Installation of Apache Web and Proxy Server

The Internet as we know it today has its roots in the data network devised by the DARPA for military and educational use. It was released as a public commercial network only in 1995 after corporations and the general public started taking interest in this flat fee structured, homogeneous and apparently secure data transfer medium. It suddenly started to take the shape of an interactive medium for mass publicity and consumer interaction.

With its growing popularity and pertinacious decrease in access costs, a new demand for interactive services, viz. Audio, video, telephony etc. arrived. This demand was further pampered with the growth in media technology coupled with protocols that provided near digital quality media at far lesser bandwidths. For a general reminder, the basic skeleton of the Internet is based on the TCP/IP protocol suite.

Throughout this paper, we will aim to introduce a beginner's insight into the seemingly arduous process of initiation of interactive services over a TCP/IP network.

Key Ingredients

Whenever we speak of a service, we mean the provision of some data in the form of audio, video, text or a combination of all three. This scheme is analogous to the client server model of networks, wherein a central server PC serves the needs of all clients (less powerful PC's). A server is a complete piece of code working on an equally powerful hardware. The hardware can be as simple as a tiny microchip on a PCB or as complex as a multi-processor, multi-port, parallel processing PC. It is the server software that determines the capacity of the whole server. With the tremendous proliferation in the extent of the Internet and the WWW, a universal protocol for the transmission of simple data and/or pictures etc. became apparent. This protocol became known as the Hypertext Transmission Protocol or the HTTP. This protocol sits on the application layer of the OSI Model and is concerned only with the transportation of Hypertext (a formatting markup language) between two processes on different hosts.

Now that it is established that the protocol used for transferring data and multimedia on the Internet is HTTP, we need to know the server software supporting this protocol and which are extensively used.

There are many HTTP servers available in the market. The most common of them all until early 1995 was the NCSA httpd server. Other notable servers are: -

1.W3C httpd

2.Netsite Server (Netscape Communications)
3.Spinner

4.AOLServer

5.NT (IIS)

NCSA httpd was the most widely used web server until 1995, when started on a new Open Source web server - The Apache Web server. This server was to take the market by storm and become the most dominant server software in the industry.

The Apache Web Server: -

Apache was originally based on code and ideas found in the most popular HTTP server of the time, NCSA httpd 1.3. It has since evolved into a far superior system that can rival almost any other UNIX based HTTP server in terms of functionality, efficiency and speed. It has come to offer many new features including a port to the upcoming Windows NT server. As of January 1997, it is the most popular WWW server on the Internet, according to the Netcraft survey.

The origin of the Apache Server can be traced back by this anecdote -? At that time, CERN httpd was widely used, but an inadvertent security hole was detected in it. This prompted developers around the world to get together to go on to develop a web server which performed the way they wanted.

The most remarkable fact of the Apache web server is that it's open source, meaning that everyone is free to download it and make any changes to its source code, as they want. Even though it is free and not developed by dedicated professionals, it is the most deployed web server in the Internet with about 50% of the sites running on it. It is also one of the fastest.

Apart from being open source, Apache is feature packed and very lightweight. A full download of the source is only about 800 KB. It supports pre-forking and threading. This means that it runs copies of itself in the memory space of the server and is able to do so dynamically and automatically. Pre-forking lessens the server ramp-up period, as the software is able to start-up a minimum no. of child server processes in memory in anticipation of the coming load of requests. Although it does not as yet support concurrency and parallel processing support, the developers propose to add these and many more such features in the upcoming versions.

Apache follows a modular paradigm in its basic architecture. All the features are built on top of the core model in modules. This allows for seamless product development. Additional new features can be added on top of the server just by adding the respective modules in the server configuration. This allows for portability and reduces the server size as only required modules need be compiled and run. The developers can lay more stress on developing particular modules and thus perfect them instead of having to rebuild the entire source tree for each minor improvement.

