15. THE INTERNET

This chapter describes elvis' ability to read and write data via the internet. The topics discussed here are:

In addition, you should probably read the Information via the Web and Using elvis as a Web browser sections of the Tips chapter. You may also wish to read the "html" section of the Display Modes chapter.

15.1 URLs

Wherever the traditional vi expects a filename, elvis expects a Universal Resource Locator, or URL. The names of local files are one type of URL; other types give the names of resources available from servers on other machines, accessible via the Internet.

URLs have the form protocol://server.domain.name:port/directory/file where the components of the URL have the following meanings:

protocol
This tells elvis how to read the URL. It can be one of the following:

* **file**: for local files.
* **buffer**: for already-loaded edit buffers.
* **http**: for the HTTP Internet protocol
* **ftp**: for the FTP Internet protocol
* Anything else isn't directly supported by elvis, but may be accessible via a proxy server.

server.domain.name
For the **http**: and **ftp**: protocols, this tells elvis which machine to contact on the Internet.

port
Both **http**: and **ftp**: have default port numbers (80 and 21 respectively), but if a given machine's server is listening at a non-standard port, you would give the port number here.

directory
This tells the server which directory contains the file.

file
This is the file to be loaded.

You can append a "#*name*" to the end of the URL. After loading the URL, elvis searches through its text for a <a name=*name*> tag -- even if the file isn't an HTML file.

You can also append a "?*expression*" to the end of an URL. The expression's meaning depends on the protocol. For "file:" and ":buffer", it is used as an ex line address (usually a line number or a regular expression), and the cursor will be moved to that line. This is used by the :browse command. For "http:", it is passed to the server and usually interpreted by a program residing there. For other protocols it has no meaning, and should not be used.

While in the "html" display mode, tag names are assumed to be URLs too. However, if a tag name lacks the protocol, site, port, or directory name then it inherits those properties from the current document. This is true *only* for tags, and *only* if you're in the "html" display mode. For example, if you're viewing "ftp://ftp.cs.pdx.edu/pub/elvis/README.html", and then give the command ":ta Announce.21d", then elvis will load the URL "ftp://ftp.cs.pdx.edu/pub/elvis/Announce.21d".

15.2 FTP

The name "FTP" stands for "File Transfer Protocol". It is a robust and versatile protocol for transferring data between different types of computers over the Internet, providing support for various formats or text or paged data.

Elvis always uses FTP's non-paged binary data format. Many files require this, and most other files will can be converted manually later if necessary.

If possible, when elvis is downloading a file via FTP it will display the file's size along with the number of bytes downloaded so far, so you can gauge how much longer the download will take. However, not all FTP servers support the "SIZE" command that elvis uses to learn the file's size, so sometimes elvis can't display the file's size while downloading. Also, elvis never displays the size of a directory while downloading.

Elvis converts FTP directory listings into HTML documents. For each file or subdirectory, elvis constructs a hypertext link from the directory to the file/subdirectory, so you can browse through an FTP file system using elvis' "html" display mode.

In addition to reading, elvis allows you to write via FTP. You can even append to a remote file via FTP, with a command such as...

```
:w >>ftp://ftp.somesite.com/incoming/filename
```

Unfortunately, the FTP protocol doesn't have a well-defined means for testing the attributes of a file. I tried to make elvis clever enough to infer the file type from the limited information that the FTP server can provide, but in some cases it may fail.

15.2.1 elvis.ftp or ~/.netrc

When using the World Wide Web, FTP accesses are normally anonymous. This allows you to read public files, which is all that a Web browser is expected to do anyway. On some sites, anonymous FTP may also allow you to write into a directory named "/incoming" but Web browsers don't use that facility.

Since elvis is an editor (not merely a browser), it has different requirements. You may wish to fetch a private file from your own account at some FTP site, modify it, and write it back again. You can't do that with anonymous FTP; elvis must therefore support user-specific FTP in addition to anonymous FTP.

To remain compatible with the Web, elvis normally uses anonymous FTP. If you want to access an FTP server using your own account, elvis requires you to give a directory name which begins with a "~". For example, "ftp://localhost/pub/myfile" refers to "pub/myfile" in the anonymous directory hierarchy, but "ftp://localhost/~/myfile" refers to "myfile" in your home directory using your own account's privileges. You can also put a user name after the tilde to access some other user’s account, as in "ftp://localhost/~bob/bobsfile".

Elvis doesn’t prompt for your password when you access an FTP site using a non-anonymous account, though. Instead, it searches for account information in the ".netrc" file in your home directory, or if that doesn’t exist then it searches for "elvis.ftp" anywhere in your elvispath. Regardless of the name, the file has the exact same format.

