

---

---

<b>CHAPTER 1</b>	<b>The Virtual Server</b>	<b>1</b>
1.1	Introduction to the Virtual Server System	1
1.2	How the Virtual Server System Works	4
1.2.1	Virtual Internet Services Offered	4
1.2.2	The Virtual Server Administrator	5
1.3	A Tour of the Virtual Server	5
<b>CHAPTER 2</b>	<b>The Virtual Web Service</b>	<b>7</b>
2.1	Introduction	7
2.2	Virtual Web Server Configuration Files	8
2.2.1	The httpd.conf File	8
2.2.2	srm.conf Parameters	10
2.2.3	access.conf Configuration File	11
2.3	Creating Web Pages	13
2.3.1	HTML Editors and Tools	13
2.4	Using the Common Gateway Interface (CGI)	13
2.4.1	Writing Your Own CGIs	14
2.4.2	Imagemaps	14
2.4.3	Simple Forms	16
2.4.4	Server Side Includes	17
2.5	Monitoring Your Virtual Web Server Log Files	18
2.6	Generating Automated Getstats Reports	22
2.7	Using WWW Stat as an Alternative to Getstats	23
2.8	Resetting Your Virtual Web Server Log Files	23
<b>CHAPTER 3</b>	<b>The Virtual FTP Service</b>	<b>25</b>
3.1	Introduction to FTP	25
3.2	Anonymous FTP vs. FTP	26
3.3	The Name of the Virtual FTP Service	26
3.4	Your Anonymous FTP Directory	26
3.5	Making an “incoming” Directory for Customers	26
3.6	Log In Banners and Directory Messages	27
3.7	Creating Non-Anonymous FTP Accounts	28
3.8	Monitoring Anonymous FTP Activity	30
<b>CHAPTER 4</b>	<b>The Virtual Email Service</b>	<b>31</b>
4.1	Introduction to Email	31
4.2	A Tour of the Virtual Email Handling System	32
4.3	Creating Email Aliases	32
4.4	Creating Email Lists	32

---

4.5	Creating Email Autoresponders	33
4.6	Error Messages	34
<b>CHAPTER 5</b>	<b>The Virtual POP Service</b>	<b>35</b>
5.1	Introduction to POP	35
5.2	Creating POP Accounts for Email Users	36
5.3	Changing a Mailbox Password	37
5.4	Removing a POP Email Account	37
5.5	Listing POP Email Account Users	37
5.6	Configuring the POP Client Software	38
5.6.1	Configuring Eudora	38
5.6.2	Configuring Microsoft's Internet Exchange™ and Netscape's Mail™	38
<b>CHAPTER 6</b>	<b>Helpful Commands and Useful Information</b>	<b>41</b>
6.1	Overview	41
6.2	The quota Command	41
6.3	The vdiskuse Command	42
6.3.1	Dead Processes Taking Up Disk Space	42
6.4	The vnukelog Command	43
6.5	The traceroute Command	43
6.6	The Contrib Directory	44
6.7	Creating Symbolic Links	45
<b>CHAPTER 7</b>	<b>The iManager Server Extension and iRoot Plug-in</b>	<b>47</b>
7.1	The iManager Extension	48
7.1.1	Installing the iManager Extension	48
7.1.2	Running the iManager Extension	48
7.1.3	Editing a File with iManager	49
7.1.4	Deleting a File with iManager	49
7.1.5	Copying a File with iManager	49
7.1.6	Moving a File with iManager	50
7.1.7	Linking a File with iManager	50
7.1.8	Changing the Permissions of a File with iManager	50
7.1.9	Uploading a New File to your Server with iManager	50
7.1.10	Making a New Directory with iManager	51
7.2	The iRoot Plug-in	51
7.2.1	Installing the iRoot Plug-in	51
7.2.2	Running the iRoot Plug-in	52
7.2.3	Adding Email and FTP Users - vadduser	52
7.2.4	Changing Email and FTP Users' Passwords - vpasswd	52
7.2.5	Removing Email and FTP Users - vrmuser	53
7.2.6	Adding an Email Alias	53
7.2.7	Deleting an Email Alias	53
7.2.8	Updating Your Aliases File - vnewaliases	53

- 
- 
- 7.2.9 Changing Your Server Root Password - passwd 53
  - 7.2.10 Creating and Installing a Digital Certificate for SSL Encryption 54

<b>APPENDIX A</b>	<b>An Overview of UNIX</b>	<b>55</b>
A.1	Overview	55
A.2	Why Use Unix?	55
A.3	Essential Commands and Concepts	56
A.3.1	Login	56
A.3.2	The Shell, Commands and Arguments	56
A.3.3	On-line Manuals	57
A.3.4	I/O re-direction: stdin, stdout, stderr, pipes	57
A.3.5	Special characters: Interrupt, End-Of-File, Quoting, 'Job Control'	59
A.3.6	Files, Permissions, Search PATH	60
A.4	The Unix Philosophy	61
A.5	A 'Typically Unix' Solution	61
A.6	More about Pipelines and Concurrent Execution	63
A.7	Other Especially Useful Unix Programs to Read About	63
A.8	Other Sources of Information	65



# The Virtual Server

---

## 1.1 Introduction to the Virtual Server System

---

Your personal virtual server will allow you establish an Internet presence with a high speed connection at a fraction of the cost. Normally, to establish a presence on the Internet would require you to purchase a high speed Internet connection, costly equipment, and even a support staff. With the Virtual Server System you get the Internet services you want without the worries and costs of an Internet connection and your customers will not be able to tell the difference.

Without Virtual Server technology you normally would have to do the following to establish a presence on the Internet for your company (see Figure 1):

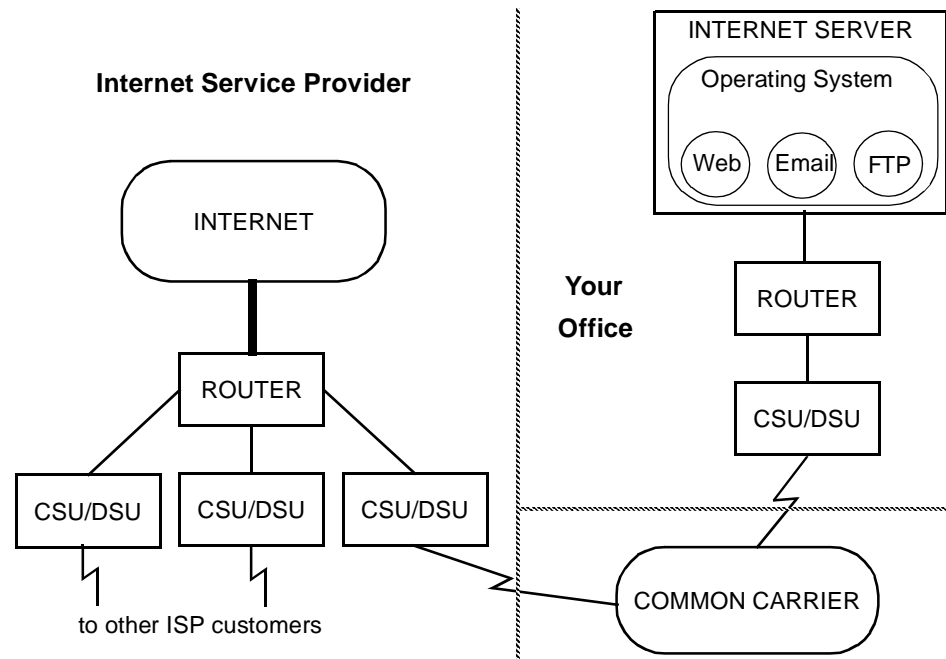
1. Purchase a lease line or frame-relay circuit from your local common carrier (such as US WEST) from your office to an Internet Service Provider (ISP).
2. Purchase a high speed CSU/DSU (a device similar to a MODEM) for your office and maybe a second CSU/DSU for your ISP.
3. Purchase a high speed router (a networking device used for converting the signal you get from the CSU/DSU into something your server can use) for your office and, again, maybe for your ISP.
4. Purchase Internet bandwidth from your ISP.

5. Purchase an Internet server computer for your office configured with a good amount of memory, disk storage and a tape backup system.
6. Purchase and install a server operating system (such as UNIX or Windows NT) for the server above.
7. Purchase and/or install web server software, FTP (see later chapter), and electronic mail software on the Internet server above.

---

FIGURE 1

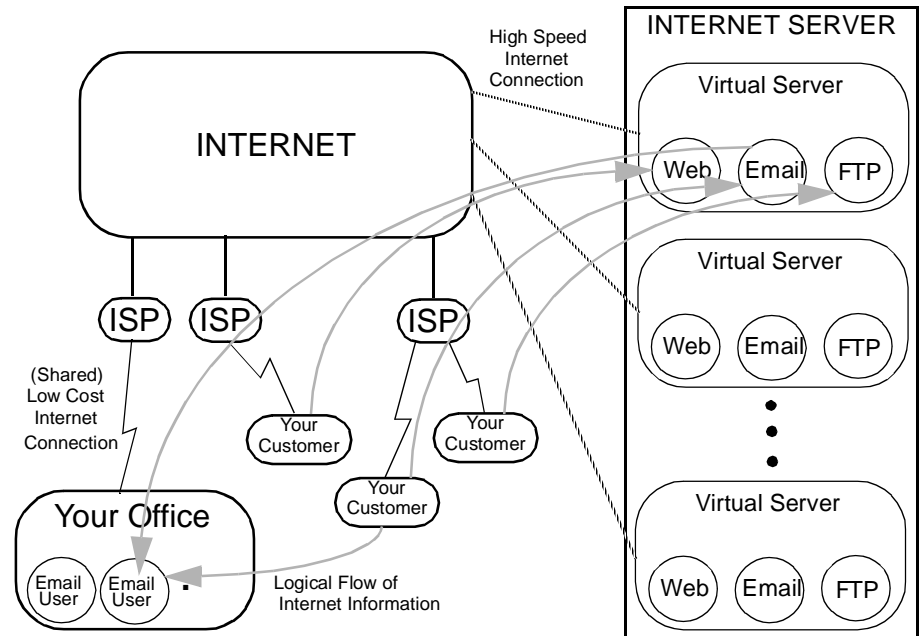
A Typical High Speed Internet Connection



Depending on the ISP, speed of the connection, equipment and software you buy it may cost between \$5,000 to \$10,000 just to establish a server on the Internet and between \$1,000 to \$20,000 a month for ISP and common carrier charges.

The Virtual Server System uses the idea of sharing a server and a high-speed Internet connection with other companies and, therefore, significantly reducing the cost of establishing an Internet presence (see Figure 2). Most companies that have high-speed connections do not use the full potential bandwidth of their Internet connection continuously. However, to avoid delays to their customers they will purchase the fastest Internet connection they can afford. For smaller companies this is usually a 56Kbit/sec or fractional T1 frame-relay (time-shared) connection. With slower connections customers may notice significant network delays during peak usage hours to these sites, however, customers accessing larger companies with faster connections (such as a full, dedicated T1 or better) will not experience these delays. With a virtual server you have access to a high-speed connection to the Internet at a fraction of the cost and your customers will not be able to tell the difference.

FIGURE 2 The Virtual Server System



For office electronic mail (Email) services the Virtual Server System can save your business money. Since each employee would have his or her own mailbox on the virtual server a small office can get away with only purchasing one or two dial-up accounts with an ISP and sharing these accounts. For example, consider a small office of 5-10 employees. For about \$20-\$30 per month individuals can purchase dial-up account that give them between 40-450 hours<sup>1</sup>. Rather than purchasing a dial-up accounts for each employee (normally what is required for individual mailboxes) you could purchase one or two accounts and configure the mail client software on each employee's computer to get their mail from the virtual server using the employee's Email username and password. The computers, however, would share the one or two dial-up accounts to get access to the Internet. For example, a ten employee company could save up to \$250 per month in just Internet access charges.

The Virtual Server System may not be for everyone. It allows a company to establish a high-speed Internet presence or test the potential of Internet marketing without incurring the high costs of bringing a high-speed connection and server to the office. In other words, it allows a company to test the Internet waters before committing high start up costs of equipment and without the sacrifice of starting with an inferior Internet service.

If your company after a period of time of being on a virtual server has developed a very popular web page (on the order of 30,000-50,000 hits or accesses a day) you will want to consider getting your own Internet connection and/or server to better support your

1. Prices vary with ISP and area.

customers' needs. We can help you if you want to have your own dedicated server offsite. Contact us for more information concerning these dedicated server products and services.

---

## 1.2 How the Virtual Server System Works

---

The Virtual Server System uses the idea of multiplexing a single UNIX server into multiple virtual machines. Each virtual machine (or virtual server) user will get their own set of virtual services that they may customize to their liking or needs. Each virtual server has its own unique domain name and IP (Internet Protocol) address. When a request comes in, the Virtual Server System will determine which virtual server will service the request based on this unique IP address and then invoke the appropriate service to respond to the request. Therefore, since the virtual server will invoke a unique server based on the designation address and port number of an IP request, each virtual server user may have his own custom configuration or data files for their set of virtual services. Also these virtual servers will log events to the user's own log directory so that they may make access accounting summaries and generate individual reports.

### 1.2.1 Virtual Internet Services Offered

The Virtual Server System currently offers the following Internet services:

1. **World Wide Web (WWW) or HTTP.** This gives your customers access to textual and graphical information about your company and services.
2. **Netscape Compatible Encryption.** This allows you to offer your customers secure pages on the World Wide Web. This is useful for online order forms requiring credit card information.
3. **Anonymous and Non-Anonymous File Transfer Protocol or FTP.** This gives your customers access to files that you may want to give them (demo programs, patches, documents, etc.). The non-anonymous option allows you to create specific FTP directories for users who want to update their own pages on your site. Non-anonymous FTP also allows you to create password protected FTP directories.
4. **Electronic Mail using Simple Mail Transfer Protocol or SMTP.** This allows you to exchange electronic messages with your customers.
5. **Post Office Protocol or POP.** This gives your employees their own electronic mailbox to individually access.

Other services are being added to the Virtual Server System and will be announced very soon. The following chapters will explain the above services and how to efficiently use them.

Please note that the virtual server does not support multiple telnet or shell accounts. However, it is possible to have individual mail and FTP accounts which are accessible by each user using the Post Office Protocol or POP and non-anonymous FTP. There is no limit to the number of these accounts (other than storage) and they would be created by your company's virtual server administrator.

### 1.2.2 The Virtual Server Administrator

The virtual server administrator is a user who is in charge of maintaining your virtual server. He or she will be given a username and a password to the virtual server administrator account. This username and password gives the administrator access to a normal shell account on a UNIX server which is home to your virtual server. The virtual server administrator responsibilities include:

- Adding or deleting virtual Email and FTP accounts.
- Adding or deleting Email aliases (forwarding addresses).
- Uploading or downloading files to the virtual anonymous FTP server.
- Maintenance of the HTML files of the virtual web server.
- Installation and maintenance of CGI (Common Gateway Interface) programs.
- Maintenance of virtual server log files.