Apart from its modular architecture, Apache features much more such as user authentication, CGI, image maps, content negotiation, proxy facilities, cookies, virtual hosts and a very versatile dynamic shared objects (DSO) support. Apache can behave as a full featured Proxy all by itself and thus eliminates the need for any other proxy software. The virtual host feature enables a single Apache based server to serve for many virtual host names on a single machine. It supports both IP and name based virtual hosts. More details regarding virtual hosts can be found at the Apache docs. Another plus with Apache is that it is very easy to configure and get running. Although, new users sometimes get into a spot with configuring it, the accompanied help files are more than enough to solve any configuration and/or compiling problem. The installed server has utmost 3 configuration files which have in-built tags to enable even the most novice to safely get the server running with the maximum of functionality. A list of sites filled with Apache information is appended at the bottom of this report. We will now shift our focus from what Apache is, to how to compile it and configure it for our purpose on a typical Red Hat Linux distribution.

Downloading Apache

Information on the latest version of Apache can be found on the Apache web server at http://www.apache.org/. This will list the current release, any more recent beta-test release, together with details of mirror web and anonymous ftp sites.

Compiling Apache (The General Overview)

Compiling Apache consists of three steps: Firstly select which Apache modules you want to include into the server. Secondly create a configuration for your operating system. Thirdly compile the executable. All configurations of Apache are performed in the src directory of the Apache distribution. Change into this directory.

1. Select modules to compile into Apache in the Configuration file. Uncomment lines corresponding to those optional modules you wish to include (among the AddModule lines at the bottom of the file), or add new lines corresponding to additional modules you have downloaded or written. (See API.html for preliminary docs on how to write Apache modules). Advanced users can comment out some of the default modules if they are sure they will not need them (be careful though, since many of the default modules are vital for the correct operation and security of the server). You should also read the instructions in the configuration file to see if you need to set any of the Rule lines.

2. Configure Apache for your operating system. Normally you can just type run the Configure script as given below. However if this fails or you have any special requirements (e.g., to include an additional library required by an optional module) you might need to edit one or more of the following options in the Configuration file: EXTRA_CFLAGS, LIBS, LDFLAGS, INCLUDES.

Run the Configure script:

% Configure

```
Using 'Configuration' as config file
+ configured for <whatever> platform
+ setting C compiler to <whatever> *
+ setting C compiler optimization-level to <whatever> *
+ Adding selected modules
+ doing sanity check on compiler and options
Creating Makefile in support
Creating Makefile in main
Creating Makefile in os/unix
Creating Makefile in modules/standard
```

(*: Depending on Configuration and your system, Configure might not print these lines. That's OK).

This generates a Makefile for use in stage 3. It also creates a Makefile in the support directory, for compilation of the optional support programs.
(If you want to maintain multiple configurations, you can give an option to Configure to tell it to read an alternative Configuration file, such as Configure -file Configuration.ai).

3. Type *make*.

The modules we place in the Apache distribution are the ones we have tested and are used regularly by various members of the Apache development group. Additional modules contributed by members or third parties with specific needs or functions are available at <http://www.apache.org/dist/contrib/modules/>. There are instructions on that page for linking these modules into the core Apache code.

Installing Apache

You will have a binary file called httpd in the src directory. A binary distribution of Apache will supply this file.

The next step is to install the program and configure it. Apache is designed so as to be configured and run from the same set of directories, where it is compiled. If you want to run it from somewhere else, make a directory and copy the conf, logs and icons directories into it. In either case you should read the security tips describing how to set the permissions on the server root directory.

The next step is to edit the configuration files for the server. This consists of setting up various directives in up to three central configuration files. By default, these files are located in the conf directory and are called srm.conf, access.conf and httpd.conf. To help you get started there are same files in the conf directory of the distribution, called srm.conf-dist, access.conf-dist and httpd.conf-dist. Copy or rename these files to the names without the -dist. Then edit each of the files. Read the comments in each file carefully. Failure to setup these files correctly could lead to server not working or being insecure. There should be also an additional file in the conf directory called mime.types. This file usually does not need editing.

First edit httpd.conf. This sets up general attributes about the server: the port number, the user it runs as, etc. Next edit the srm.conf file; this sets up the root of the document tree, special functions like server-parsed HTML or internal imagemap parsing, etc. Finally, edit the access.conf file to at least set the base cases of access.

In addition to these three files, the server behavior can be configured on a directory-by-directory basis by using .htaccess files in directories accessed by the server.