The file is divided into words. Some of these are keywords; others are data associated with the preceding keywords. Line breaks are no more significant than any other whitespace, but for the sake of readability it is a good idea to keep all data for a particular FTP site on a single line. To describe multiple accounts on a single FTP site, I suggest you put each account on a separate line.

A typical line will have a format like this...

    machine *server.domain.name* login *yourlogin* password *yourpass*

...where *server.domain.name* is the name of an FTP server, *yourlogin* is your login name there, and *yourpass* is your unencrypted password.

Warning! Because this file contains unencrypted passwords, you must be very careful to make this file unreadable by other users.

For more information about the ~/.netrc file, look in the Unix manual at the **ftp**(1) and **netrc**(5) man-pages. Elvis’ use of the ~/.netrc file is intended to be compatible with ftp’s use of the that file.

The first entry for a given FTP site is assumed to be your own account; it is used for accessing URLs which begin with a tilde but have no user name. For other accounts (with the user name after the tilde), elvis looks for the first entry in which the machine and user fields both match the values from the URL. This is a slight extension of ~/.netrc’s traditional meaning.

15.3 HTTP

The name "HTTP" stands for "HyperText Transfer Protocol". It is a light-weight protocol which is used mostly for fetching Web pages. "Light-weight" means that it handles small transfers efficiently, with very little overhead for initialization.

Elvis supports HTTP. In fact, elvis requires you to configure HTTP support if you want FTP support; you can't have FTP without HTTP.

Elvis doesn't support authentication or security. Authentication would embed your user name and password into each request you make; this would be useful for accessing subscription sites for things like stock market information, or an online encyclopedia. Security would encrypt each request; this would be useful when the request contains sensitive information such as your credit card number. Since elvis doesn't support these features, it can't replace a real Web browser.

HTTP is a read-only protocol. Consequently, you can't write to an "http:" URL.

15.3.1 Proxy Server

A proxy server is an HTTP server which allows you to send it a request for an URL which resides on a totally separate system. There are two reasons why you might want to do this.

The first reason is to get around a firewall. For security reasons, many Local Area Networks are configured to severely restrict the flow of data between the LAN and the Internet. This prevents you from accessing an Internet site directly. A proxy server may straddle that firewall, so it can receive requests from computers on the LAN, and send requests to Internet sites.

The second reason is that the URL you send to an HTTP proxy doesn't necessarily have to be an "http:" URL. It could use Gopher, WAIS, or whatever. Elvis can thus use its built-in HTTP support, and the services of an HTTP proxy, to support any protocol which the proxy supports.

15.3.2 elvis.net

The elvis.net configuration file tells elvis which requests must go through a proxy, and which can be sent directly. When a proxy is needed, it also gives the name of the proxy.

The file is divided into words. The word "direct" means that following words are domain names which can be accessed directly (without a proxy). The word "proxy" is followed by the name of an HTTP proxy server, and then by the names of domains which must use that proxy.

The domain names in this file can either be the complete names of servers ("www.othersite.com"), or a partial name which starts with a dot (".othersite.com") which matches all sites whose name has the same ending.

When elvis wants to read from a URL, it scans through the file and uses the first "proxy" or "direct" list which mentions the URL’s domain. If the domain doesn’t appear in any of list, then the last "proxy" or "direct" list is used; i.e., the last "proxy" or "direct" command is used as the default. As a simple example, if *all* requests must go through a proxy, then elvis.net would contain:

```
    proxy www.myproxy.com
```

When elvis is given a URL that uses any protocol other than "ftp:" or "http:" it searches through the elvis.net file for a proxy using the usual rules, except that if the URL’s domain isn’t mentioned in any list, then elvis will use the last proxy mentioned in the file (even if there is a "direct" list after that). If there is no proxy, then elvis gives an error message. For example, if you can access all sites directly, but also have access to a proxy which can fetch non-ftp/http URLs for you, then elvis.net should look like this:

```
    proxy www.myproxy.com
    direct
```

As a final example, here’s a file which causes all accesses to sites whose domain name ends with ".mylan.com" to be accessed directly, but all other sites accessed via a proxy. This is a typical set-up for a proxy which is used to reach out through a firewall.

```
    direct .mylan.com
    proxy www.myproxy.com
```

Note that all proxy accesses use the HTTP protocol, even if they are to FTP sites. This means that you can only *read* through a proxy; you can’t write through a proxy because HTTP is a read-only protocol.

The proxy accesses work by sending an HTTP "GET" request to the HTTP server, passing it the whole URL rather than just the directory and filename. The server must then be clever enough to parse the URL, and fetch the requested document from the proper site, using the proper protocol. Not all servers are that smart, so don’t be surprised if your closest server can’t handle proxy requests.

If elvis.net doesn’t exist, then all FTP or HTTP accesses will be direct, and other protocols will generate an error. The same behavior can also be generated by creating an elvis.net file containing only the word "direct".