It is important to assign someone as the virtual server administrator who has a little UNIX and programming experience (or who is at least willing to learn about UNIX and programming). Since each virtual server account is given full access to all the configuration files it is fairly easy to paint yourself into a corner<sup>1</sup>. However, it is much easier to maintain a virtual server system than a full UNIX system, since most of the work is done for you.

### 1.3 A Tour of the Virtual Server

---

The following is a short tour of the virtual server from the virtual server administrator's point of view. When you are given a virtual server account you will be given the following information about your server:

- **The IP (Internet Protocol) address of your virtual server.** This is usually something like 192.41.5.20 and is not important to remember since your unique domain name is an alias for this address. It may be useful, however, when testing your server while your domain name is being registered.
- **The virtual server administrator login name.** This is the name the administrator will use so he or she can modify and maintain the virtual server.
- **The virtual server administrator password.** This is the password associated with the login name above. You may change this password by using the UNIX *passwd* command. If you forget this password send Email to us.
- **The actual server host name.** This is the name of the actual machine where your virtual server is hosted.

---

1. As the old UNIX saying goes: "UNIX gives you enough rope to hang yourself plus a couple of extra feet."

Once you log in to the actual server as the virtual server administrator you will see the following directories:

- **bin** This directory contains many useful programs such as *sh* and *cat* for developing scripts for your virtual servers. It also contains the virtual server programs.
- **dev** This directory contains the device node *null* which is sometimes useful for scripts. It is also used by the virtual servers.
- **etc** This directory contains many configuration files associated with electronic mail (*sendmail* and *POP*) including the *aliases*, *passwd*, and the *sendmail.cf* files.
- **ftp** This directory contains what your customers will see when they use your virtual anonymous FTP services (more on this in Section 3.1 on page 25).
- **shlib** This directory contains all the shared library code for your virtual server program. If you don't know what shared libraries are then don't worry about it.
- **usr** The *usr* directory contains the following sub-directories:
  - bin** This is like the *bin* directory above. It contains additional support programs.
  - local/etc/httpd** This directory is your virtual *httpd* server's "root" (more on this in Section 2.1 on page 7).
  - log** This directory contains useful log "messages" from your virtual servers.
  - mail** This directory will contain any virtual mailboxes for accounts that you may have created (more on this in Section 5.2 on page 36).
  - spool** This directory is used by your virtual mail handler for temporary files.
- **www** This is not really a directory at all but rather a *symbolic link* to your *usr/local/etc/httpd* directory. This is useful for getting to your virtual *httpd*'s root directory quickly (from your home directory you can type *cd www* rather than *cd usr/local/etc/httpd*).

These directories represent your virtual server's *root* directory even though the command *pwd* (print working directory) reports something like */usr/home/<administrator\_login\_name>*. It is important to remember that the directory */usr/home/<administrator\_login\_name>* gets translated to simply */* when any of the virtual services (such as *httpd*, *sendmail*, *pop*, or *ftpd*) run. For example, when you develop CGI's (Common Gateway Interfaces) for your virtual *httpd* you need to use the path */usr/local/etc/httpd/* rather than the path */usr/home/<administrator\_login\_name>/usr/local/etc/httpd/*.

# The Virtual Web Service

---

## 2.1 Introduction

---

The World Wide Web project has taken the Internet by storm in spite of earlier criticism. A World Wide Web (WWW, or Web) service makes accessing your information easier for your customers. WWW services provide all the power your company needs to make exciting and dynamic web pages. Accessing WWW services has been a problem because transferring the large graphical files over slow connections was intolerable. Now with the newer MODEM's (14.4 to 28.8 Kbps) and the cheaper rates for SLIP (Serial Line Internet Protocol) or PPP (Point-to-Point Protocol), Internet connection graphical interfaces are available to even individual home users. That means all home users can access your company's WWW service with a small investment for equipment and service and that is good news for your business.

The virtual web server is based on the NCSA HTTP (Hyper-Text Transfer Protocol) server<sup>1</sup>. The httpd files are located in *usr/local/etc/httpd* in your home directory. Please note that the directory *www* in your directory is a symbolic link to the directory *usr/local/etc/httpd*. Therefore, a quick way to get into the *usr/local/etc/httpd* is to simply do

---

1. See <http://hoohoo.ncsa.uiuc.edu/docs/Overview.html> for more information about NCSA's httpd server.

a `cd www` from your home directory. The following sub-directories are located within your `usr/local/etc/httpd` directory:

- **cgi-bin** This directory contains some CGI (Common Gateway Interface) programs that you may use with your web pages. Of course, you may install others that you find elsewhere or develop yourself.
- **cgi-src** This directory contains the source code for the CGI's found in the `cgi-bin` directory above.
- **conf** This directory contains configuration files for the virtual web server. These files are described in detail in the section below.
- **htdocs** This directory contains your initial home page (i.e. `index.html`) and other html and graphic files.
- **logs** This directory contains various log files. Section 2.5 on page 18 describes how to use these files.

---

## 2.2 Virtual Web Server Configuration Files

---

The virtual web server configuration files are pre-configured for you when you get a virtual server account, however, you may need to customize these files a little to take advantage of some advanced features. The following sections give an overview of the contents of these files. For most web pages you should not have to change these files from the Internet Server default setup. Therefore, you may want to quickly scan these sections and refer back to them when needed.

### 2.2.1 The `httpd.conf` File

The `httpd.conf` file is for general configuration of the virtual `httpd`. Table 1 below describes some of the parameters in `httpd.conf`.

---

TABLE 1.

`httpd.conf` Parameters

Parameter	Description
<code>AccessConfig filename</code>	<i>filename</i> is either an absolute pathname or a partial pathname relative to <b>ServerRoot</b> that specifies the location of the <code>access.conf</code> configuration file. The default is <b>AccessConfig conf/access.conf</b> .
<code>AgentLog filename</code>	<i>filename</i> is the file where you want to keep record of the client agent software. The directive is for statistical purposes and tracing of protocol violations. The default is <b>AgentLog logs/agent_log</b> .
<code>DNSMode level</code>	<i>level</i> is the level of DNS resolution the server does on every request. The options are <i>none</i> , <i>mimumum</i> , <i>standard</i> , and <i>maximum</i> . The default is <b>DNSMode none</b> . You can change this to any of the other options, but your server will not run as fast.

---

**TABLE 1.**

httpd.conf Parameters

Parameter	Description
ErrorLog <i>filename</i>	<i>filename</i> is either an absolute path name or a partial path name relative to ServerRoot that specifies the location of the error log file. The virtual web server includes information such as segmentation violations, bad scripts, timed out clients, and .htaccess files that attempt to default access.conf directives. The default is <b>ErrorLog logs/error_log</b> .
IdentityCheck [on   off]	Determines if the remote user is logged in as himself. This directive only works if the client application is running an RFC 931-compliant identity daemon (unlikely). The default is <b>IdentityCheck off</b> .
ResourceConfig <i>filename</i>	<i>filename</i> is either an absolute path name or a partial path name relative to ServerRoot that specifies the location of the srm.conf configuration file. The default is <b>ResourceConfig conf/srm.conf</b> .
ServerAdmin <i>email_address</i>	<i>email_address</i> specifies the webmaster's address. The default is <b>ServerAdmin webmaster@&lt;your domain-name&gt;</b> .
ServerName <i>hostname</i>	<i>hostname</i> specifies the domain name of your server or a DNS alias. The default is <b>ServerNameName www.&lt;your domainname&gt;</b> .
ServerRoot <i>pathname</i>	<i>pathname</i> defines the absolute path of the root of your web server above which users cannot trespass. The default is <b>ServerRoot /usr/local/etc/httpd</b> . (Remember that the path /usr/home/<login name> gets translated to simply / when the virtual server runs.)
TimeOut <i>seconds</i>	<i>seconds</i> defines the maximum amount of time (in seconds) the service waits for the client to submit a request once it has been connected, and the maximum amount of time the service should wait for the client to accept a request. The default is <b>TimeOut 1800</b> .
TransferLog <i>filename</i>	<i>filename</i> is either an absolute path name or a partial path name relative to ServerRoot that specifies the location of the log that records data of service requests, such as host, date, and file name. The default is <b>TransferLog logs/access_log</b> .
TypesConfig <i>filename</i>	<i>filename</i> is either an absolute path name or a partial path name relative to ServerRoot that specifies the location of the MIME configuration file. The default is <b>TypeConfig conf/mime.conf</b> .

### 2.2.2 srm.conf Parameters

The srm.conf (server resource management) configuration file specifies the location in which the service finds your scripts and documents. The following table describes the parameters found in the srm.conf.

TABLE 2.

srm.conf Parameters

Parameter	Description
AccessFileName <i>filename</i>	<i>filename</i> specifies the name of the file that you can include in any directory that specifies access permissions for that directory. The default is <b>AccessFileName .htaccess</b> .
AddDescription <i>text fileID</i>	Associates descriptive <i>text</i> with a type of file defined by extensions, a file name, an absolute path name, or a file name using wild cards (for example <b>AddDescription "image file" *.gif</b> ).
AddEncoding <i>kind ext</i>	Specifies that files with <i>ext</i> are of type <i>kind</i> so that appropriate actions can be taken. For example, if the file is compressed, the browser can automatically uncompress it (for example, <b>AddEncoding compress Z</b> ).
AddIcon <i>path name1 name2...</i>	Specifies the icon to display with a kind of file; used when browsers display FTP menus.
AddIconbyEncoding <i>path name1 ...</i>	Performs the same task as AddIcon except that the encoded information determines the icon used.
AddIconType <i>path type1 type2 ...</i>	Performs the same task as AddIcon except that the MIME type determines the icon used.
AddType <i>kind ext</i>	Supersedes MIME definitions for the specified extensions ( <i>ext</i> ) found in the mime.types file.
Alias <i>name path</i>	Substitutes <i>path</i> for <i>name</i> in path names.
DefaultType <i>type</i>	Specifies the default MIME type. The default is <b>DefaultType text/html</b> .
DefaultIcon <i>pathname</i>	<i>pathname</i> specifies the default icon to use when Fancy-Indexing is on. The default is <b>DefaultIcon /icon/unknown.xbm</b> .
DirectoryIndex <i>filename</i>	Specifies the <i>filename</i> to return when the URL request does not specify a file or the request is just your service (for example, http://www.domain.com). The default is <b>DirectoryIndex index.html</b> .

**TABLE 2.**

srm.conf Parameters

<b>Parameter</b>	<b>Description</b>
DocumentRoot <i>path</i>	<i>path</i> specifies the absolute path to the directory from which the virtual web server retrieves documents. The default is <b>DocumentRoot /usr/local/etc/httpd/htdocs</b> . (Remember when the virtual web server runs the path /usr/home/<your login name> will become simply /.)
FancyIndexing [on   off]	Adds icons, file name data, headers, and footers to lists of files automatically indexed. The default is <b>FancyIndexing on</b> .
HeaderName <i>filename</i>	Specifies the <i>filename</i> to be used at the top of a list of files automatically indexed. The default is <b>HeaderName HEADER</b> .
IndexIgnore <i>kind1 kind2 ...</i>	Specifies kinds of files to be ignored during file processing. The default is <b>IndexIgnore */.* *~ *# */HEADER* */README</b> .
IndexOptions <i>option1 option2 ...</i>	Specifies a variety of indexing parameters, including FancyIndexing, IconsAreLinks, ScanHTMLTitles, SuppressLastModified, SuppressSize, and SuppressDescription.
OldScriptAlias <i>name path</i>	Performs the same task as Alias (backward compatibility for HTTP V1.0).
ReadmeName <i>filename</i>	<i>filename</i> specifies the footer information to attach to automatic directory indexes. The default is <b>ReadmeName README</b> .
Redirect <i>pathname URL</i>	Remaps <i>pathname</i> of document to new <i>URL</i> . There is no default for this directive.
ScriptAlias <i>name path</i>	Is similar to Alias but used for scripts. This directive substitutes <i>path</i> for <i>name</i> in path names.

### 2.2.3 access.conf Configuration File

The access.conf configuration files define what service features are available to all WWW browsers. The default is to make everything available to all browsers. Many of the parameters in the access.conf are sectioning directives. They stand out because they use angle brackets.

Sectioning directives have a beginning and ending delimiter, for example:

```
<directory /usr/local/etc/httpd/cgi-bin>
  AllowOverride Limit
</directory>
```

The above example enforces extra restrictions on the directory */usr/local/etc/httpd/cgi-bin* using the AllowOverride directive. The AllowOverride directive may use one of the following directives:

- **All.** Access control files are unrestricted in this directory.
- **AuthConfig.** Enables the use of AuthName, AuthType, AuthUserFile, and AuthGroupFile directives.
- **FileInfo.** Enables the use of AddType and AddEncoding directives.
- **Limit.** Enables the use of the limit sectioning directive.
- **None.** No access control files are allowed in this directory.
- **Options.** Enables the use of the Options directive.

The directives enabled by the AuthConfig are defined as the following:

- **AuthName.** Sets the authorization name of the directory.
- **AuthType.** Sets the authorization type of this directory. Currently, there is only one type: Basic.
- **AuthUserFile.** Specifies the file to use that contains the list of users and passwords used in user authentication.
- **AuthGroupFile.** Specifies the file that lists user groups for user authentication.

The directives enabled by Options include the following:

- **All.** All features are enabled for the directory.
- **ExecCGI.** CGI scripts can be executed in this directory.
- **FollowSymLinks.** Allows the server to follow symbolic links.
- **Includes.** Server-side include files are enabled in this directory.
- **IncludesNoExec.** Enables server-side include, but disables the exec server-side include command.
- **None.** No features are enabled for the directory.
- **SymLinksIfOwnerMatch.** The server only follows symbolic links if the target file or directory is owned by the same user ID as the link.

Limit sectioning directive may include the following:

- **allow *hostname*.** Allows specified hosts to access the service.
- **deny *hostname*.** Prevents specified hosts from accessing the service.
- **order *ordering*.** Determines the order in which the allow and deny directives are evaluated. Customary values are “deny,allow” and “allow,deny”.
- **require *entity1 entity2...*** Entity values can be user, group, or valid-user. These are the authenticated users or groups that can access the system. Valid-users are users identified by AuthUserFile.

Any directive between the delimiters apply to the listing following the first delimiter. For example:

```
<Limit GET>
order allow, deny
allow from all
</limit>
```

In this example, the sectioning directive, Limit, determines who can retrieve information from the service, which, in this case is “allow from all.”

### 2.3 Creating Web Pages

---

The default homepage for your virtual server account is located in the file `usr/local/etc/httpd/htdocs/index.html` or, simply, `www/htdocs/index.html`. Your virtual server account includes access to many popular UNIX editors including *pico*, *vi*, *emacs* and others. You can also edit your web pages on your PC and upload them to the virtual server.

