Additional laboratory

This is addicional laboratory session where you will get familiar with the working environment. Firstly, you will learn about the different servers present in the lab and how desktops are configured. Different servers can be accessed from desktop computers for several purposes. Secondly, you will learn how to connect to different servers for transferring files to or from systems where you have a user account.

1. FTP application

One of the most common uses of the Internet is transferring files from one system to another. And one of the older tools to do that is the **ftp** application.
We can use ftp to transfer files to a remote server anywhere on the Internet from our client computer. There is a basic check for user authenti cation on the server computer, however, as it happened with telnet, all the traffic is sent unencrypted.
Sometimes there is no need for a user to have an account on the server system in order to download something. That is why **anonymous** user was created. This name can be used when connecting to an ftp server and then we will be asked for our email address as the password (but nothing is really checked). This special anonymous user may have some restricted access to the files on the ftp server and almost in any case it has read-only permit, so no uploads are (usually) allowed for this user.
We are going to use the ftp client that comes with Windows, which is a text-based application though there are graphical ftp clients available too. This program is very similar to those available on Linux systems (or other flavors of Unix).
In order to start a session, a server name is needed (as it happened with telnet) and it is better to provide the server name on the same command line. We can start an ftp connection using the "Run …" option of the Start menu.
FTP client text-based user interface is based on a list of commands that are shown in the next table:

Command	Action
`? o help`	Shows the available commands
`Help command`	Shows this command's help
`ascii`	ASCII file transfer mode is selected
`binary`	Binary file transfer mode is selected
`bye`	End the ftp session
`cd <dir>`	Change the remote folder to <dir>
`close`	Closes the connection to the current server
`dir`	Current remote folder listing
`get <file>`	Download a file from the server
`hash yes/no`	Progress mark on/off
`lcd <dir>`	Change to the local folder <dir>
`ls`	Similar to dir command but different format

`mget <file>`	Download multiple files
`mput <file>`	Upload multiple files
`open <server>`	Connect to that <server>
`prompt`	Enable or disable interactive mode
`put <file>`	Upload a file to the server
`quit`	Same as bye

Once you are connected to a server, you can use and test some of the commands above. You can use **dir** to know which files are available on the server. You can change to another folder with **cd** and you can download or upload files with **get** and **put**.
Two file transfer modes are available on ftp: **ASCII**, which is intended for transferring text files and **binary**, which is intended for creating an exact binary copy of the original file. **ASCII** mode will adapt details of the representation of text files when transferring them, things like the proper end-of-line marker; that changes from one operating system to other. However, transferring a database or a compressed file using **ASCII** mode will corrupt it, so you have to be careful about which mode you use.
Multiple file transfers are possible by using wildcards and the commands **mput** and **mget**. When doing a multiple file transfer, the user will be asked for permission each time a new file is going to be transferred. The command **prompt** allows us to disable being asked for each file transfer.
File transfers happen between the current folders of server and client.
It is not possible to transfer several folders' content using a single ftp command.

Once you are done, the command **quit** or **bye** will close the connection and will finish you ftp session. The **bye** command will keep the application running, so you can connect to a new server by using **open** command. However **quit** command will finish the client application.

Exercise #1.

Connect to the ftp server `ftp.rediris.es` using the anonymous user. Move freely among several of the folder trying to figure out what type of information is available on folder /pub

2. File transfer with other applications

While command-line ftp clients are available on almost any system, they are not the only ones. Web browsers (Internet Explorer, Firefox, etc) do include an ftp client too, but that only handles file downloads.
Unless you create a special URL, the client access to the server is done using the anonymous user and you only need to type on the address line something like this "**`ftp://full.server.name/`**". Please note that this URL does not start with the typical "**`http://`**".
If you need to use a different user name then you can create a URL like this "**`ftp://username@full.server.name/`**". Then a dialog box will ask you for the password.

Exercise #2.

Connect to the previous ftp server using a web browser.

3. Internet standards

Exercise #3.

Search and save the RFCs 2821, 959 and 2616. Take a look of these documents. See the manner in that the IETF specifies the Internet standards. What protocols are specified in these RFCs?

4. Remote Terminal Service

Using a computer is usually done by you sitting at the keyboard. It is what we call **local execution**. All the software you run, either applications or the system's graphical interface only interact with this local user. If you are writing a document on a word-processing software what you can see it on the screen is generated by this same computer.
In the early years of computing, even the idea of a personal computer was an alien concept. The first computers were huge and very expensive so it became quite common that those systems were shared among several users at once. While only one processor was available, the use of some clever techniques made possible to create the illusion that several programs were running at the same time.
As computers got cheaper and the personal computers (PC) appeared; it became apparent that using a computer from a remote location was a very interesting thing to do. People could avoid a long drive and connect remotely to the computer instead. Even if you are not far away, it is always handy not to have to move to another seat to print a job or to do any other administrative work at another computer.
That is why the remote terminal service was created so users can log into a system remotely from a distant terminal (or PC). Remote terminal service allows a remote user (client) to connect to a computer (server) to have a work session, usually interacting with an application called *shell*.
The basic idea of such a service is that any key pressed at the keyboard has to be transmitted to the server where it will look as if a local user was typing this same key. Besides, any output intended for the display has to be sent back to the client so it can be shown to the user. The end result is that the user works exactly in the same way as if she were at a local terminal of that computer system.
Many computers allow more than one remote user simultaneously.