Setting the system time properly is also quite important.

Proper operation of a public web server requires accurate time keeping, since elements of the HTTP protocol are expressed as the time of day. So, it's time to investigate setting up NTP or some other time synchronization system on your Unix box, or whatever the equivalent on NT would be.

Starting and Stopping the Server

To start the server, simply run httpd. This will look for httpd.conf in the location compiled into the code (by default /usr/local/apache/conf/httpd.conf). If this file is somewhere else, you can give the real location with the -f argument. For example:

```
/usr/local/apache/httpd -f /usr/local/apache/conf/httpd.conf
```

If all goes well this will return to the command prompt almost immediately. This indicates that the server is now up and running. If anything goes wrong during the initialization of the server you will see an error message on the screen. If the server started ok, you can now use your browser to connect to the server and read the documentation. If you are running the browser on the same machine as the server and using the default port of 80, a suitable URL to enter into your browser is

http://localhost/

Note that when the server starts it will create a number of child processes to handle the requests. If started Apache as the root user, the parent process will continue to run as root while the children will change to the user as given in the httpd.conf file.

When httpd is run, if it complains about being unable to "bind" to an address, then either some other process is already using the port that have been configured for Apache to use, or running httpd as a normal user but trying to use a port below 1024 (such as the default port 80).

If the server is not running, read the error message displayed when you run httpd. You should also check the server error_log for additional information (with the default configuration, this will be located in the file error_log in the logs directory).
If you want your server to continue running after a system reboot, you should add a call to httpd to your system startup files (typically rc.local or a file in an rc.N directory). This will start Apache as root. Before doing this ensure that your server is properly configured for security and access restrictions.

To stop Apache send the parent process a TERM signal. The PID of this process is written to the file httpd.pid in the logs directory (unless configured otherwise). Do not attempt to kill the child processes because they will be renewed by the parent. A typical command to stop the server is:

```
kill -TERM `cat /usr/local/apache/logs/httpd.pid`
```

For more information about Apache command line options, configuration and log files, see Starting Apache at the Apache Manual directory in your installation. For a reference guide to all Apache directives supported by the distributed modules, see the Apache directives in the same directory.

Compiling Support Programs

In addition to the main httpd server which is compiled and configured as above, Apache includes a number of support programs. These are not compiled by default. The support programs are in the support directory of the distribution. To compile the support programs, change into this directory and type

```
make
```

Compiling Apache (Step By Step Overview)

This is the most important step in the deployment of The Apache Web Server. The source downloaded from the Internet can be put to use only after compiling it on the server machine. As the brief overview has already been supplied elsewhere, a practical example is given below.

An important aspect of installing any package on a Linux machine is its ownership. As is apparent, the web server is meant to be running on the machine all the time, irrespective of the users logged in. If the ownership were to be bound to a particular unprivileged user, only that user would be able to initiate and/or modify the configuration parameters of the server. Thus, any daemon (or server) is always installed by the root with full privileges.

Another important facet is the location of the installed binaries and the documentation. The files should be installed such that they are easily traceable by any user and that follows a general scheme. Generally, all the daemons binaries reside in the /usr/bin or the /usr/sbin directories. These directories are added automatically in the path settings of all local users. These are more frequently than not, links to actual binary files resident on other directories. The full installation should always be placed in the /usr/local directory.

Choosing which modules to include is a decision, dependent solely on the purpose/role of the server. Some of the modules accompanied by Apache are compiled in by default, while some others need to be specially included in the compilation. The modules included in the Apache 1.3.12 distribution are :-

(1) Environment structure

(a) mod_env (default)
(b) mod_setenvif (default)
(c) mod_unique_id (not by default)

(2) Content type decision

(a) mod_browser (default)
(b) mod_mime (default)
(c) mod _mime_magic (not by default)
(d) mod_negotiation (default)
(e) mod_alias (default)
(f) mod_rewrite (advanced) (not by default)
(g) mod_userdir (default)
(h) mod_spelling (not by default)
(i) mod_dir (default)
(j) mod_autoindex (default)
(k) mod_access (default)
(l) mod_auth (default)
(m) mod_auth_dbm (Unix NDBM) (not by default)
(n) mod_auth_anon (not by default)
(o) mod_digest (not by default)