If you do FTP your home page to the server, *make sure you FTP in ASCII mode and not Binary mode*. All text files, including home pages should be FTP'd in ASCII mode.

The default page for each subdirectory of your web site is `index.html`. If you add any directories and want a page to default for that directory, name it `index.html`. For example, if you create a directory called `test` under your `htdocs` directory, then place a file called `index.html` in the `test` directory, that will be the page that comes up when people enter `http://www.yourdomain.com/test/`.

The details of HTML is beyond the scope of this document. however, there are many good tutorials and references available both on-line and in book form. For an extensive list, go to:

[http://www.yahoo.com/Computers\\_and\\_Internet/Software/Data\\_Formats/HTML/](http://www.yahoo.com/Computers_and_Internet/Software/Data_Formats/HTML/)

#### 2.3.1 HTML Editors and Tools

HTML editors help you quickly create web pages for your server. Some are WYSIWYG (What You See Is What You Get) type editors and help you get your page just right. For a list of HTML Editors go to:

[http://www.yahoo.com/Computers\\_and\\_Internet/Software/Internet/World\\_Wide\\_Web/HTML\\_Editors/](http://www.yahoo.com/Computers_and_Internet/Software/Internet/World_Wide_Web/HTML_Editors/)

For free graphics, icons, and backgrounds go to:

[http://www.yahoo.com/Computers\\_and\\_Internet/Graphics/](http://www.yahoo.com/Computers_and_Internet/Graphics/)

### 2.4 Using the Common Gateway Interface (CGI)

---

The Common Gateway Interface (CGI) allows you to extend your virtual server. CGI allows you to support the situations when the client wants to send information to the virtual server for more complicated processing. In general, web servers don't process information themselves but rather hand off the work to gateway programs. The CGI specification defines the mechanisms by which HTTP servers communicate with gateway programs. Therefore, you need to understand the HTTP protocol and the CGI specification to write server-side gateway programs and client HTML documents that use these programs.

Advanced CGI development is beyond the scope of this document. Some example CGIs are described in later sections, however, for more information about CGIs see:

[http://www.yahoo.com/Computers\\_and\\_Internet/Internet/World\\_Wide\\_Web/CGI\\_\\_Common\\_Gateway\\_Interface/](http://www.yahoo.com/Computers_and_Internet/Internet/World_Wide_Web/CGI__Common_Gateway_Interface/)

### 2.4.1 Writing Your Own CGIs

You can also write your own CGI using the following languages:

- C/C++
- **sh** (bourne shell)
- **cs**
- **perl**

Please keep in mind when you write your own CGIs that your virtual server's home directory (`/usr/home/<your login name>/`) becomes the root directory (`/`). Therefore, you do not need to include `/usr/home/<your login name>` in the full, absolute paths in your scripts or programs. You can test your CGIs interactively with the *virtual* command. For example:

```
Salmon: {4} % virtual mycgi arg1 arg2 arg3
```

In this example, the program *mycgi* runs in the virtual environment and is given the arguments *arg1*, *arg2*, and *arg3*. The virtual environment is the environment that your cgi's would run under if they were called from the web. If you simply ran your cgi without the virtual command, it would not run the same as if it were being run from your web pages.

### 2.4.2 Imagemaps

Active images, or clickable images, allow your users to click on different areas of an image and have different things happen, depending on where they clicked. For example, the active image could be a map of a building and clicking on a room will take you to an information page on that room.

To create an active image follow the steps below:

1. Create an image that you want to make an active image. You must use the GIF format for the image.
2. Include the image in your HTML document using the IMG element with the ISMAP attribute. Make this image a link to the imagemap CGI with the name of your map file on the end ("my\_mapfile" in the example below). For example:

```
<A HREF="http://some.site.com/cgi-bin/imagemap/my_mapfile">  
<IMG SRC="image.gif" ISMAP>  
</A>
```

3. In your `www/conf` directory create the file *imagemap.conf* with the full path to your map file. For example:

```
my_mapfile: /usr/local/etc/httpd/htdocs/my_mapfile.map
```

Note: A small bug in the imagemap program requires you to have a line return at the end of this file. Therefore, when you are editing *imagemap.conf*, be sure to hit the “enter” key when you are done with the last line.

You can add other mappings in *imagemap.conf* by just adding another line. For example:

```
my_mapfile: /usr/local/etc/httpd/htdocs/my_mapfile.map
my_2ndmap: /usr/local/etc/httpd/htdocs/my_2ndmap.map
```

4. Create the map file in your *usr/local/etc/httpd/htdocs* (or *www/htdocs*) directory. Generating a map file is not hard, however, you will need some kind of graphic editor that gives the X,Y coordinates of an image. Some editors that work for this are:
  - Adobe Photoshop** A commercial graphics program available from most software stores.
  - Paintshop Pro** A very good shareware program from <ftp://gatekeeper.dec.com/pub/micro/msdos/win3/desktop/psp30.zip>
  - Mapedit** A freeware program that directly creates map files from <ftp://sunsite.unc.edu/pub/packages/infosystems/WWW/tools/mapedit> (See <http://sunsite.unc.edu/boutell/mapedit/mapedit.html> for more information.)

The general form for a map file entry is:

```
method URL x1,y1 x2,y2 ... xn yn
```

where method specifies the manner in which the region is being specified (circle, rect, poly, point, or default), the URL is the page you want to appear when someone clicks within the region, and xn,yn are integer coordinates of defining the region starting with a point measured from the left-hand corner of the image. You will need to get these points from one of the tools mentioned above or some other way. The following is an example of what the map file should look like:

```
# Map file (htdocs/my_mapfile.map) for image.gif
circle /room1.html 50,20 50,30
rect /room2.html 25,78 40,85
poly /room3.html 45,38 35,50 40,72 50,75 60,72 65,50 55,38
point /room4.html 10,20
point /room4.html 11,20
point /room4.html 10,21
point /room4.html 11,21
default /default.html
```

Here is how each of the different methods of defining clickable regions work:

**circle** <URL> <center point> <edge point> This maps the region inside the circle to the <URL>. <center point> is the X,Y coordinate for the center and <edge point> is the X,Y coordinate for a point on the edge of the circle.

**point** <URL> <X,Y> This maps a single point <X,Y> to the <URL>. This is often used with other point methods for small areas as in the example above.

**poly** <URL> <X1,Y1>, <X2, Y2>, ... <Xn,Yn> This maps the region defined by a multisided polygon to the given <URL>. The polygon is automatically closed with a line from the last point <Xn,Yn> to the first point <X1, Y1>.

**rect** <URL> <Upper left Corner> <Lower right corner> This maps the region inside a rectangle defined by <Upper left corner> and <Lower right corner> to the <URL>.

**default** <URL> This maps the region not defined by any of the above within an image to the <URL>.

5. After the map file(s) are in place your active image should work; so give it a try.

### 2.4.3 Simple Forms

It is often useful to have your customers give you feedback (hopefully in the form of more product orders). The Common Gateway Interface can be used to collect this information. This section shows one way to do this with a simple form processing program.

The *formmail.pl* program is a perl script that processes information that a user enters into an on-line form and then sends that information to an Email address you specify. To set up an on-line form using *formmail.pl* do the following:

1. Copy *formmail.pl* from */usr/local/contrib/formmail.pl* to your *www/cgi-bin* directory and change its permission so it is executeable

**Note:** The */usr/local/contrib* directory is not in your directory structure. it is different from your *usr/local* directory. Follow these instructions *exactly* in a telnet session to copy the correct files. For more information on the Contrib directory see “The Contrib Directory” on page 44:

```
cp /usr/local/contrib/formmail.pl ~/www/cgi-bin/formmail.pl
chmod 755 ~/www/cgi-bin/formmail.pl
```

2. Create a HTML document with the following in the top part of the *BODY* section of the document:

```
<FORM ACTION="/cgi-bin/formmail.pl" METHOD="POST">
<INPUT TYPE=hidden NAME="recipient" value="you@youraddress.com">
<INPUT TYPE=hidden NAME="subject" value="Customer Feedback">
```

In the example above, the line with NAME equal to *recipient* specifies who the mail is sent to, *subject* specifies what you want the subject of your mail to be.

3. After the section above, create the form that you want your customers to see and close it with *</FORM>*. For example:

```
<PRE>
Name:
<INPUT NAME="Name" SIZE="40" MAXLENGTH="40">
Email:
<INPUT NAME="Email" SIZE="40" MAXLENGTH="40">
Address:
<INPUT NAME="Address" SIZE="60" MAXLENGTH="60">
City:
<INPUT NAME="City" SIZE="20" MAXLENGTH="30">
State: Zip:
<INPUT NAME="State" SIZE="2" MAXLENGTH="15"> <INPUT NAME="Zip"
SIZE="7" MAXLENGTH="7">
Phone #:
<INPUT NAME="Phone #" SIZE="14" MAXLENGTH="14">
Comments:
```

```
<TEXTAREA NAME="Comments" ROWS="12" COLS="48"></TEXTAREA>
</PRE>
<INPUT TYPE="SUBMIT" NAME="Request"><INPUT TYPE="RESET">
</FORM>
```

4. Try out your form.

#### 2.4.4 Server Side Includes

If you have been “surfing” the Web much you may have noticed that some web pages have dynamic information that changes every time you access the page such as the number of times the page has been accessed, the current date and time and so forth. This feature is called Server Side Includes or Server Includes. This section, which is the final CGI example, shows how to implement a Server Side Include to display the number of “hits” on a web page. This is done with a little perl script written by Jonathan Lewis called “c4.pl”. To setup a web page hit counter do the following:

1. Copy *c4.pl* and *lock.pl* from the */usr/local/contrib/* directory to your *www/cgi-bin* directory and make sure they are executable.

**Note:** The */usr/local/contrib* directory is not in your directory structure. It is different from your *usr/local* directory. Follow these instructions *exactly* in a telnet session to copy the correct files. For more information on the Contrib directory see “The Contrib Directory” on page 44:

```
Salmon: {4} % cp /usr/local/contrib/c4.pl ~/www/cgi-bin
Salmon: {5} % cp /usr/local/contrib/lock.pl ~/www/cgi-bin
Salmon: {6} % chmod 755 ~/www/cgi-bin/c4.pl
Salmon: {7} % chmod 755 ~/www/cgi-bin/lock.pl
```

2. Create a file called *.htaccess* in the same directory that you have the HTML for the web page you plan to have the counter on with the following in it:

```
Options Indexes FollowSymLinks Includes
AddType application/x-httpd-cgi .cgi
AddType text/x-server-parsed-html .html
```

If you are also using files with the *.htm* extension and want to have server side includes on those pages, be sure to add this line to the above lines of your *.htaccess* file:

```
AddType test/x-server-parsed-html .htm
```

3. Create a file with the same “basename” as your *.html* file that you plan to put the counter on, but with the extension *.count*, rather than *.html*. For example, for *index.html*, I would create a file by the name of *index.count*. Put this file in the same directory as *index.html*. In this file put the beginning count in it (i.e. “0”). An easy way to do this step is as follows:

```
Salmon: {9} % cat > index.count
0
^D (the control-D character)
```

4. In your HTML file put something like the following in the BODY section:  
**Over <!--#exec cgi="/cgi-bin/c4.pl" --> Internet Customers Served.**

5. Reload your web page and see if it works.

---

## 2.5 Monitoring Your Virtual Web Server Log Files

---

It is useful to monitor your virtual server's web usage. This is a good way to get feedback on how your virtual server is being used by potential customers. The virtual web server log files are kept in the `www/logs` directory, however, you will want to use a program to digest this information into something useful. The program `getstats`<sup>1</sup> can help you do this. `Getstats` can be used either interactively (from the command line) or periodically in "batch mode" using `cron` (see "Generating Automated Getstats Reports" on page 22). For interactive mode just type the `getstats` command at the Unix prompt with the appropriate report option(s). For example:

```
Salmon: {4} % getstats <report option>
```

Currently there are twelve major types of reports this program can produce. You can use as many options as you like to create combinations of reports. The following is some of the type of reports that can be generated using `getstats`:

### 1. `getstats -c` (concise report):

```
HTTP Server General Statistics
Local date: Fri Feb 11 18:17:07 PM PST 1994
Covers: 02/09/94 to 02/11/94 (3 days).
All dates are in local time.
Requests last 7 days: 4495
New unique hosts last 7 days: 358
Total unique hosts: 358
Number of HTML requests: 1854
Number of script requests: 472
Number of non-HTML requests: 2169
Number of malformed requests (all dates): 5
Total number of all requests/errors: 4500
Average requests/hour: 90.2, requests/day: 2164.7
Running time: 11 seconds.
```

This basic set of statistics is always output when `getstats` runs. Using the `-c` option will only produce this statistics paragraph.

### 2. `getstats -m` (monthly report):

```
HTTP Server Monthly Statistics
Covers: 10/30/93 to 11/08/93 (9 days).
All dates are in local time.
```

```
Each mark (#) represents 1000 requests.
```

```
-----
```

```
Oct (10/30/93): 569 : #
```

---

1. `Getstats` is written by Kevin Hughes (kevinh@eit.com). See <http://www.eit.com/software/getstats/getstats.html> for more information.

Nov (11/04/93): 2 :

...

The **-m** option will produce a monthly report of server use. The dates in the report are the first day of reported activity for that month.

**3. getstats -w (weekly report):**

HTTP Server Weekly Statistics

Covers: 12/28/93 to 01/27/94 (32 days).

All dates are in local time.

Each mark (#) represents 500 requests.

-----

Week of 12/27/93: 1878 : ###

Week of 01/03/94: 5606 : #####

Week of 01/10/94: 23287 : #####

...

The **-w** option will produce a weekly report of server use. The dates in the report are always the Monday of that particular week.

**4. getstats -ds (daily summary):**

HTTP Server Daily Summary

Covers: 12/28/93 to 01/27/94 (32 days).

All dates are in local time.

Each mark (#) represents 1000 requests.

-----

Mon: 16018 : #####

Tue: 13219 : #####

Wed: 9904 : #####

...

The **-ds** option produces a daily summary, which shows the aggregate number of requests for a particular day of the week.

**5. getstats -d (daily report):**

HTTP Server Daily Statistics

Covers: 12/28/93 to 01/27/94 (32 days).

All dates are in local time.

Each mark (#) represents 100 requests.

-----

12/28/93 (Tue): 88 :

12/29/93 (Wed): 258 : ##

12/30/93 (Thu): 591 : #####

12/31/93 (Fri): 775 : #####

...

The **-d** option produces a daily report, which shows the number of requests per day and the date.

**6. getstats -hs (hourly summary):**

HTTP Server Hourly Summary

Covers: 12/28/93 to 01/27/94 (32 days).

