Networking

Unix and Linux are by design net-centric operating systems. In many cases, you will use the network seamlessly without realizing, e.g., remotely stored passwords and networked file systems. Of the myriad of network tools available, there are only a handful with which you must be familiar for basic survival, and those will be described herein.

1 World Wide Web

Because web utilities are so prevalent in today’s society, very little discussion is necessary here. The most popular web browsers on Unix and Linux systems are Netscape (www.netscape.com) and Mozilla (www.mozilla.org). Depending on which is available on your system, you can invoke either of these by typing netscape or mozilla on the command line. Many systems also make available a text-oriented browser called Lynx (lynx) for use on rare occasions when you only have a text interface available.

2 Remote Login

Historically, one could remotely login to a different machine on the network using a utility called Telnet. However, for security reasons, SSH (Secure SHell) has become the de facto standard. SSH provides transparent authentication and encryption to permit a secure connection between two untrusted hosts on the network. Unlike Telnet, SSH encrypts passwords and other information sent across the network so that malicious persons cannot intercept and meaningfully decipher sensitive information.

You can use SSH by issuing the command ssh followed by the name of the machine to which you are attempting to login. For example, the command

    ssh aurora.richmond.edu

will establish a secure connection between the current user’s machine and aurora. You must have a valid account on the remote machine to login successfully. By default, the account name on the local machine is supplied as the account to use for logging into the remote machine. If your account name is different on the remote machine, you must supply that account name as an argument to SSH. For example, if your account name is jstu32 on aurora, you can specify the account name using either

    ssh -l jstu32 aurora.richmond.edu

or

    ssh jstu32@aurora
Also note that, as in the last example, if the machine is local to the department network, you need only supply the name of the machine without \texttt{.richmond.edu}.

The first time you login to a machine, you may see a message like

\begin{quote}
\textbf{The authenticity of host 'aurora (141.166.206.212)' can't be established.}
\textbf{Are you sure you want to continue connecting (yes/no)?}
\end{quote}

Simply respond \texttt{yes} (you should not see the message on subsequent logins) and you will be prompted for your password. To end your \texttt{ssh} session, type \texttt{exit} or \texttt{logout}.

For further information, refer to the man pages on \texttt{ssh}.

### 3 Remote File Transfer

FTP (File Transfer Protocol) has historically been the most commonly used utility for transferring files from one machine to another. However, like Telnet, FTP is not a secure protocol and has since lost its foothold to SCP (Secure CoPy), provided as part of the SSH suite. SCP offers secure, encrypted transfer of files between untrusted hosts.

As an example, to copy a file named \texttt{sort.cc} into the home directory of the account \texttt{jstu32} on the remote host \texttt{aurora.richmond.edu}, you can issue the command

\begin{verbatim}
scp sort.cc jstu32@aurora.richmond.edu:
\end{verbatim}

To copy \texttt{sort.cc} from the home directory of \texttt{jstu32} on remote host \texttt{aurora.richmond.edu} to the current directory (known as \texttt{.}) on the local host, you can issue the command

\begin{verbatim}
scp jstu32@aurora.richmond.edu:sort.cc .
\end{verbatim}

Similar to SSH, if the remote host is local to the department network, you need not supply \texttt{richmond.edu}. Also, if your account name is the same on the remote host, you need not supply the account name in the \texttt{scp} command.

The colon within each of the commands above is important. By default, any path that occurs after the colon is relative to the user's home directory. For example, the following command will copy \texttt{sort.cc} into \texttt{code/}, a subdirectory (which must exist) in the home directory of \texttt{jstu32} on \texttt{aurora}.

\begin{verbatim}
scp sort.cc jstu32@aurora:code/
\end{verbatim}

Alternatively, you can specify an absolute pathname instead of using the default home directory. The following command will copy \texttt{sort.cc} to the directory \texttt{/tmp/code/} on \texttt{aurora}.

\begin{verbatim}
scp sort.cc jstu32@aurora:/tmp/code/
\end{verbatim}

You can copy multiple files at once by issuing a command similar to the following example.
scp sort.cc sort.h swap.cc swap.h jstu32@aurora:code/

You can also copy an entire directory structure by supplying a \texttt{-r} flag (recursive copy) to \texttt{scp}. The command

\texttt{scp -r code/ jstu32@aurora:}

will copy the local directory \texttt{code/} and all its contents into the home directory of \texttt{jstu32} on the remote host \texttt{aurora}. Note that, when recursively copying an entire directory, the directory need not exist on the remote host. If you want to copy many selected files, you may want to take advantage of the archiving capabilities — see the “Archiving and Compression” document.