(3) HTTP response

(a) mod_headers (not by default)
(b) mod_cern_meto (not by default)
(c) mod _expires (not by default>
(d) mod_asis (raw HTTP) (default)
(e) mod_include (SSI) (default)
(f) mod_cgi (not by default)
(g) mod_actions (default)

(4) Internal Content Handlers

(a) mod_status (not by default)
(b) mod_info (Server config summary) (not by dafault)

(5) Request logging

(a) mod_log_config (default)
(b) mod_log_agent (not by default)
(c) mod_log_referer (not by default)
(d) mod_usertrack (cookies) (not by default)

(6) Misallenious

(a) mod_imap (default)
(b) mod_proxy (not by default)
(c) mod_so (not by default)

(7) Experimental

(a) mod_mmap_static (not by default)
(b) mod_example (not by default)
(c) mod_isapi (default)
(d) mod_vhost_alias (not by default)

The included modules are mod_mime_magic, mod_rewrite, mod_expires, mod_cgi, mod_status, mod_proxy, mod_so, mod_vhost_alias.

Now, many of these modules are not compiled in by default as their use depends on the role of the server. As for example, a typical web-server may require CGI functionality, but PHP may not be required by it. Thus, only a minimal set of modules, which are deemed necessary for all configurations are compiled in by default and the rest are to be enabled by the editing of the configuration file in the src directory.

Modules required by us for the Web Services

We aim to provide a set of services (accessible through the internet) which are a basis for the successful deployment of interactive operativity of the system. A database mechanism becomes evident in this scenario, as every service has some subscribers and to maintain a record of the subscribers and their preferences requires a database. Keeping this point in mind, it becomes imminent for us to install a web-based front-end to the popular database systems in the market, viz. Oracle, mSQL, PostgreSQL, et al. We promptly chose PHP (PHP Hypertext Preprocessor) for the purpose. More information about the PHP module will be available later .

But the problem arises when it is found that PHP support is not included by default in the Apache distribution and that it has to be downloaded from elsewhere. Thus, it is a good idea to selectively install Apache with the supplied modules and then add the PHP module as a Dynamic Module, using Apache's Dynamic Shared Object (DSO) support.

Given below is a brief description of the modules which are not compiled in by default but which we chose to compile.

1. mod_mime_magic module

This module can be used to find out the MIME type of a file by looking at a few bytes of its contents. It is derived from the Unix file command which uses "magic numbers" and other hints from a file's contents to figure out what the file contents are. When compiling an Apache server, this module should be at or near the top of the list of modules in the Configuration file. Modules are listed in increasing priority so that will mean this one is used only as a last resort, just like it was designed to.

To use mod_mime_magic you have to enable the following line in the server build Configuration file:

```
AddModule modules/standard/mod_mime_magic.o
```

This should be listed before mod_mime in the build Configuration file so that it will be used after mod_mime.
mod_mime_magic is intended as a "second line of defense" for cases mod_mime cannot resolve.

2. mod_rewrite module

This module uses a rule-based rewriting engine (based on a regular-expression parser) to rewrite requested URLs on the fly. It supports an unlimited number of rules and an unlimited number of attached rule conditions for each rule to provide a really flexible and powerful URL manipulation mechanism. The URL manipulations can depend on various tests, for instance server variables, environment variables, HTTP headers, time stamps and even external database lookups in various formats can be used to achieve a really granular URL matching.

This module operates on the full URLs (including the path-info part) both in per-server context (httpd.conf) and per-directory context (.htaccess) and can even generate query-string parts on result. The rewritten result can lead to internal sub-processing, external request redirection or even to an internal proxy throughput. But all this functionality and flexibility has its drawback: complexity. So don't expect to understand this entire module in just one day.

Since our servers are meant to cater to a wide array of service demands, a set of rules for URL-to_filename mapping may make complex tasks easier. Such rules are also required from the server security point of view.

3. mod_expires module

This module controls the setting of the Expires HTTP header in server responses. The expiration date can set to be relative to either the time the source file was last modified, or to the time of the client access.