All dates are in local time.

Each mark (#) represents 200 requests.

```
-----  
midnite: 1266 : #####  
1:00am: 1206 : #####  
2:00am: 1238 : #####  
...
```

The **-hs** option produces an hourly summary, which shows the aggregate number of requests for a particular hour.

**7. getstats -h (hourly report):**

HTTP Server Hourly Statistics  
Covers: 12/28/93 to 01/27/94 (32 days).  
All dates are in local time.

```
-----  
12/28/93 (Tue)  
  
3:00 pm: 39 : #  
4:00 pm: 12 :  
5:00 pm: 36 : #  
...
```

The **-h** option produces an hourly report, which shows the number of requests per hour, the day of the week, and the total number of requests for each day.

**8. getstats -f (full report):**

HTTP Server Full Statistics  
Sorted by number of requests.  
Covers: 12/28/93 to 01/27/94 (32 days).  
All dates are in local time.

```
# of Requests : Last Access (M/D/Y) : Hostname  
-----  
  
6994 : 01/26/94 : kmac  
1751 : 01/26/94 : eitech  
1096 : 01/27/94 : jhvh-1  
...
```

The **-f** option tells getstats to create a full report sorted by host name (and IP address). Use the **-fa** option to make a full report sorted by the number of accesses, the **-fd** option to create a full report sorted by the last access date, or the **-fb** option to create a full report sorted by the number of bytes transferred.

**9. getstats -r (request report):**

HTTP Server Request Statistics  
Sorted by number of requests, 1560 unique requests.  
Covers: 12/28/93 to 01/27/94 (32 days).

All dates are in local time.

# of requests : Last Access (M/D/Y) : Request

-----  
4260 : 01/27/94 : /eit.home.html  
3330 : 01/27/94 : /graphics/stripes.bottom.gif  
2831 : 01/27/94 : /graphics/ball.black.gif  
...

The **-r** option tells getstats to create a report of requests sorted by the request name. Use the **-ra** option to sort by accesses, **-rd** to sort by the last access time, **-rb** to sort by the number of bytes transferred, and **-rf** to sort by individual file sizes.

### 10. getstats -dn (domain report):

HTTP Server Domain Statistics  
1 level, sorted by domain name, 22 unique domains.  
Covers: 02/09/94 to 02/10/94 (2 days).  
All dates are in local time.

# reqs : # uniq : Last Access (M/D/Y) : Domain

-----  
180 : 28 : 02/10/94 : (numerical domains)  
27 : 1 : 02/10/94 : .at  
28 : 3 : 02/10/94 : .au  
22 : 2 : 02/10/94 : .ca  
...

The **-dn** option generates a domain report, sorted by domain name. Use **-da** to sort by the number of requests, **-dd** to sort by last access date, **-db** to sort by the number of bytes transferred, or **-du** to sort by the number of unique domains. The unique domain number is the total number of unique sites under a domain. In the example above, for instance, a total of 3 unique sites came from the .au domain.

### 11. getstats -dt (directory tree report):

HTTP Server Tree Report  
Covers: 12/28/93 to 01/07/94 (12 days).  
All dates are in local time.

# of Requests : Last Access (M/D/Y) : Dir/File

-----  
55 : 01/07/94 : /reports  
51 : 01/07/94 : /ht93  
562 : 01/07/94 : /demos  
487 : 01/07/94 : /asiceda  
...

The **-dt** option generates a directory tree report, which cannot be sorted. The number of requests and last request date for directories and files is displayed. The request count for directories is the amount of requests for that directory plus the sum of all requests for the files and subdirectories under it.

If you find this report is empty, try using **getstats -dr "/www/htdocs/" -dt**.

For a report of specific directories, try the **getstats -sr "<dirname>/\*" -d** report. In this report *-sr* stands for search string, *<dirname>* would be replaced with your directory structure under your *www/htdocs* directory, "\*" is a wildcard for all files within that directory structure, and *-d* is the daily report option.

**12. getstats -e (file) (error report):**

HTTP Server Error Report (All Dates)

-----

```
kmac [Thu Dec 30 23:20:21 1993] get / foo
kmac [Thu Dec 30 23:20:37 1993] get foo /
kmac [Thu Dec 30 23:20:55 1993] get http://www.eit.com/ foo
```

**-e** generates a report of all malformed (or ignored) requests for all dates in the order they were encountered in the log file. If a filename is given as the argument to the option, bad requests will be appended to an error file, where they can be analyzed later.

**13. getstats -a (all reports):**

The **-a** option will produce all of the above reports, with list reports sorted by the number of accesses, if possible. If you want a report sorted another way, however, specify the correct option after the **-a** flag.

example: `getstats -a -fb`

This will create all reports sorted by number of requests, with the exception of the full report, which is sorted by byte traffic, and the error report, which must be specified on the command line.

---

## 2.6 Generating Automated Getstats Reports

---

You should set up a cron tab to automatically compute the daily statistics and send you a daily report, a weekly report, and a monthly report.

Additionally, you should "Nuke" the getstats log file at the start of every month to free up more disk space. This cronfile will do this for you, or you can use the *vnukelog* command (see "The vnukelog Command" on page 43)

1. Store this three line file in your home directory in a file called *cronfile* for example. Make sure it is only three lines. If the lines are long, let them wrap, but *do not add a hard return*:

```
58 23 * * * /usr/local/bin/getstats -d -f | /usr/bin/mail -s "Web Daily Stats" stats@yourdomain.com
59 23 * * 7 /usr/local/bin/getstats -w -f | /usr/bin/mail -s "Web Weekly Stats" stats@yourdomain.com
01 00 1 * * /usr/local/bin/getstats -w -f -n | /usr/bin/mail -s "Web Monthly Stats" stats@yourdomain.com
```

2. Run *crontab* to install the cronfile by typing *crontab cronfile* at a telnet prompt.
3. The first line will send a full daily report to *stats@yourdomain.com* each day at 23:58 (11:58 pm). *Of course you must change this Email address to yours.*

The second line will send a full weekly report at the end of each week at 23:59 (11:59 pm).

The third line will send a full monthly report and “nuke” (-n) the log file at 00:01 (12:01 am) on the first day of each month.

The “-f” specifies a full report. If you do not want a full report, you can change the report settings to your liking.

For more information on cron type *man crontab* and *man 5 crontab* at the virtual server’s UNIX prompt.

---

## 2.7 Using WWW Stat as an Alternative to Getstats

---

If you do not like the getstats program, feel free to load *wwwstat.pl* from the Contrib directory (see “The Contrib Directory” on page 44). This program is a bit more powerful than getstats and has a Web interface.

There are many other commercial stat programs out there that are also more inclusive than the ones we provide. You are certainly welcome to download any of them and install them on your server.

---

## 2.8 Resetting Your Virtual Web Server Log Files

---

Your log files can grow quite large and quite fast, especially if your site is getting a large number of hits per day. The size of these files can eat into your disk quota and tie up needed space.

To reset your log files, use the *vnukelog* command (see “The *vnukelog* Command” on page 43).



# The Virtual FTP Service

---

## 3.1 Introduction to FTP

---

Connecting to a remote computer using FTP (or File Transfer Protocol) is similar to using *TELNET* or *rlogin*, except that you do not have all the tools of a shell, and your access to files is limited. You use FTP to transfer files. The files can be of any type. For example, they may be text files or binaries in any format -- it does not need to be HTML or in some graphic format, for example. Also, you can transfer files between different types of computers. You might, for example, transfer files between an UNIX server and a PC (with a FTP client). Part of the reason FTP is so popular is that FTP clients are so easy to acquire for every platform.

FTP is great for transferring files but terrible for browsing. If you have worked with a file structure of any size, you know how difficult it can be to navigate through it. Although file names can be descriptive they're not descriptive enough. Some FTP administrators even put README or INDEX files in every directory or use automated messages (see "Log In Banners and Directory Messages" on page 27) to give the user some clue as to what is in the directory. FTP archives, however, are easier to set up than WWW sources, since you do not need to translate your documents into HTML.

### 3.2 Anonymous FTP vs. FTP

---

The virtual system supports *Anonymous FTP* or FTP that anyone can access without a password, and *Non-Anonymous FTP*, that requires a username and password to gain access. With anonymous FTP you just enter *anonymous* or *ftp* for the username and usually your Email address as the password.

### 3.3 The Name of the Virtual FTP Service

---

There is a de facto, but not required, standard for naming FTP and other services. They are the names users will try first and are the easiest to remember. The format for the FTP service is:

ftp.domain.type

Where *domain* represents the domain name of your FTP server (often your company's name) and *type* represents the type of organization or the *top-level domain name* (i.e. *edu* is used for educational institutions, *com* is used for businesses, *net* is used for network services providers, *org* is used for non-profit organizations, and *gov* is used for government entities).

By default, if your domain name is registered by us, then your virtual anonymous FTP services will be in the standard form above. Naming your FTP service abc.domain.type or zzz.domain.type is not illegal but it is not a good idea.

Other services follow a similar naming convention. For example, World Wide Web servers are named *www.domain.type* and Post Office Protocol (POP) servers are named *pop.domain.type*. Again, the virtual server system will use these names by default for the respective services.

### 3.4 Your Anonymous FTP Directory

---

Using anonymous FTP is the safest way to grant access to the virtual FTP service because it is restricted to the *ftp* directory in your home directory. With this restrictive access and by assigning permissions correctly you can limit the harm they can do.

Your *ftp* directory in your home directory, by default, contains one sub-directory: *pub*. *pub* traditionally contains the archive files available to anonymous FTP customers. This is the directory where you should put the files you want to make available to your customers. You can make other directories as needed.

### 3.5 Making an "incoming" Directory for Customers

---

In some cases it would be desirable to allow your users to *upload* files to your virtual anonymous FTP server. Separating these files into their own directory is a form of damage control. If someone uploads a virus hopefully its damage is confined to the incom-

ing directory. If you see no reason to permit uploads then there is no reason to create one, so simply skip the rest of this section.

It is recommended that the incoming directory be given only write permissions. With *write-only* permission, it is somewhat inconvenient because it prevents customers from looking at files others have uploaded to the server. On the other hand, it prevents other users from perverting or deleting those same files. A by-product of allowing users to read others' uploaded files (having the read permission set) is that they can upload completely unrelated files on your virtual server. These uploaded files can be bland or blasphemous. For the sake of your company's image, you would not want to inadvertently support a porn or *WaReZ* (pirated software) archive on the Internet.

To make an *incoming* directory do the following:

1. Inside your *ftp/pub* directory create a directory with a name like "incoming" (`mkdir ftp/pub/incoming`).
2. Create a file named *.incoming* (don't forget the ".") in the *ftp/pub/incoming* directory (`touch ftp/pub/incoming/.incoming`). The *.incoming* file flags the directory as a write-only directory. Files in this directory can not be read or listed.

---

### 3.6 Log In Banners and Directory Messages

---

Some FTP servers have messages that are displayed just after you log in or change into a new directory. These messages give the user helpful information about your FTP site or the directory they just changed in to. For example, you may want to give a short description of the files in a directory or information about the server they just logged in to.

To create a message that is displayed to the FTP user just after they log in, create a file named *.welcome* in your *~/ftp/pub* directory:

```
salmon: {8} % cat > ~/ftp/pub/.welcome
Welcome to ACME Rockets Inc Anonymous FTP Server!

Please send any questions, comments, or problem reports about
this server to ftp@acme-rockets.com.
^D (Control-D)
```

To create a message that is displayed to the FTP user when they change into a directory, create a file named *.message* in the directory that you want the message to appear. For example, if you offer a demo version of software that your company sells you could create a directory called *demo* with a file named *.message* in that directory:

```
salmon: {10} % mkdir ~/ftp/pub/demo
salmon: {11} % cat > ~/ftp/pub/demo/.message
This directory contains demo versions of ACME Rocket's products:

missile.zip - Missile CAD(tm) Version 1.0 (DEMO)
nuke.zip - Thermo Nuclear War Simulator(tm) Version 2.1 (DEMO)
^D (Control D)
```

### 3.7 Creating Non-Anonymous FTP Accounts

---

If your virtual server is configured to handle non-anonymous FTP accounts, it is quite simple to add FTP accounts for your users. This allows you to create accounts that can be used to upload or download web content, files into the anonymous FTP file area, or in private FTP upload/download area. The most important thing for you to decide is what type of access you want your users to have.

The most common use for non-anonymous FTP on the virtual servers is for companies who want to resell some of their space to their clients and allow them to upload and maintain their own home pages. Another use for non-anonymous FTP is for companies that have valuable information that they want to make available via FTP for only those that have been given a specific password.

Adding Non-Anonymous FTP accounts uses the same procedure as adding POP Mail accounts mentioned in Section 5.2 on page 36. Creating this FTP account will also create an Email POP account. If you do not wish the user to access Email on your server, simply do not tell them about the Email account.

To add the Non-Anonymous FTP accounts, log into the server using the virtual administrator login name and password and use the *vadduser* command (or the *iroot* program as described in Section 7.2 on page 51). For example:

```
salmon: {2} % vadduser
```

Email User Names are up to 8 characters and consist of upper or lower case alphabetic characters or digits. They must start with an alphabetic character and should generally be all lower case.

Email User Name: **biff**

Now enter a password for this user's POP mail account. For security reasons you may want to use a password that is longer than 6 characters and that has at least one non alphabetic character. The password will \*not\* be echoed to the screen and you will be required to type it twice.

POP password: <**biff's password**>

Retype POP password: <**biff's password again**>

Now enter the Email User's full name. Please use less than 80 characters and no ':' characters.

Full Name: **Bifford McLean**

Do you want this user to have FTP access to your virtual server (assuming that you have the FTP service with your account)? Please answer yes or no. If you are planning to add the FTP service to your account later and want this user to have FTP access you can answer "yes" now.

FTP, yes or no: **yes**

You have three choices on where to put this user's "home directory":

- (1) /usr/home/biff
- (2) /usr/local/etc/httpd/htdocs/biff
- (3) /ftp/pub/biff

Pick 2 if you want this user to be able to upload his or her own web pages. (The URL would be something like <http://www.yourcompany.com/biff>) Pick 3 if you want this user to be able to upload files to your anonymous FTP archive (<ftp://ftp.yourcompany.com/biff>).

Otherwise, pick 1.

Home directory option, 1, 2, or 3: **2**

Enter the FTP upload quota on this account in megabytes (0 for no quota): **10**

FTP/Email User added successfully.

After entering the FTP user's name, the password, and the full name for the account, the FTP option, and the FTP upload quota, it will be added. This will setup the electronic mailbox for the user and the FTP account selected.

Let's take a look at what the *home directory* possibilities are.

The first option allows you to create the home directory under your */usr/home* directory. If the directory were called *test*, it would be created at */usr/home/test*. This would be an ideal place for you to create an FTP directory for users to upload information to your server that your system administrator would verify and place in the proper directory structure.

