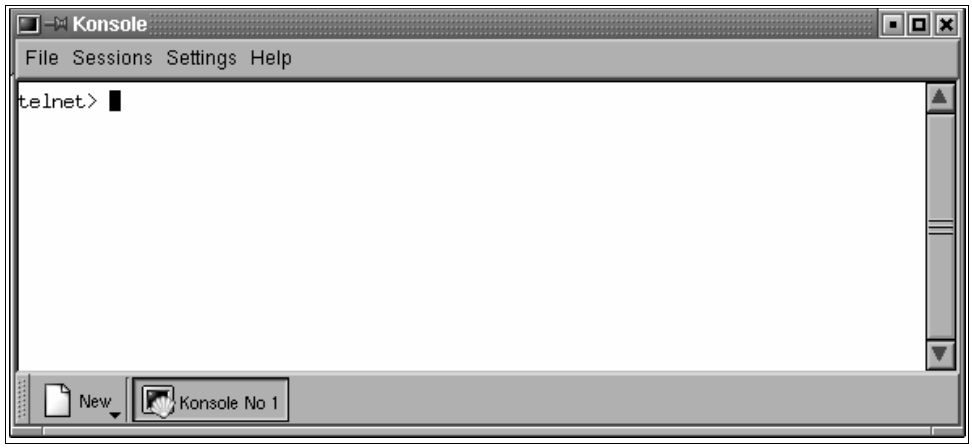
To create a login using telnet you 'open' a connection to the required machine. This can be either a domain name, or an numeric IP address (see right). If the machine is up, you will get a login request. You then enter your user name, and then your password, as if you were logging in locally.

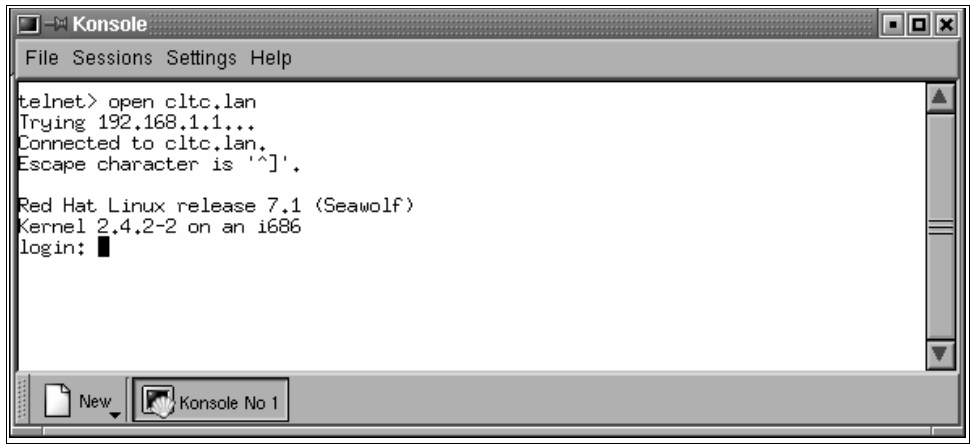
As well as using the 'telnet' icon, you can run the client program by just typing '`telnet`' from the Linux command line.

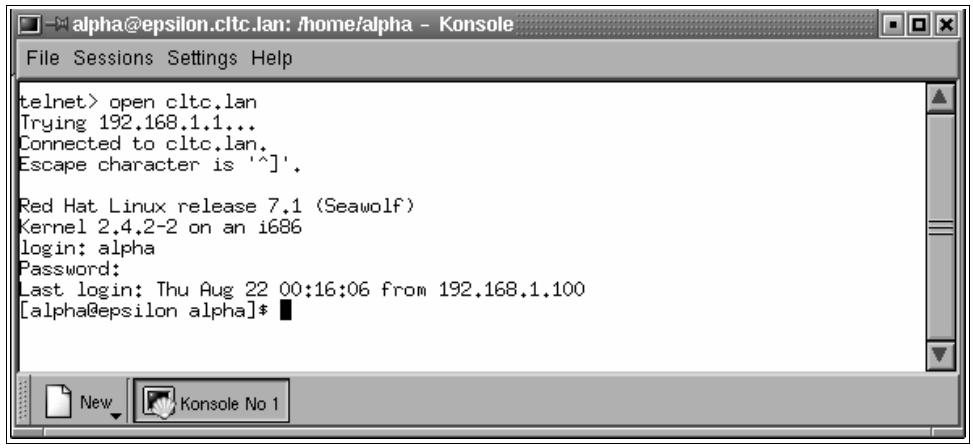
There are various commands that you need to know to use `telnet`:

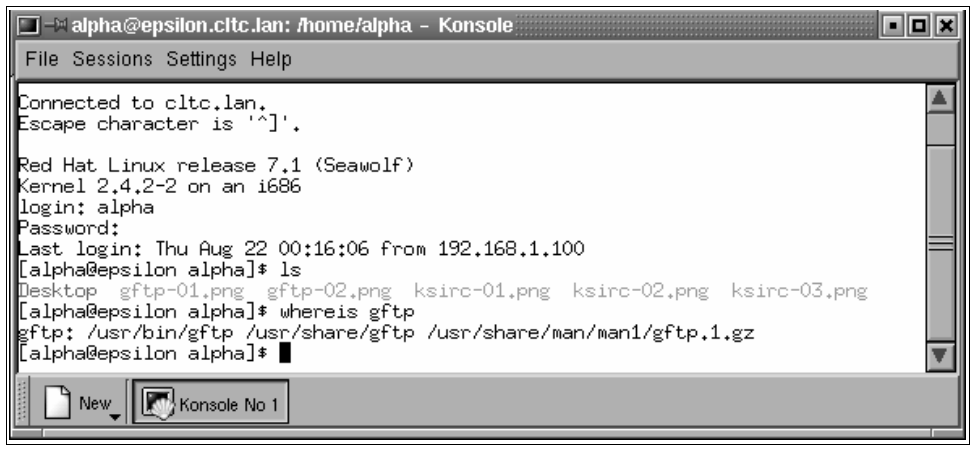
- `open`, followed by the name/IP address of the server, opens a connection.
- `close` closes the current connection.
- Entering `logout` or `exit` will terminate the connection, and exit the client.
- `quit` closes the telnet client.
- `help` or ? Prints a fill list of the commands you can use with telnet.