The Expires HTTP header is an instruction to the client about the document's validity and persistence. If cached, the document may be fetched from the cache rather than from the source until this time has passed. After that, the cache copy is considered "expired" and invalid, and a new copy must be obtained from the source.

When a service has many subscribers/users, the interim relief received in the form of fewer requests per page per visit drastically cuts short the server load. The expiration date setting takes care that the document shown on the user's screen is not outdated.

4. mod_status module

The Status module allows a server administrator to find out how well their server is performing. A HTML page is presented that gives the current server statistics in an easily readable form. If required this page can be made to automatically refresh (given a compatible browser). Another page gives a simple machine-readable list of the current server state.

The details given are:
The number of children serving requests
The number of idle children
The status of each child, the number of requests that child has performed and the total number of bytes served by the child (*)
A total number of accesses and byte count served (*)
The time the server was started/restarted and the time it has been running for
Averages giving the number of requests per second, the number of bytes served per second and the average number of bytes
per request (*)
The current percentage CPU used by each child and in total by Apache (*)
The current hosts and requests being processed (*)
(*) These data are supplied only if the ExtendedStatus directive is set to on in the configuration file.

5. mod_usertrack module

As the name suggests, this module helps keep track of the visitors of a site by tracking their clicks/requests with a file called cookie. The expiration date of a cookie can be set in the server configuration directives. This module helps in keeping a record of the surfing habits of a visitor and then can be used to tune the site according to user's needs. It also helps in building statistics for the site.

6. mod_so module

This is an experimental module. On selected operating systems it can be used to load modules into Apache at runtime via the Dynamic Shared Object (DSO) mechanism, rather than requiring a recompilation. A major advantage is the saving of the server load as the modules load only when required. In the major distributions available in the market viz. Redhat, Debian etc. nearly all the modules are supplied in the DSO form. This makes the server very fast and avoids crashes.

7. mod_auth_db module

This module provides for user authentication using Berkeley DB files. The main purpose of this module is to restrict access to certain resources on the server on the basis of either a group or a user file. Meaning that only certain users are able to access the protected resources. This mechanism is very useful for our purpose as we may provide certain subscription based services and would like to restrict access to them to only the visitors who pay for them. This kind of authentication is more secure than the kind used in Web-Based email (SQL) and thus needs to be deployed only for critical resources. It uses an encrypted file format so that the visitors are unable to download them and/or decipher them.

8. mod_proxy module

This module, as the name suggests, adds proxy capabilities to the Apache server. Apache can behave as a full-fledged Proxy for the entire subnet or the intranet. This means that only the server need be connected to the external world, whilst the entire intranet/subnet looks to Apache for interaction with the outside network or the Internet. This setup is an alternative to the firewall mechanism employed in data sensitive networks. The server can also exhibit caching capabilities, thus saving on the external bandwidth required by the Organisation. Proxy also improves security and user monitoring by restricting access to undesired sites.

Now, the main advantage of these capabilities in our setup is that we may successfully incorporate a large number of client PC's or hosts on our subnet without having to allocate domain specific IP resources. The above advantages notwithstanding, this also improves efficiency by making accessible the vast data resources in and outside the organisation. Security is another issue that is satisfied. The need for deployment of any other proxy server like Squid is thus eliminated.

Configuring the Server

Now that the thinking part is over, time has come to configure the server. As has already been touted, Apache features very easy configuration. Before we go into more details, it would be worthwhile to know something more about a typical web server.
A typical web server contains the following subdirectories :
conf : The directory contains all the Configuration files used to control the operations of the server.
logs : The directory where the access log file and the error log files are usually kept.
cgi-bin : This directory contains sample gateway scripts and precompiled gateway binaries. If

you build your own gateway, this is usually where you will put them. The directory is called cgi-bin because gateways use a standard interface with the Web server called the Common Gateway Interface.
icons : This directory contains an assortment of icons that are used for directory indexing. other directories : src, support, htdocs, Browser.
A word about the configuration files. Previously, there used to be four different files for the server configuration. These were :
httpd.conf : Main server configuration file
srm.conf : Server resource configuration file
access.conf : Global Access Control File (ACF)
mime.types : mime types to be declared in this file

Although still in use in some servers, this setup has largely been changed to a single file configuration. Now only a single file httpd.conf is used to configure the entire Apache server. This can however be overridden and four separate files can be used as above. As configuring a single file is easier than setting up four different files, we chose to follow the unary file setup and all setup information was thus incorporated in the httpd.conf file. A sample httpd.conf file is attached in the Appendix. There is nothing much to be explained as far as the setup/configuration goes as the conf file is already very well documented.