The second option allows you to create the home directory under your */usr/local/etc/httpd/htdocs* directory. If the directory were called *test*, it would be created at */usr/local/etc/httpd/htdocs/test*. This is ideal if you wanted to allow user *test* to update his own home pages. He would have FTP access to the *test* directory, and anything below that he created. However, he would not be able to access anything above the *test* directory. His home pages would be found at <http://www.yourcompany.com/test>.

The third option would be used if you wanted your user to be able to upload files to your anonymous FTP archive. The directory created for the user *test* would be */ftp/pub/test*. Files in this directory could only be added and deleted by the user *test*, but anyone would have access to download these files.

The FTP upload quota allows you to limit how much of your virtual server's disk space one of your users may use. If they try to upload more than what their remaining quota allows, they will receive a FTP error message.

### 3.8 Monitoring Anonymous FTP Activity

The *messages* file located in your *usr/logs* directory contains valuable information describing how much your virtual anonymous FTP server has been used. This information, however, may not be in a very readable form. A program by the name *xferstats* can be used to make a summary of your anonymous FTP activity as shown in Figure 3.

*xferstats* may be run periodically by the *cron* facility. This can be done as follows:

1. Create a file named *cfile* with the following information:
 

```
# Crontab file (see crontab(5))
# Every Sunday morning at 2:13am process FTP xferstats and "nuke" message file
13 2 * * sun /usr/local/bin/xferstats -m user@xyz.com -n
```
2. Run *crontab* to install the cron file (*cfile*) you just created:
 

```
salmon: {1} % crontab cfile
```

For more information on *cron*, type *man crontab* and *man 5 crontab* at the virtual server's UNIX prompt.

FIGURE 3

Example Output from *xferstats*

```
TOTALS FOR SUMMARY PERIOD Aug 16 TO Aug 17

Files Transmitted During Summary Period      3
Bytes Transmitted During Summary Period     762
Systems Using Archives                       0

Average Files Transmitted Daily              2
Average Bytes Transmitted Daily             381

Daily Transmission Statistics

      Date      Number Of  Number of  Average  Percent Of  Percent Of
      -----      Files Sent  Bytes Sent  Xmit Rate  Files Sent  Bytes Sent
-----
Aug 16                2          508  508.0 KB/s   66.67     66.67
Aug 17                1          254   0.3 KB/s    33.33     33.33

Total Transfers from each Archive Section (By bytes)

      Archive Section  Files Sent  Bytes Sent  ---- Percent Of ----
      -----
/pub                    3          762   100.00    100.00

Hourly Transmission Statistics

      Time      Number Of  Number of  Average  Percent Of  Percent Of
      -----      Files Sent  Bytes Sent  Xmit Rate  Files Sent  Bytes Sent
-----
03                    1          254   0.3 KB/s    33.33     33.33
05                    2          508  508.0 KB/s   66.67     66.67
```

# The Virtual Email Service

---

## 4.1 Introduction to Email

---

As a common feature on local networks, you may already be familiar with Electronic Mail or Email. You can send and receive letters across local network or Internet lines when available. You can even broadcast messages to many people at once. The Internet has mailing lists that you can retrieve from or broadcast to. You might, for example, broadcast a message to an entire newsgroup or discussion group. Of course, you want to broadcast messages sparingly. Flooding newsgroups with Email about your company every week is not proper Internet etiquette and is strictly prohibited. Doing so will provoke the ire of Internet users.

The advantage of using Email as a way to communicate over the Internet is its immediacy. Most people on workstations have a mail client running. Rather than waiting for people to fire up their browsers and search across the Internet to find your company's home page, Email can arrive at their workstation immediately. Again, it is not proper Internet etiquette to broadcast Email to a wide body of users in order to promote your company. Instead, you could use Email to foster your company's relationships with its established clients.

## 4.2 A Tour of the Virtual Email Handling System

---

The Virtual Email System is directly based on sendmail 8.6.12 (don't worry if this does not mean anything to you). Your virtual sendmail uses the following configuration files and directories:

- **etc/aliases** This file contains aliases for mail addresses. The next section will discuss how this file is used.
- **etc/aliases.db** This is a binary version of the file above and is created by running the *vnewaliases* command.
- **etc/sendmail.cf** This is the master configuration file for sendmail. Unless you are a sendmail guru you may want to stay away from this file since it is fairly cryptic.
- **usr/mail** This directory is the home for Email users' mailboxes. POP also uses this directory for temporary files.
- **usr/spool/mqueue** This directory is used by sendmail to temporary queue messages going out. Most of the time it should be empty.

## 4.3 Creating Email Aliases

---

Email aliases allow you to forward electronic mail sent to a mail user name on your virtual server another electronic mail address. For example, you could make all the mail sent to *webmaster@yourdomain.com* sent to *you@xyz.com*. To do this do the following:

1. Edit the *etc/aliases* file and add the following at the end:  
**webmaster: you@xyz.com**  
You can use one of the many editors available on the server including *pico*, *vi*, or *emacs*.
2. Run the program *vnewaliases* to update the *etc/aliases.db* file by typing *vnewaliases* at a telnet command prompt:  
Salmon: {4}% **vnewaliases**
3. You may also FTP the aliases file to your local machine to edit if you wish. Once you edit the file, FTP it back to your server in the *etc* directory and type *vnewaliases* at a telnet command prompt. *Be aware that you must FTP this file in ASCII format and not Binary format because it is a text file.*

You do not need to create a mail account as described in section 5.2 on page 36 in order to have an Email alias. After an alias is created the mail is not stored on the virtual server and is simply forwarded.

## 4.4 Creating Email Lists

---

It is sometimes convenient to create a single Email address that is an alias for many Email addresses (an electronic mailing list). There are two ways to do this. First, you can add multiple Email addresses separated by a comma to a single line in the *etc/aliases* file as described above. For example:

```
myfriends: joe@xyz.com, fred@foobar.edu, mary@abc.org
```

Of course, this only works for small lists. For larger lists you can use *:include:* and a separate file for the aliases. To do this for your mailing list do the following:

1. Add the following to the bottom of the *etc/aliases* file add the following:  
**myfriends:        :include:/etc/mylist**
2. Create the file *etc/mylist* with each Email address on a separate line. For example:  
**joe@xyz.com**  
**fred@foobar.edu**  
**mary@abc.org**  
.
3. You will need to change the permissions of the list file, and the entire *etc* directory. While in your home directory in a telnet session type:  
**chmod 755 etc/mylist**  
**chmod 755 etc**
4. Run *vnewaliases* to update your *etc/aliases.db* file.

### 4.5 Creating Email Autoresponders

---

You may want to setup Email addresses that your clients can mail to that will automatically send them a reply with information on a particular subject. You may also want to have an Email address that sends a reply to the sender informing them that their message has been received.

These are various types of autoresponders that can be setup on your virtual server system quite easily. Simply follow these steps in a telnet session:

1. Copy the “autoreply” program from “/usr/local/contrib” to your “usr/bin” directory. You must type the following lines in a telnet session:  
**cp /usr/local/contrib/autoreply ~/usr/bin/autoreply**
2. Make sure it is executable:  
**chmod 755 ~/usr/bin/autoreply**
3. Create an autoreply message (the message sent back to the customer) in your home directory:  
**pico ~/.autoreply (pico is an easy to use editor)**
4. Add something like the following to your *etc/aliases* file:  
**info: name@yourcompany.com, “/usr/bin/autoreply -f info-reply -a info”**
5. Run *vnewaliases* to update your *etc/aliases.db* file by typing *vnewaliases*.

With the above example, when your customer sends you mail at “info@yourcompany.com” it will be received at name@yourcompany.com and send back whatever you have in the *~/.autoreply* file. By the way, you can use the “-m” option to specify a different message file (i.e. “autoreply -m /etc/mymessage” would send the file */etc/mymessage* instead of the default *~/.autoreply*). You will need to use this option when setting up more than one autoreply message.

The “-f” allows you to change who the autoreply message will be from. In the example above the “From:” field the customer gets will read *info-reply@yourcompany.com*. This

way your customer can reply to your message without getting the autoresponder again. You will also need to create an alias for *info-reply*.

the “-a” specifies who autoreply can reply for; this should be the same as what is to the left of the “:” in the *etc/aliases* file.

Note that in the example above mail sent to *info@yourcompany.com* will also be sent to *name@yourcompany.com*. Without the *name@yourcompany.com*, the mail from the customer would not be seen by anyone. Of course, you can leave this off if the Email address is used only for sending an automated response to the customer that requires no other interaction.

---

### 4.6 Error Messages

---

If something is not working the way you think it should be then take a look at the *usr/log/messages* file for error messages. The virtual Email server will log messages to this file. Also, all mail that comes in or goes out is logged in this file. This is a good way to keep track of mail usage on your virtual server.

This file can grow to quite a large size over time. It is recommended that you delete this file when this happens to free up your disk space. You can reset this log file and all your web access log files by issuing the *vnukelog* command (see “The *vnukelog* Command” on page 43.) To run this, type *vnukelog* at a telnet command prompt.

## The Virtual POP Service

---

### 5.1 Introduction to POP

---

Post Office Protocol or POP allows users to read their Internet Email without having to log into a server and learn a cumbersome Email program and/or operating system. Instead the user continues to use the operating system he or she is familiar with and a compatible POP client. Now a company or individual can choose from many high quality, low cost, POP clients for nearly every major operating system, including Windows, MacOS and OS/2.

The Virtual POP Service allows a company, new to Email systems, to inexpensively establish an Internet Email box for each employee which is activated by their own individual password. This is because with the virtual server a company does not need to purchase an expensive commercial gateway or pay for the excessive dedicated Internet connection. Commercial SMTP (the Internet Email protocol) gateways for Novell or Microvolts Email systems can cost thousands of dollars to install.

With the Virtual POP server ISP costs can be lowered significantly by having the employees access their Email with a shared dial-up Internet connection. For example, a small company could purchase a single dial-up account for the company. Rather than sharing the ISP's single POP Email box included with the account the company could configure each individual employee's computer to use an unique mailbox off the company's virtual POP server giving each member of the company their own individual

Internet mail address. Each employee would have their own unique Email account username and password but would share the dial-up or ISP account username and password. The Email account username and password is configured in the POP client software as shown in Section 5.6 on page 38.

---

## 5.2 Creating POP Accounts for Email Users

---

To create POP accounts for Email users, log into the server using the virtual administrator login name and password and use the *vadduser* command. This will also add a non-anonymous FTP account for your user if you wish (see “Creating Non-Anonymous FTP Accounts” on page 28). For example:

```
salmon: {2} % vadduser
```

Email User Names are up to 8 characters and consist of upper or lower case alphabetic characters or digits. They must start with an alphabetic character and should generally be all lower case.

Email User Name: **biff**

Now enter a password for this user’s POP mail account. For security reasons you may want to use a password that is longer than 6 characters and that has at least one non alphabetic character. The password will \*not\* be echoed to the screen and you will be required to type it twice.

POP password: <**biff’s password**>

Retype POP password: <**biff’s password again**>

Now enter the Email User’s full name. Please use less than 80 characters and no ‘:’ characters.

Full Name: **Bifford McLean**

Do you want this user to have FTP access to your virtual server (assuming that you have the FTP service with your account)? Please answer yes or no. if you are planning to add the FTP service to your account later and want this user to have FTP access you can answer “yes” now.

FTP, yes or no: **yes**

You have three choices on where to put this user’s “home directory”:

- (1) /usr/home/biff
- (2) /usr/local/etc/httpd/htdocs/biff
- (3) /ftp/pub/biff

Pick 2 if you want this user to be able to upload his or her own web pages. (The URL would be something like <http://www.yourcompany.com/biff>) Pick 3 if you want this

---

## Changing a Mailbox Password

---

user to be able to upload files to your anonymous FTP archive (ftp://ftp.yourcompany.com/biff).

Otherwise, pick 1.

Home directory option, 1, 2, or 3: **2**

Email User added successfully.

After entering the Email user's name, the password, and the full name for the account, and the FTP option, it will be added. This will setup the electronic mailbox for the user and the FTP account if selected.

---

### 5.3 Changing a Mailbox Password

---

From time to time users will forget their POP account password. You can not recover this password but you can reset the password to something else with the *vpasswd* command. For example:

```
salmon: {3} % vpasswd biff
New password: <Biff's new password>
Retype new password: <Biff's new password again>
```

In this example, the Email user biff's password is changed.

---

### 5.4 Removing a POP Email Account

---

To remove an unwanted Email user account use the *vruser* command. For example:

```
salmon: {4} % vruser
Please enter the Email User Name to be *removed*.
Email User Name: biff
Are you sure you want to remove "biff"'s account (y/n)? y
```

```
Password entry removed...
Mail file removed...
POP file removed...
Email User account removed successfully.
```

In this example, *biff's* POP Email account and FTP account are removed. However, if you added an FTP account for the user, the directory will not be deleted. You will need to do this manually.

---

### 5.5 Listing POP Email Account Users

---

To view a list of your current Email/FTP account users use the *vlistuser* command. At a telnet command prompt type:

Salmon: {1} % vlistuser

---

## 5.6 Configuring the POP Client Software

---

In general, the POP client needs to be configured to get its mail from the virtual server and to use the virtual server as a *SMTP relay* host. This section shows how to configure Eudora, one of the most popular POP client software packages for the PC. Please note that Eudora is also available for the Macintosh and is similarly configured.

### 5.6.1 Configuring Eudora

Qualcomm's Eudora™ for both the Macintosh and PC is available from *ftp://ftp.qualcomm.com/Eudora*. Eudora comes in two forms: freeware and a commercial version. The commercial version has many enhancements but costs a little money (for more information send mail to *eudora-sales@qualcomm.com*).

Eudora for Windows is shown in Figure 3 on page 39. To configure Eudora to use the virtual server *petstore.com* for the Email user *Biff* I would do the following:

1. Start up Eudora by double clicking on the Eudora icon.
2. Select **Options** in the **Tools** pull-down menu (see Figure 3 on page 39).
3. In the **Options** dialog box, click on the **Personal Information** icon to change the following values (see Figure 4 on page 39):  
**POP Account:** biff@pop.petstore.com  
**Real Name:** Bifford McLean  
**Return Address:** biff@petstore.com
4. In the **Options** dialog box, click the **Hosts** icon to change the following value:  
**SMTP Server:** smtp.petstore.com
5. Configure the other parameters to the user's preference and click on the OK button.

Once this is done Eudora is ready for the Email user. When Eudora is invoked it will prompt the user for his or her Email password ("Creating POP Accounts for Email Users" on page 36).

### 5.6.2 Configuring Microsoft's Internet Exchange™ and Netscape's Mail™

Microsoft's Internet Exchange and Netscape's Mail client are both able to be configured to receive Email from your virtual POP server. The settings will be similar to those listed in "Configuring Eudora" on page 38. For more specific information on these programs, see their respective help files contained with the programs.