5. TELNET application

For a long time UNIX systems have been using a popular remote terminal application called **telnet**. This is one of many client-server applications used on the Internet. The server computer is running a piece of software called telnet server (telnetd) and the remote user is running the telnet client application (telnet) which will connect to the desired server. Client and server software will work together so the user gets the desired service.
The telnet server will check the username and password so only valid users of the server system are allowed in. If a user validation is successful then the user will interact with a shell program (it is a command interpreter). Using the shell the remote user can copy, move, rename or edit files on the

server system.
You just need to type this to start a telnet session from the Windows shell (CMD.EXE):

```
>telnet nombre.completo.de.servidor
```

Alternatively, you can type the same on the "Run …" dialog box from the Start menu.
You can start the telnet program without typing a server name on the command line; however the usual way is to type one server name. It is also possible to add a second parameter on the command line called the port number. This feature will prove quite useful in future laboratory exercises.
However, telnet application has a big drawback: all the information exchanged between server and client is transmitted unencrypted. Anybody that could sniff that traffic will be able to learn what we have been doing and, even worse, our username and password. That is why telnet is no longer the tool of choice for accessing remote servers and we must use the SSH (Secure Shell).

6. Using TELNET to dialog directly with servers

We shall again be using Telnet to talk to our remote server here, like POP. The principle behind sending an email is simple – your local computer connects to the remote mail server, talks to it using SMTP – "Simple Mail Transfer Protocol". When the mail is sent, the session is over and the remote server closes the connection.

When you use an email client like Outlook or Eudora, the mail client does all this for you. It automates the process of talking to your mail server to send and recieve emails. But what if you don't have, or don't want to use, a mail client? We can use Telnet!

First choose "Run" in yor Start menu and type in Telnet. Telnet is an application that allows us to communicate with remote computers. In this example, we shall be communicating with Yahoo's SMTP mail server. Choose "Remote System" from the "Connect" Menu. This will give you a box, with 3 input boxes. Type in the host name – the address of the mail server. For my Yahoo, the SMTP mail server is at smtp.mail.yahoo.com .

Now about the port: The port is a sort of a "gateway" to a computer. On the internet, each protocol, by convention has one or two port numbers assigned for itself. The HTTP connection is usually done using ports 80 and 8080; while POP transactions are done using port 110. For SMTP, port 25 is used. So type in 25 for the port.

6.1 Converstion with the Server

Now Click on connect. Once you're connected to the mail server, the mail server will respond with something like this:

220 smtp017.mail.yahoo.com ready.

Now we need to introduce ourselves to the computer and specify the sender's address. Technically, it is possible to use any SMTP server to send a mail with any server's name as the sender. This is called "Message Relaying". Since almost all servers have this feature turned off, we will simply type in the name of the SMTP server itself. [Note that you will not be able to see what you type.]

HELO smtp.mail.yahoo.com

The server will respond with:

250 Hello smtp.mail.yahoo.com, pleased to meet you.

Now we specify the sender:

MAIL From:
The server replies:

250 ... OK

There are a few observations to be made here – note that you can specify any sender here. So if you wanted to cheat the server and send bogus mail, the SMTP will not stop you – it has no security provisions. To add security, they combine the SMTP with POP authentication. So, you will have to login using the POP protocol once before using SMTP. [see Dec 2001 issue for POP mail] Also note that whenever the server sends a message, there's a 3 digit code along with it.

For example, when it sends 250, it means that the Transaction's okay. If it's 220, it means Service Ready. If it's 500, it means there's been a syntax error in the command that you sent, and so on. There's lots of these codes, each having a specific meaning.

This is a very useful thing, as mail program using the protocol will not need to read any of the English text – they will simply read the code to understand what the response is.

Now we type in the recipient and the data, and then quit:

RCPT To:

250 ... Recipient ok

DATA

354 Enter mail, end with "." on a line by itself
From: anaplexian@yahoo.com
To: anaplexian@yahoo.com
Subject: Hi there!

This is a test message!

.

250 Mail accepted

QUIT

221 smtp.mail.yahoo.com delivering mail
[connection closed]

Take a look at the format of the email – it had a bunch of details like From, To, and Subject listed, and then I left a line and then started my email. This is because a normal email comprises of minimum two parts, the header and the body, which are separated by a blank line.

The moment you send this email and close transaction using QUIT, the mail server will send the mail off to it's destination.

So now we can send email using SMTP, (and recieve using POP) all without the use of a mail client or a web browser. Note that the commands we did are only a part of the whole list – there's a lot more you can do with SMTP and POP.

So next time you want to check your mail, do it the cool way – use Telnet!!

SMTP CheatSheet

List of Basic SMTP Commands:
HELO: identifies client

MAIL: identifies the sender of the message.

RCPT: identifies the recipient. More than one RCPT command can be issued if there are multiple recipients.

DATA: To type in the message

QUIT: terminates conversation and closes connection.

Exercise #4.

Use the TELNET client to contact with your E-mail server. Use the SMTP commands and see the results.