Take care when using \texttt{scp}. No prompting is provided, and therefore files can be easily and inadvertently overwritten! For more information, refer to the man pages on \texttt{scp}.

\section{Remote X Windows}

Beyond text-based remote login, you can also see on the display in front of you GUI applications that you are running on a remote host. This is very useful if you are working on a machine (say, in G30) that does not have a particular application installed, but the application is installed on another computer to which you have access (say, \texttt{cypress}). It may also be convenient to use this capability even if the application is installed locally but you need to access a file available only on another machine.

X Windows is the Unix/Linux window manager responsible for displaying all the windows and applications on a display. X Windows is an \textit{event-driven} system, which means that X Windows intercepts \textit{events} (such as mouse clicks and key presses) and passes them to the appropriate application. The X Windows system that is intercepting events and displaying applications is called an X Windows \textit{server}. Application programs are called \textit{clients}. X Windows clients may be either \textit{local} or \textit{remote}. A local client is an application that is running on the same machine as the sever that is hosting the application. A remote client is an application that is running on a different machine on the network but is being displayed on the screen managed by the server. When an event is detected on the local machine that corresponds to a remote client’s window, that event is sent to the remote client over the network. The remote client then reacts to the event (for example, by redrawing its window) and the reaction is reflected on the screen in front of you.

\subsection{Remote X Windows on a PC}

In order to display X Windows applications on a PC desktop, you will need software to appropriately interpret the applications and associated events. Exceed by Hummingbird is an example of software that displays X Windows applications on a PC. Exceed is available on all lab machines in G25 and G30, and a one-year licensed version for you to install on your personal machine is available — follow the corresponding link from the Department’s Unix help document page.

Simply start Exceed by clicking on the Exceed icon (or by locating Exceed through the menus from the \texttt{START} button). You should be presented with a window entitled “XDMCP Display Manager Chooser” — select an appropriate host from the list and click \texttt{OK}.
Networking

If there are no hosts listed, you will need to configure Exceed. Right click on the Exceed tab in the task bar at the bottom of the screen. From the menu, choose Tools and then Configuration.... Click on the “Communication” icon. Within the “Communication” window, the mode should be set to “XDMCP-broadcast”; if not, use the drop-down menu to set it appropriately. Now click Configure. From the “Configuration” window, within the “XDMCP Broadcast” frame make sure the “Use Broadcast Address” checkbox is checked, and click OK. (In lieu of this last step, you can instead click Edit to manually add hostnames, e.g., aurora.richmond.edu, to the file xdmcp.txt.) From the “Communication” window, click OK, and in the resulting dialog box click Yes to perform a server reset. You should have now have a list of hosts in the “XDMCP Display Manager Chooser” from which to select.

After selecting a host, login using your username and password for the Math/CS department network (which may or may not be the same as your username and password for the general UR network — see the help document on “Unix login and password management”). You should now see Sun’s Common Desktop Environment (CDE). If you click on the terminal icon in the command bar within Exceed, a new terminal will open in Exceed. Note that the terminal is actually running on the host you selected, but is being displayed on your local machine! The other icons (e.g., XEmacs) work similarly.

4.2 Remote X Windows on Unix/Linux

If you are already working in a Unix/Linux environment, you do not need special software to display remote X Windows — the local X Windows server will be able to interpret the X Windows events from a remote machine.

In some cases (this step is not necessary if you are networking within the Department’s Unix network only), you may need to first give permission locally so that applications on the remote machine may use the local X Windows server. To do this, use the xhost command on the local machine, as in the following example which assumes the remote machine is aurora.

```
xhost +aurora
```

Login to the remote host (use ssh — see Section 2 of this document). Once you are logged in, you must specify that any applications started on the remote host should be displayed on the local host instead of the remote one. To do this, you need to set the DISPLAY environment variable on the remote machine. First, determine which shell you use by displaying the contents of the SHELL environment variable:

```
echo $SHELL
```

If you use the bash shell, set the display using the following command.

```
export DISPLAY=cypress:0
```

If instead you use the csh or tcsh shell, set the display using the following command.

```
setenv DISPLAY cypress:0
```
In this example, cypress is the name of the host running the X Windows server (the machine at which you are sitting). The :0 specifies the display number.

Now you are ready to execute a client application on the remote host aurora that will be displayed on the local host cypress. For example, executing xv on the remote host will cause the XV application to be displayed on your screen.

Refer to http://www.x.org/ for more information.