FIGURE 3 Eudora for Windows

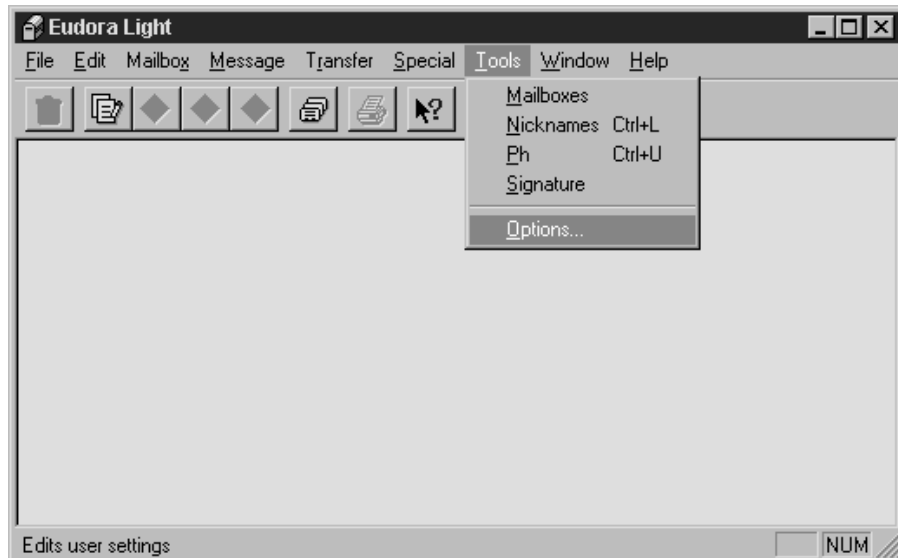
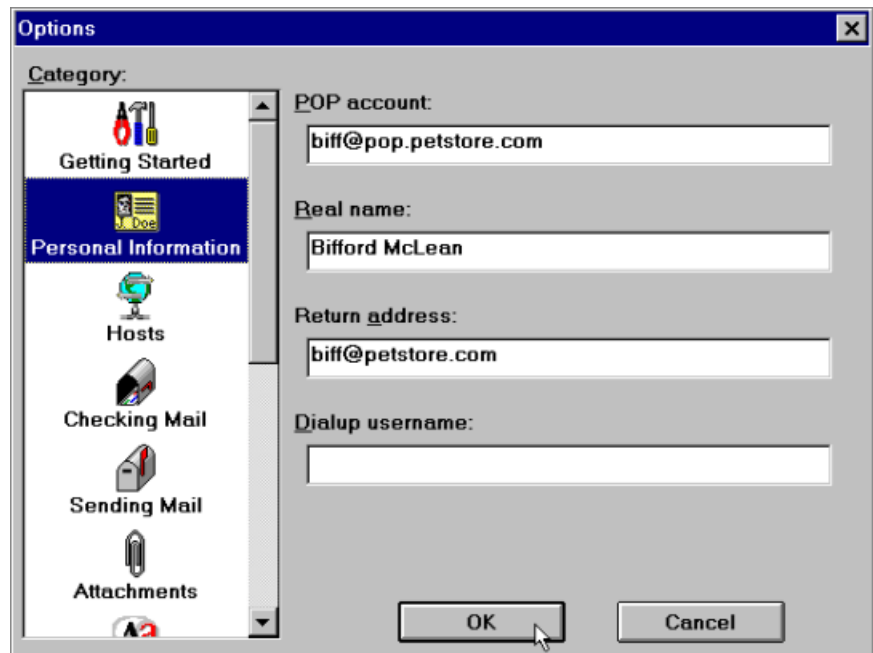


FIGURE 4 Eudora's Options Dialog Box





CHAPTER 6

# Helpful Commands and Useful Information

---

## 6.1 Overview

---

The virtual server system is a powerful system that allows you to do many things that you simply cannot with a virtual “host.” This chapter is designed to introduce you to some commands and information that will help you make the most out of your virtual server.

---

## 6.2 The quota Command

---

The *quota* command is used to help you identify how much disk space you have on your virtual server, and of that space, how much you have already used. At a telnet command prompt, simply type *quota*.

```
Salmon: {1} % quota
```

The command will output your *blocks*, *limit*, *files*, and *file limit*.

1. To see what your quota is, look at the **limit column**. A block is 1024 bytes. If for example, your limit says 51200, you would divide that number by 1024 to get your disk space in Megabytes. In this example, your total space would be 50 Meg.
2. To see how much space you have used, look at the **blocks column**. Divide this number by 1024 to see how much space you have used. If for example, your blocks say 37851, you would be using roughly 37 Meg of your space.
3. The **files and file limit columns** that display tell you the total number of files you can have on your system.

---

### 6.3 The `vdiskuse` Command

---

The `vdiskuse` command is useful in showing you where most of your disk space utilization is going. The command will return a listing of your directory structure along with the amount of disk space used in each directory.

To use the `vdiskuse` command, simply telnet to your server and type the following:

```
Salmon {1} % vdiskuse
```

If the output is too long to fit onto one screen, you can add the `more` command. The `more` command will show just one screen of information. When you hit the space bar the next screen will scroll into view.

```
Salmon {2} % vdiskuse | more
```

You can also have this report Emailed to you if you don't want to read it off your screen. Type this at the telnet command line and replace `yourname@domain.com` with your Email address:

```
Salmon {3}% vdiskuse | mail yourname@domain.com
```

#### 6.3.1 Dead Processes Taking Up Disk Space

Sometimes you will have files that are counted against your quota that you cannot see with the `vdiskuse` command. For example you will run `quota` and it will show you as using 25 Meg but `vdiskuse` will show you as using only 10 Meg.

This happens when you try to delete files that are opened for some reason. The most common cause of this is trying to delete your log files when they are actually being written to by the server. That is why it is so important to use the `vnukelog` command.

However, if this happens type the following at a telnet command prompt replacing `<username>` with your login name:

```
Salmon {4}% ps -aux | grep <username>
```

It will return a list of processes that might have one that looks similar to this:

```
<username> 4769 0.0 0.3 1436 196 ?? IW 16Oct96 0:00.06 httpd
```

Look for processes that are over a day old such as this one and kill them by process id. The process id is the first number listed after your username. In the above example the process id is 4769. This will free up the space the locked files were taking:

```
Salmon {5}% kill <process id>
```

---

## 6.4 The vnukelog Command

---

Your access log files will likely be your largest user of disk space. It is wise to reset or *nuke* your log files when they get large. (For more information on your log files see “Monitoring Your Virtual Web Server Log Files” on page 18) You can do this by simply typing *vnukelog* at a telnet command prompt.

```
Salmon: {1} % vnukelog
Successfully deleted messages_log.
Successfully deleted access_log.
Successfully deleted error_log.
Successfully deleted agent_log.
Successfully deleted referer_log.
```

---

## 6.5 The traceroute Command

---

When users have trouble reaching their virtual server, or their connection time is slow, the traceroute command can be used to find out where the trouble is. This command can be issued from the users local machine running Microsoft’s Windows 95, or from the virtual server itself if you are able to telnet to the server.

The traceroute command follows the data route from the machine where you begin the route to the machine you route to. The route returns many different bits of information, but the most important thing to look for are asterisks ( \* ). These indicate packet loss along the route. Packet loss along the route translates to slow, or completely inaccessible service.

If the packet loss occurs from your ISP computer systems, you can call them and ask why things are so slow. If it occurs somewhere in between your ISP and your virtual server, you will need to wait it out or try another ISP that might have an alternate route. If the packet loss occurs within the virtual servers computers, send Email to us to see when it will be cleared up.

To run the command from your virtual server, simply telnet in and type */usr/sbin/traceroute www.domain.com* (substituting your ISP’s domain name for *www.domain.com*). For example, if you were having trouble connecting from your service provider whose domain name was *acmenet.com*, you would type the following on your virtual server:

```
Salmon: {1} % /usr/sbin/traceroute www.acmenet.com
```

If you were having trouble even logging in to your virtual server, you can run a traceroute from your Windows 95 machine. At a DOS prompt you type *tracert www.domain.com*. You would substitute *www.domain.com* with the domain name of

your virtual server. For example, if your virtual server domain name was acmenet.com, you would type the following:

```
C:\Windows>tracert www.acmenet.com
```

---

## 6.6 The Contrib Directory

---

As a virtual server reseller/client, you have access to a large array of programs that can really enhance the functionality of your virtual server. These programs are all located in the "contributed" area, or **/usr/local/contrib**. This /usr/local/contrib is not a directory within your directory structure, but a directory of the host machine root from which you are able to copy programs.

The programs and utilities that are located in the /usr/local/contrib are not directly supported by our technical support since they were largely developed by third parties. Many of the contributed programs come with a large amount of documentation that should be reviewed and understood. Furthermore, many of the programs have license agreements that you should make yourself familiar with and agree to.

For the most current listing of programs available for you in the Contrib directory Email us. The following is a list of programs and their functionality at the time of this printing:

### 1. **Formmail.pl**

This script is an easy way to handle form content and have it mailed to some specified address. The form can also be modified to send an automated response back to the remote client.

### 2. **Majordomo**

Majordomo is a perl program written to handle routine administration of Internet mailing lists.

### 3. **PGP (Pretty Good Privacy)**

PGP(tm) uses public-key encryption to protect E-mail and data files. PGP is well featured and fast, with sophisticated key management, digital signatures, data compression, and good ergonomic design.

### 4. **PGPformmail.pl**

This script is an easy way to handle form content and have it mailed to some specified address encrypted with your PGP(tm) public key. The form can also be modified to send an automated response back to the remote client.

### 5. **Whois Gateway**

A simple CGI to perform lookups on domain names.

### 6. **WWW Board 2.0**

WWW Board is a web bulletin board message system. The script provides some administrative utilities to manage the bulletin board.

### 7. **WWW Count 2.2**

A CGI program to keep record of the raw hits of a web page. It generates a GIF image of the number of hits and returns to the browser as an in-lined image.

### 8. WWW Stat 1.0

An access statistics package with a web interface.

You will also find links to our free CGI library and Java Applet examples at this site.

---

## 6.7 Creating Symbolic Links

---

You might find it necessary at one time or another to create symbolic links between files, or even directories. On the virtual server system, we have created a symbolic link for the **usr/local/etc/httpd** directory. The link is simply **www**. From your home directory, you can type `cd www` and it will place you in the same directory as if you had typed `cd usr/local/etc/httpd`.

Links can be for your convenience in navigating around your site, or can be used to create a series of home pages that actually point to a different page. One of the most used links on the virtual server system is the link to the `index.html`.

The default home page that comes up for each directory of your web server is the file named `index.html`. If you do not have an `index.html` file, then a directory listing will come up in the users web browser. However, some people do not want to call their default page `index.html` for various reasons. There are two ways around this. First, you can change the **DirectoryIndex** parameter in your `srm.conf` file (see Section 2.2.2 on page 10), or you can create a simple symbolic link.

To create the link, telnet to your server and go to the directory where your files are located. Assuming you have a file called `welcome.htm` and you want it to be the default home page for that directory, type this at the command prompt:

```
Salmon: {1} % ln -s welcome.htm index.html
```

This will establish a link that points `index.html` to `welcome.htm`.



# The iManager Server Extension and iRoot Plug-in

We have developed the iManager server extension and iRoot plug-in. These programs provide users with a web interface to many common administrative and maintenance tasks associated with a virtual server. These extensions enable the user to maintain his or her server from a web interface without ever needing to log on to the virtual server in a telnet or FTP session. A user can now do many tasks easily and efficiently from their browser of choice.

These programs are available for users to load onto their server from the Server Extensions directory (contact us for exact location). These extensions can be used in conjunction with remote web authoring tools such as FrontPage from Microsoft™. With these tools a secretary can be the webmaster for her company.

The rest of this chapter will introduce you to the simplicity and functionality of the iManager extension and iRoot plug-in.

---

## 7.1 The iManager Extension

---

The iManager extension will enable you to do the most common tasks associated with maintaining the files on your server with a nice web interface. This extension will reduce your need to telnet to your server to change file properties. The iManager will execute many commands for you that are commonly used so you can keep your Unix knowledge to a minimum. These tasks include:

- Editing a File
- Deleting a File
- Copying a File
- Moving a File
- Linking a File
- Changing the Permissions of a File
- Uploading a New File to your Server
- Making a New Directory

### 7.1.1 Installing the iManager Extension

To install iManager, telnet to your Virtual Server and perform the following steps:

1. Telnet to your server and make your home directory the current working directory. To do this, type "cd" at your telnet prompt and then hit the return key.
2. Untar the iManager archive file, type the following:  

```
tar -xvf /usr/local/contrib/imanager.tar
```
3. Customize the look and feel of iManager by modifying the files header and footer which contain the open body tag, <body>, and close body tag, </body> respectively. (Hint: use iManager to edit these files- see instructions below).

You can now use the iManager utility on your server by accessing the URL:

```
http://your_domain_name/cgi-bin/admin/gateway.cgi
```

You can either open the URL with your browser directly, or link the URL to either text or an image on one (or more) of your web pages.

### 7.1.2 Running the iManager Extension

The iManager extension is also run from your web browser. Enter the following URL into your web browser substituting <domainname> with your domain name:

```
http://www.<domainname>.com/cgi-bin/admin/vs/imanager.pl
```

This page will prompt you for your root password. Simply enter your virtual server root password and click on the *Start iManager* button.

When iManager is started, you will see a screen that shows you your current working directory on your server followed by a list of entries in that directory.

You can move around the directories by modifying the **Path Specification** box and clicking on the **Change Directory** button. If you want to change to a directory below your current working directory, you can also simply click on the directory name that is in the list below the **Change Directory** button.

If your list of entries within your current working directory contains a graphic file or a home page file, you can click on the name of that file to bring up a small browser window that will show you the graphic image or the home page file.

You will also notice that the list of entries shows you not only the name, but also the file size, the date the file was last modified, the mode or permissions of the file, and a series of options to edit, delete, copy, move, link, or chmod (change permissions) for that file. These options will be discussed in more detail below.

### 7.1.3 Editing a File with iManager

The iManager extension allows you to edit the text files, such as html files, from within your web browser. This is useful if you need to make a quick change and do not want to telnet to your server to do it.

When you start iManager and look at the list of entries you will notice the column titled **Edit**. The entries will either have **ED**, **CD**, or a blank in this column. If the entry has **ED**, then it is an editable text file. Click on the **ED** to begin editing.

If the entry has the **CD** option in the **Edit** column, then you know the entry is a directory. By clicking on the **CD** you will change to that directory as your current working directory and a new entry list will appear.

If the entry has no option in the **Edit** column, then you know that the file is neither a directory nor an editable file. Graphic images will appear like this.