Once you have logged in you can enter commands as if you were at the Linux command line locally. This can actually present write a security risk. Therefore accounts that are remotely logged into usually have restric–

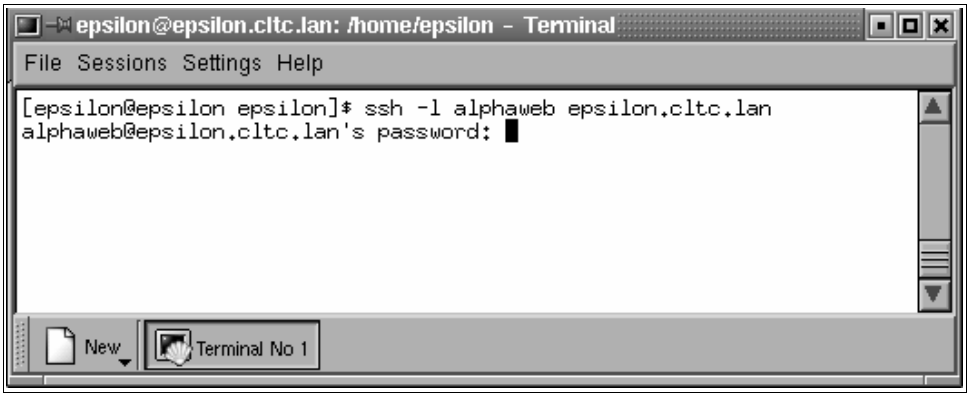


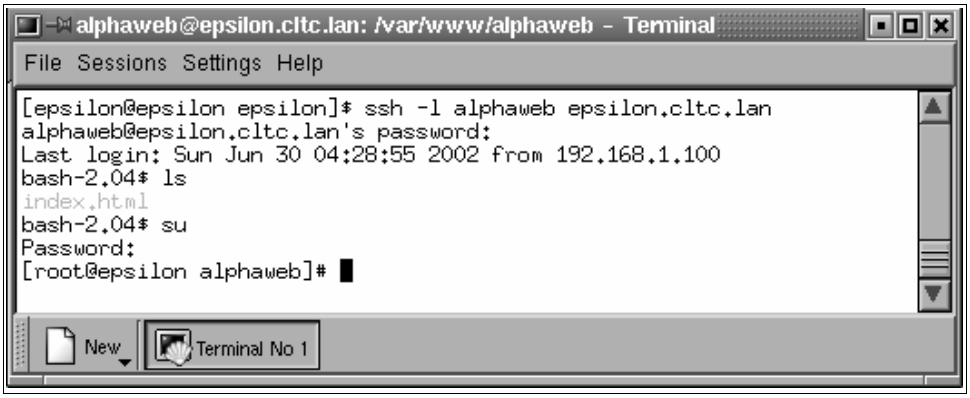






tions upon the commands or resources you can access from the remote connection.





Secure shell login

Both `rlogin` and `telnet` are useful for logging into a remote system you use processes or resources present on that machine. But for those who wish to undertake any type of work that has security implications, such as maintaining a web server, these clients are insecure.

The alternative is the 'secure shell' client – or *ssh*. `ssh` uses session–based encryption to prevent anyone who has access to the stream of network traffic reading information. Like telnet, `ssh` requires a 'daemon' program to make it work on the server – `sshd`.

`ssh` is run from the command line (see right, top). You can do this from the Linux command prompt, or open a terminal window from KDE. ssh requires that you give a server name – otherwise you get an error. If you don't provide a user name, you'll be logged in under your current user name. Therefore, on someone else's server, you'll have to specify a user name. You do this by specifying the switch '–l', then the user name. The format of the while command is:

`ssh –l` *`user-name server-name`*

To get a more complete list of options, enter `ssh –h`

Every server must be authenticated by your client program. Therefore if you've never used the server before you'll get a warning that the server is not authenticated. You are then asked to type 'yes' or 'no'. If you type 'yes' authentication proceeds, and the details will be added to the information on your computer (so you won't get the error in future).

After connection you are asked for a password. If the password is accepted you'll get a command prompt (see right, second image). You can now work securely on the server.

With `ssh` you can't log in as the 'root' user. For server maintenance you'll nearly always need to use root privilege. Therefore what you so is use the `su` command to temporarily log in as root. After entering `su`, you are asked for the root password. If the password is accepted the command prompt will change indicating that you are now root. When you have finished working as root, you leave the root shell be entering `exit`.

If you work on a server remotely, the only problem with `ssh` and `telnet` is that you can't move files. When you need to move files you'll have to use FTP. However, because the server should accept multiple connections on the same account, you can stay connected with `ssh` whilst opening FTP to move the files across.

Finally, there is a graphical front end (see below) for KDE called 'K Secure Shell' – `kssh` (although this isn't used with the CLTC system). This provides a few more options to log in, and additionally allows you to edit the information about servers stored on your machine.