### 7.1.4 Deleting a File with iManager

The iManager extension also allows you to delete files and directories from your server without actually telneting to your server to do it.

When you start iManager and look at the list of entries you will notice the column titled **Delete**. All the file and directory entries below will have the **RM** listed in this column. The **RM** stands for the rm (remove) unix command. Clicking on the **RM** will take you to a screen to confirm that you want to delete the selected file or directory.

### 7.1.5 Copying a File with iManager

The iManager extension can copy files on your server to a new file and a new location.

When you start iManager you will see the column titled **Copy**. Click on the **CP** option for any of your files or directories and you will be taken to the **Copy File** or **Copy Directory** page. On this page you will be prompted to enter the name of the new copy you are creating for the file or directory you selected. You are also able to change the

path for your copy if you would like to place it in a directory other than your current working directory.

#### 7.1.6 Moving a File with iManager

Moving a file in iManager is also quite simple. When iManager is running you will see the column titled **Move**. Click on the **MV** option for any of your entries and you will be taken to the **Move (Rename) File** or **Move (Rename) Directory** page.

This page is very similar to the **Copy File** or **Copy Directory** page. However, this page uses the **mv** Unix command which actually moves the file or directory to a new location or renames the file or directory.

#### 7.1.7 Linking a File with iManager

The iManager also allows you to link files with symbolic links. Symbolic links enable you to have one file actually point to another file. For example, if you have a few files with different names that are all actually the same file, you can link them all to one file. This way if you need to change this file you can change it once and it will affect all the linked files.

For a more extensive understanding of symbolic links see “Creating Symbolic Links” on page 45.

To link a file with iManager, click on the **LN** under the **Link** column. This will take you to the **Create Link to File** or **Create Link to Directory** page. At this point you will be prompted to enter the name and path of the file you want to link to the file you selected.

If you have a file called `index.html` and want to link a new file called `welcome.html` to it, select the **LN** next to your `index.html` file. On the **Create Link to File** page enter the Filename as `welcome.html`. Now when someone accesses `welcome.html` from the web it will actually give them the `index.html` file. If you edit your `index.html` file, the `welcome.html` file will show the changes.

#### 7.1.8 Changing the Permissions of a File with iManager

iManager allows users to change the permissions of files from their browser. To change the permission of a file click on the **CM** next to the filename. If you are unsure about what file permissions you need for a file or directory then leave them alone.

#### 7.1.9 Uploading a New File to your Server with iManager

You can use iManager to upload a file for you from your local computer to your virtual server without the need of an FTP client. From the iManager main screen go to the **Upload File to Current Working Directory** section.

In this section enter the **Local Filename** of the file located on your computer that you want to upload to your virtual server. You can use the **Browse** button if you need to look around your computer to find the proper file.

Then enter the **Remote Filename** or name that you will want to call the file once it is loaded onto your server. You can also change the **Remote Path** if want to upload the file to a directory other than your current working directory.

When you have selected the correct file and correct remote filename and path, click on the **Upload File** button.

### 7.1.10 Making a New Directory with iManager

Within iManager you are able to add a new directory to your virtual server under your current working directory. Simply go to the **Make New Directory** section of iManager and enter the new directory name. Click on the **Make New Directory** button and it will be created.

---

## 7.2 The iRoot Plug-in

---

The iRoot plug-in will enable you to do the most common tasks associated with maintaining your server. These tasks include:

- Adding Email and FTP users
- Changing Email and FTP users' passwords
- Removing Email and FTP users
- Adding, Deleting, and Updating your Email aliases
- Changing your server Root password
- Creating a Digital Certificate for Allowing SSL (Netscape™ compatible) Encryption on your server
- Installing a Digital Certificate on your server

### 7.2.1 Installing the iRoot Plug-in

To install iRoot, telnet to your Virtual Server and perform the following steps:

1. Make your home directory the current working directory, type "cd" at your telnet prompt and then hit the return key.
2. Untar the iRoot archive file, type the following:

```
tar -xvf /usr/local/contrib/iroot.tar
```

You can now use the iRoot utility on your server by accessing the URL:

**[http://your\\_domain\\_name/cgi-bin/admin/gateway.cgi](http://your_domain_name/cgi-bin/admin/gateway.cgi)** or

**[http://your\\_domain\\_name/cgi-bin/admin/vs/gateway.cgi](http://your_domain_name/cgi-bin/admin/vs/gateway.cgi)**

You can either open the URL with your browser directly, or link the URL to either text or an image on one (or more) of your web pages. The iManager utility will look

for the existence of the iRoot plugin and if found will include an authentication form for the iRoot utility.

### 7.2.2 Running the iRoot Plug-in

The iRoot plug-in is run from your web browser. Enter the following URL into your web browser substituting <domainname> with your domain name:

**`http://www.<domainname>.com/cgi-bin/admin/vs/iroot.cgi`**

This page will prompt you for your root password. Simply enter your virtual server root password and click on the *Start iRoot* button.

### 7.2.3 Adding Email and FTP Users - vadduser

Once you have the iRoot plug-in running, follow these steps:

1. Go to the *vadduser Wizard - Username* section.
2. Enter a new username and click on the *Next* button.
3. You will be taken to the *Password* screen. Enter the new users password and confirm then click on the *Next* button.
4. At the *Full Name* screen enter the users full name and then click on the *Next* button.
5. At the *FTP Privileges* screen, decide whether or not you want the user to have FTP privileges. If you choose *No*, click on the *Next* button to proceed on to the final screen to confirm the user settings you have chosen. If you want them to have FTP privileges, choose *Yes* and click on the *Next* button to proceed on to the *Home Directory* selection screen.
6. At the *Home Directory* selection screen, decide which home directory you would like the user to have then click on the *Next* button.
7. At the *FTP Quota* screen, choose the amount of your disk space you would like to allocate to this user then click on the *Next* button. This will take you to the final screen to confirm the user settings you have chosen.

While using the vadduser Wizard or any of the iRoot Wizards, you can use the *Prev* and *Next* buttons to navigate forward and back to make any changes you need to before you complete the final step.

### 7.2.4 Changing Email and FTP Users' Passwords - vpasswd

Once you have the iRoot plug-in running, follow these steps:

1. Go to the *vpasswd Wizard - Username* section.
2. Select a username and click on the *Next* button.
3. This will take you to the *Password* screen.
4. Enter the users new password and confirm your entry. Then click on the *Next* button to take you to the *Confirmation* screen.
5. If the settings are correct, click on the *Change Password* button to change the password. If they are not correct, click on the *Cancel* button to quit or the *Prev* button to change your entries.

### 7.2.5 Removing Email and FTP Users - *vruser*

Once you have the iRoot plug-in running, follow these steps:

1. Go to the *vruser Wizard - Username* section and select a user to delete.
2. Click on the *Next* button to go to the confirmation page.
3. If the user is the one you want to delete, click on the *Remove User* button.

### 7.2.6 Adding an Email Alias

Once you have the iRoot plug-in running, follow these steps:

1. Go to the *Add Alias Wizard - Alias Name* section to add an Email alias.
2. Choose an alias name and click on the *Next* button to go to the *Alias Type* screen.
3. Select a *simple* alias (see “Creating Email Aliases” on page 32), an *include list* (see “Creating Email Lists” on page 32), or an *autoreply* (see “Creating Email Autore-  
sponders” on page 33).

The rest of the setup is fairly easy to follow. If you have questions as to which alias you need or exactly how you need to implement the alias, include list, or autoreply, please reference the above sections of the manual.

4. You will finally be prompted to click on the *New Aliases* button to run the *vnewaliases* command so your changes will take effect.

### 7.2.7 Deleting an Email Alias

Once you have the iRoot plug-in running, follow these steps:

1. Go to the *Delete Alias Wizard - Alias Name* section.
2. Select an alias name to delete from the list and click on the *Next* button.
3. This will take you to the confirmation screen to verify the alias to delete.
4. Click on the *Remove Alias* button to remove your selection, or click on the *Prev* or *Cancel* button to start over.
5. You will finally be prompted to click on the *New Aliases* button to run the *vnewaliases* command so your changes will take effect.

### 7.2.8 Updating Your Aliases File - *vnewaliases*

Once you have the iRoot plug-in running, follow these steps:

1. Go to the *vnewaliases Wizard* section.
2. Click on the *New Aliases* button to update your aliases file.

If you would rather update your aliases file manually, go to section 4.3 on page 32.

### 7.2.9 Changing Your Server Root Password - *passwd*

Once you have the iRoot plug-in running, follow these steps:

1. Enter your current root password.

2. Enter your choice for a new password and confirm it.
3. Click on the *Change Password* button to update your password.

#### **7.2.10 Creating and Installing a Digital Certificate for SSL Encryption**

Once you have iRoot running, go to the Digital Certificate section to create and install a digital certificate for your server.

A digital certificate allows you to have secure home pages via SSL (Secure Sockets Layer) Encryption. This is especially important if you are asking customers to enter their credit card data into your on-line forms.

For more information on digital certificates and encrypted home pages send us an Email request.

# An Overview of UNIX

---

## A.1 Overview

---

Unlike a traditional introduction to Unix, the emphasis of this one is on philosophy and brevity. When you understand how the creators of Unix intended you to use it, you'll approach Unix on its "best side". This introduction intends to help a new Unix user get started on the right foot quickly. For more information, readers are referred to the Unix manuals and other listed references. As little detail as possible has been duplicated from the manual.

---

## A.2 Why Use Unix?

---

In some ways, Unix is "old technology" - it was invented in the late 1960's for a small computer with a 64K-byte address space, it is largely character oriented (not graphic). Why is it still here? Why is it spreading to more and more systems from PC's to Cray Supercomputers? One answer is that Unix is written in a mostly machine independent way (in the high level language "C") and is therefore more easily moved to new machines. Once Unix has moved, a large base of applications also moves easily and your investment in learning Unix continues to pay off. Another answer is that many problems are still character oriented (or at least can be approached that way) and for these problems, like a sharp tool in the hands of a skilled user, Unix really helps you get

your work done. Also, you can use Unix from any kind of terminal and over dial-up phone lines or computer network connections.

In the space below, I hope to convey, with a minimum of specific information, the essence of “The Unix Philosophy” so that you can use and enjoy Unix at its best. To try to summarize in just two sentences (for those who really believe in such brevity): Unix comes with a rich set of connectable tools which, even if they don’t directly address the problem at hand, can be conveniently composed (using the programmability of the command interpreter) into a solution. Unix also imposes relatively few arbitrary limits and assumptions on the user or the problem domain and has thereby proven to be a suitable platform on which to build many useful and highly portable research and commercial applications.

---

### **A.3 Essential Commands and Concepts**

---

Before I can realistically hope to say more about Unix in general, or give meaningful examples, I must briefly explain some Unix commands and concepts. These descriptions are intentionally minimal. You will soon see how to find more detail in the manuals.

#### **A.3.1 Login**

Unix is a multi-user operating system. This means that several users can share the computer simultaneously. To protect each user’s data from damage by other users, Unix requires each user “login” to the system to identify him/herself (with a login name) and authenticate him/herself (with a password). During the login process, a user’s defaults and “terminal type” are usually established. The mechanism Unix uses to allow concurrent users also allows each user to have more than one program (also called “process” or “commands”) running concurrently. You will see shortly how convenient this is.

#### **A.3.2 The Shell, Commands and Arguments**

Once you have logged in, you will be running a program called your “login shell”. The shell is a program which executes the commands you type in and prompts you when it is ready for input. One of the nice features of the Unix shell is that it is a powerful programming language unto itself, however one need not program it to use Unix. There are several different “shell” programs in common use: csh (c-shell), sh (bourne-shell), ksh (korn-shell), vsh (visual-shell) to name a few. Most people use “csh”.

Unix commands consist of a program name followed by options (or arguments) to that program (if any). One or more spaces follow the program name and separate arguments. Each program examines its argument list and modifies its behavior accordingly. By convention, arguments which begin with a dash are called “switches” or “flags” and they are used to request various non-default program behavior or to introduce other arguments. It is occasionally important to remember that it is the shell which does filename expansion (such as turning “\*.old” into “a.old list.old program.old”). Programs normally don’t ever see unexpanded argument lists. Many Unix programs can also take

implicit arguments. These are available (to every program you run) via the “environment”. Your “terminal type”, stored in an environment variable called `TERM`, is an example of this. The manual for each program you use should list the environment variables it examines and the manual for your shell explains environment variables in detail.

### A.3.3 On-line Manuals

Before getting into any specific commands and examples, note that most Unix systems have both on-line and printed manuals. Many commands will be mentioned below in passing without explanation. It is assumed that the interested reader will look them up in the manual.

The on-line manuals generally contain only the numbered sections of the printed manuals. The tutorials and in-depth articles are usually only in printed form. This introduction intends to reproduce as little of the information contained in the Unix manuals as possible. For more information on any Unix command, type “man command” (“man man”, for example gets you “the manpage” for the on-line manual command: man). (Note: if you are prompted with the word “more”, you are interacting with the “more” program. Three quick things to know: you may type a space to get the next screen full, the letter “q” to quit, or “?” for a help screen.)

Among other things, the man-page for the “man” command points out that “man -k word” will list the summary line of all on-line man-pages in which the keyword: word is present. For example, “man -k sort”, will produce something like this:

```
comm (1) - select or reject lines common to two sorted files
look (1) - find lines in a sorted list
qsort (3) - quicker sort
qsort (3F) - quick sort
scandir, alphasort (3)- scan a directory
sort (1) - sort or merge files
sortbib (1) - sort bibliographic database
tsort (1) - topological sort
```

This tells you that section 1 (user commands) of the manual has man-pages for *comm*, *look*, *sort*, *sortbib*, *tsort*. Use the man command on any of these to learn more. The other numbered sections of the Unix manual are for system calls, subroutines, file formats, etc. You can find out about each section of the manual by saying, for example, “man 2 intro”. Enough about manuals.

### A.3.4 I/O re-direction: `stdin`, `stdout`, `stderr`, pipes

By convention, whenever possible, Unix programs don’t explicitly specify from-where to read input or to-where to write output. Instead, programs usually read from “standard input” (`stdin` for short) and write to “standard output” (`stdout`). By default, standard input is the keyboard you logged in on and standard output is the associated display, however, the shell allows you to re-direct the standard output of one program either to a “file” or to the standard input of another. Standard input can be similarly redirected. Perhaps Unix’s greatest success comes from the ability to combine programs easily (by

joining their standard inputs and outputs together forming a pipeline) to solve potentially complex problems.

“Standard error” (stderr) is not usually re-directed, hence programs which write warnings, prompts, errors, etc. to stderr will write them to the display even when normal input and output is usefully re-directed. (Note that since I/O devices are implemented as files on Unix, I/O re-direction also works to and from physical devices.) The syntax for I/O re-direction is fully described in the manual for the shell you are using (probably csh).

The following are some simple examples of I/O re-direction. For clarity, the shell’s ready-for-input-prompt has been shown as “Ready%” and explanations have been inserted in *italics*. Everything the user would type is shown in slightly bold type after the Ready% prompt.

Running the “date” command prints today’s date and time on standard output

```
Ready% date
Wed Mar 22 13:06:30 PST 1989
Ready%
```

Put the standard output from the date command in a file called “myfile”

```
Ready% date > myfile
Ready%
```

Use the word-count program to count the number of lines, words, characters in “myfile”

```
Ready% wc < myfile
1 6 29
Ready%
```

Pipe the output of the date command directly into the word count command. Note that commands in a pipeline such as this can run simultaneously.

```
Ready% date | wc
1 6 29
Ready%
```

Use output from one program as command line arguments to another

```
Ready% echo My computer, 'hostname', thinks today is 'date'
My computer, samburu, thinks today is Wed Mar 22 13:06:30 PST 1989
Ready%
```

Look in the on-line dictionary for words beginning with “pe” and count how many are found

```
Ready% look pe | wc
294 294 2548
Ready%
```

Pipe those 294 lines through `cat -n` to insert line numbers and then through `sed` to select only lines 5-8

```
Ready% look pe | cat -n | sed -n 5,8p
5 peaceful
6 peacemake
7 peacetime
8 peach
Ready%
```

Now, from those 294 words, select only those containing “va” somewhere and re-direct them into the argument list of the `echo` command

```
Ready% echo I found these: `look pe | grep va`.
I found these: Pennsylvania Percival pervade pervasion pervasive.
Ready%
```

`Grep` (search) through all files with names ending in “.c” for lines beginning with “#define”. (`Grep -l` lists the file names containing the lines which match instead of the lines themselves). These file names are redirected to form the command line of the `vi` editor - hence, edit all “.c” files which contain “define” statements.

```
Ready% vi `grep ^#define *.c`
The depiction of an interactive session with the “vi” editor is omitted.
Ready%
```

### A.3.5 Special characters: Interrupt, End-Of-File, Quoting, ‘Job Control’

When a program reads from a file or from a pipe it can tell when there is no more to read. This condition is called reading the “end-of-file” or EOF. When standard input is a terminal, the EOF must be explicitly typed because the program must otherwise assume you are still typing. Normally EOF is typed as a CONTROL-D (indicated in print as `^D`). Think of the control key as another SHIFT key - it must be pressed and held when the D is typed. If the EOF is not the first thing on a line, two must be typed.

If you are running a program and you wish to interrupt it completely, you can often do so by typing `^C`. You can try this with the “wc” program (run `wc` then interrupt it):

```
Ready% wc
sample input
^C
Ready%
```

run `wc` then type EOF

```
Ready% wc
sample input
^D
1 2 13
Ready%
```

Note that both `^D` and `^C` ended the program however, `^D` allowed the program to finish normally but `^C` killed it (and produced no output). If, for some reason, you want to type a special character such as `^C` and actually have it sent to your program and not generate an interrupt, you can “quote it” by typing a backslash (or sometimes a `^V`) before it. The backslash also “quotes” shell “meta-characters” such as asterisk, question mark, double-quote, backslash, etc.

“Job control” is the name given to an extremely convenient feature of many modern versions of Unix. Job control allows one to suspend a program and resume it later. If you are in the middle of running some program when the phone rings, you can type `^Z` to suspend the program (and get back to your shell prompt) without interrupting or exiting that program. After you handle the phone call, you can type `fg` to resume the original program right where you left off. Unix permits one to have a fairly large number of suspended jobs and to resume them in any order. Csh’s `jobs` command displays which jobs are stopped. (In some ways, job control is “a poor man’s window system”; however, even on Unix systems with windows, many people find job control indispensable.) For more information on job control, see the `csh` man-page.

### **A.3.6 Files, Permissions, Search PATH**

Unix files exist in directories. Every user has a “home directory”, which is the “current directory” after logging in. A user can make “sub directories” with the `mkdir`, command and make them the current directory with the `cd` command. You can print your current directory with the `pwd` command and you can refer to the parent directory as `..` (two dots). You can get back to your home directory by typing `cd` with no arguments.

Files and directories have permissions called “modes” which determine whether you, “your group”, or everyone can: read, write, or execute the file. Permissions are changed with the `chmod` command. The main reason for bringing this up now is to point out that a collection of commands which can be typed to the shell can also be put in a file, given a name, made executable and subsequently invoked as a new command by that name. This type of file is called a “shell script” and is one of the main ways Unix is customized to the work habits and chores of its users.

When a user types a command, s/he usually doesn’t type the full (and unambiguous) path name of the program: `/bin/date` for example) but instead types only the last component of the path name, `date`, thus requesting the system to search for it. To achieve predictability and efficiency, the system searches only those directories listed in your `PATH` environment variable and it searches them in that order. By placing your own version of a program in a directory you search before the system directories, you can override a system command with your own version of it.

Your version can be anything from an entirely different program to a simple shell script which supplies some arguments you always use and then calls the standard version. The command `echo $PATH` will print the value of the `PATH` environment variable to std-

out. The procedure for setting environment variables such as PATH differs from shell to shell. See the man-page for the shell you use.

---

## **A.4 The Unix Philosophy**

---

Well, so much for the nitty-gritty. I will now try to explain “The Unix Philosophy” in a bit more detail. Basically, the idea is that rather than have a custom program for each little thing you want to do, Unix has a collection of useful tools each of which does a specific job (and does it well). To get a job done, one combines the pieces either on the command line or in a shell script. For example, on Unix, a user would not expect an application to provide an input text editor. Instead, one would expect to be able to use one’s favorite (and standard) “text editor” (probably “vi”, perhaps “emacs”) for all instances of editing text. Electronic mail, C programs, shell scripts, documents-to-type-set can all be edited with the same text editor. By convention, applications invoke the text editor you have specified in your EDITOR environment variable.

Even though Unix editors are generally very powerful and capable programs, they too recognize that they are just tools and they allow you to pipe all or part of your “editor buffer” through any pipeline of Unix commands in order to do something special for which there isn’t a built-in editor command. (The editor buffer is that private copy of your file to which the editor makes changes before you save them.)

Unlike most other operating systems, Unix has only one “file type”. Any program which can read or write standard I/O can read/write any “file” (even if it is a device such as a terminal, printer or disk). Granted, not every program can make sense out of the data in every file, however, that is strictly between the program and the data - nothing imposed by Unix. The single file-type contributes greatly to the modular/re-usable pipes-and-filters approach to problem solving.

So, what is to be learned from all this? Just that it is good to construct solutions to your problems in as general and modular a fashion as possible. You will undoubtedly find that a somewhat general program (or shell script) you wrote as part of the solution to one problem will be just what you need as part of the solution to some future problem and it will be simple to hook up.

---

## **A.5 A ‘Typically Unix’ Solution**

---

Let’s assume the following problem, inspired by a real-world situation. You are a professor of English and someone walks into your office with an old manuscript claiming it is an undiscovered work by Shakespeare. You postulate (correctly) that you can use statistics about frequency of word usage to help determine its authenticity. The problem, therefore, is to come up with a histogram (count) of the number of times each word is used.

You could, of course, write a program from scratch in C or FORTRAN to do it, however a partial solution comes to mind using “awk”, a programmable text processing tool which has 2 particularly useful features: 1) lines are read and processed automatically;

2) arrays can have text-string subscripts. So, if you hadn't already written a "histogram" shell script, you write one now. (Keep it around, you will find a use for it again.) The file "histogram" has the following contents (de-mystified somewhat below):

```
awk `
NF > 0 { counts[$0] = counts[$0] + 1; }
END { for (word in counts) print counts[word], word; }
`
```

For each line with  $NF > 0$  ( $NF$  is awk-talk for number-of-fields-on-this-line, hence for each non-empty line), add 1 to that particular counter hereby associated with the-text-on-this-line ( $$0$  is awk-talk for the-text-on-this-line). Then, at the END of input, for each unique input line; print that line preceded by the count of how many times it was seen.

So, now the task is simply getting the input into a format where all punctuation marks are removed and each word appears on a line by itself. Again, you could write a program to do it; you could manually reformat the text with an editor; or you could notice that Unix has a translate command "tr" which will do just what you want when used in two steps as shown:

```
tr -dc "a-zA-Z' \012" | tr " " "\012"
```

The first "tr" command has options -dc (delete the complement of the indicated characters) so it will delete from standard input all characters except those which are listed (letters, apostrophe, space, and octal 012 (newline)). The resulting output has no punctuation. The second "tr" translates all spaces into new lines, thus causing at most one word to be on each line.

Piping the output of these two commands into "histogram" will give us word counts. Piping the output of histogram into "sort -n" will sort the histogram in numerical order. Putting the whole thing in a file and making it executable makes it available as conveniently as if it had been built into Unix.

Here then is some sample input and the output our script produces:

```
One black bug bled blue black blood
while another blue bug bled black.
```

And the output of `tr ... | tr ... | histogram | sort`:

```
1 One
1 another
1 blood
1 while
2 bled
2 blue
2 bug
3 black
```

Note that other simple solutions to the problem exist. Our awk-based histogram program can be replaced by "sort | uniq -c" (but that is less intuitive than the awk solution

and not necessarily any better). Also, “sed” could have been used in place of either or both of the “tr” commands. (Sed is much more powerful than tr however the sed command line would have been less intuitive.)

---

### A.6 More about Pipelines and Concurrent Execution

---

Probably the two biggest advantages of concurrent execution of commands in a pipeline are: 1) No disk space is required for intermediate data which flows through pipelines. 2) output can start coming out the end of the pipeline before the entire input is processed through the first program in the pipeline.

For example, imagine you want to compute a histogram for a very large file which is compressed and your disk is too full to hold the uncompressed version. You can uncompress it to standard output and pipe that directly into your histogram pipeline.

Now imagine you have a pipeline which takes 30 minutes to compute and produces data which takes 30 minutes to print. If you first computed and then printed, it would take 60 minutes. If you re-direct the output of the pipeline to the printer, the whole process only takes 30 minutes. (Note: you can output directly to a device such as a printer but in a multi-user environment the normal printing mechanism is to spool the output in a file (with “lpr”) and print it after the computation finishes.)

On Unix you can run any number of programs “in the background”, which means that the shell doesn’t wait for them to finish before giving you a new prompt. Read more about this in the manual for your shell.

You can also have programs started for you automatically at certain times of the day, week, month, etc. (read about “at” and “cron”) or when certain events happen, such as when electronic mail arrives.

---

### A.7 Other Especially Useful Unix Programs to Read About

---

Since it is not the intent to duplicate information from the Unix manual in this introduction I won’t give many details about the following programs, however, I would like to point them out so you can look them up in the manual if you are interested. Most manual pages have a “SEE ALSO” section at the end. Consider yourself invited to read up on those programs as well. (If you really want to know everything, look up every program in every directory in your \$PATH!)

- **learn** An interactive tutorial on a few subjects. (Not available on all systems). Probably most useful for learning the “vi” editor. Type “learn vi” to try it.
- **vi, emacs, ex, ed, pico** “vi” is the most common Unix screen-oriented text editor. Emacs can be another good choice. (“ed”, the original Unix text editor is essentially subsumed by vi and is much harder to use.) “ex” is really just vi in a non-screen-oriented (ed-like) mode. “pico” is a very easy to use but not very powerful screen-oriented editor. There are substantial printed manuals on vi, ex and emacs. Whichever

editor you choose, you will eventually want to read everything there is to know about it. Unix editors are very powerful and knowing how to use that power really helps a lot.

- **rm, mv, cp, rmdir** Remove; move (rename); copy a file; remove a directory.
- **ls** List directories. More options than just about any other program. Filenames which begin with dot are not listed unless the **-a** option is used.
- **stty, tset** Set such aspects of terminal I/O as: number of lines on display device, input character- or line-at-a-time; whether keyboard typing is visible.
- **cat** Concatenate files to standard output.
- **more, less, page, pr** Display data a screen or page at a time. Search and skip forward to a page of interest.
- **cmp, comm, diff, diffmk** Show differences between two files.
- **grep** Find lines which match specified pattern (incredibly useful).
- **rlogin, rsh, rcp** Login to remote Unix system, run a command on remote Unix system, copy a file to remote Unix system. Similar to below.
- **telnet, ftp** Connect to remote system of arbitrary type, copy a file.
- **talk, mail, mh, mm** Connect your terminal to another user for interactive communication. Send or read electronic mail.
- **crypt, pgp** Encrypt or decrypt data.
- **compress** Compress data or files, typical compressions are 2-3 to 1.
- **tar, cpio** Archive and restore files and directories into/from a single file on disk or removable media.
- **sed** Probably the single most useful command for rearranging or extracting pieces of data quickly. (A bit cryptic for many users, though.)
- **awk** More powerful than sed but somewhat slower; almost a general purpose programming language but definitely tailored to filtering text from stdin to stdout.
- **head, tail** First, last part of a file or stdin.
- **find** Locate files which meet specified criteria.
- **look, spell** Look up words in an on-line dictionary. Find spelling errors.
- **sum** Compute a CRC (checksum) for comparison with supposedly identical data on a remote system.
- **od** Display an octal (or hex) dump of input data. This lets you see every byte of your data as a bunch of numbers.
- **du, df** Display disk usage and free disk space.
- **script** Keep a transcript of your session in a file.
- **who, whoami, su** Who is on the system. What is my username? Become another user temporarily.
- **ps, kill** Process-status lists attributes and resources associated with each process. Kill sends to a process a “signal” which (depending upon the signal sent) will cause the process to terminate in various ways. See also the man-page for “signal”.

## A.8 Other Sources of Information

---

1. *4.4BSD Unix Manuals*, U.C. Berkeley, published by O'Reilly & Associates, Inc. 103 Morris Street, Suite A, Sebastopol, CA 95472
2. *The Unix Programming Environment*, Kernighan, B.W. and R. Pike, Prentice Hall, Englewood Cliffs, N.J.
3. *Welcome to Unix*, Rick Ells, Academic Computing Services, University of Washington, Seattle, Washington.
4. *Introducing the Unix System*, Henry McGilton, McGraw-Hill Software Series.
5. *The C Programming Language*, Kernighan, B.W. and Ritchie, D.M., Prentice Hall, Englewood Cliffs, N.J.
6. *Introducing Unix System V*, Morgan, R., McGilton, H., McGraw-Hill Software Series.
7. *Unix for People*, Birns, P., Brown, P., Muster, J.C.C, Prentice Hall, Englewood Cliffs, N.J.
8. *Unix for Dummies*, John R. Levine and Margaret Levine Young, IDG Books Worldwide, Inc., Foster City, CA.
9. *Advanced Unix Programming*, Marc J. Rochkind, Prentice-Hall, Englewood Cliffs, N.J.