The server is up !

Now that the server is configured, all that remains to be done is to start it and then test it using the steps outlined previously.

A word about CGI

CGI stands for Common Gateway Interface and is the de-facto interface for server - side scripting. CGI is used to serve as an interface between the machine (server) and the host. It provides interactivity through executing scripts on the fly. CGI supports a number of scripting languages viz. Perl, C, Java, PHP etc. All these languages have a common goal; to provide an interactive face to the network. Common uses of CGI include provision of specialized pages to the viewer, creating accounts etc.

CGI is not a language but a protocol to serve as a standard for other scripting languages. It is a better option to employ server side scripting rather than client side as it is always faster, takes no time to download and is secure.

A sample CGI script is also attached in the appendix C.

The Database System

A database system is an integral part of any service platform . The most evident use of any database is its capacity to maintain user/customer information and details. Although a database can be defined as just a collection of simple data in a specified format, the term has come to denote specific technologies. A database file can be as simple as a text file or as complex as a multi-formatted, multi-encoded binary file. An obvious use of using the latter is its speed. Text files, although simple to generate, require tremendous processing power to employ even simple search techniques. For example in the email services, the database of users and their corresponding password is a must. That's why there is increasing demand for databases during a webserver configuration. The recent application of databases is in putting the results of competitive examinations. In many places, for conducting examinations through computers, as the examinations like GRE etc. The databases are also used for keeping databases of crimes and the record of past police records with photographs, fingerprints and other informations. This helps in solving the cases as police authorities can access the database from anywhere. For this reason, much database server software has been developed, viz. Oracle, mSQL, Postgresql etc.

The Choice

Now that the role of a database system on a services platform has been clearly explained, it is time to choose among the many available, the server suitable to our setup and within our resources. With this in mind, a clear winner turns out to be PostgreSQL.

PostgreSQL is a sophisticated Object-Relational DBMS, supporting almost all SQL constructs, including subselects, transactions, and user-defined types and functions. It is the most advanced open-source database available anywhere. Being Open Source means that it is free to be deployed anywhere and is developed by a team of software professionals using the power of the Internet. It can also be modified freely by any user to incorporate desired modifications / improvements. Although, 90% of the sites today use Oracle's server, it is very expensive to deploy and the only major advantage offered by it is the speed of processing queries. Another advantage of using PostgreSQL is that it was developed entirely for the Unix/Linux/Solaris operating systems, which are a benchmark in themselves for their sheer power and durability. PostgreSQL also supports many multi-platform front-ends so that users are able to interactively administer the server using GUI's etc. One such interactive front-end is included with the distribution.

The characteristics of the database programming are following:

(1) Data is stored in a database made up of tables; the tables can be real or virtual. Users are concerned to the whole database and not to individual tables.

(2) The structure of table is defined within the database, not within each program.

(3) The order of table within the database is not important. They are identified by name.

An O-R (Object-Relationship) model for this would be as follows.

```
postmaster    Unix user
    |         |
    |         |
   database system
        |
       /|\
     database
```

That is, there is a one-to-one correspondence between postmasters, Unix users and database systems, and a database system is associated with any number of databases.

It turns out that this is not quite true. It later turned out that a database system is administered by a single postmaster, but any number of Unix users. The `postmaster' is a privileged Unix user which is a kind of superuser for the database system, which is implemented as a directory tree of files containing individual databases. Each individual database has its own directory sub-tree.

PostgreSQL database server installation

The first step towards the installation of the PostgreSQL database server is its procurement. The sources for this server can be freely downloaded from <http://www.postgresql.org > which is the official site of the developers. As always, it is better to download the sources than the precompiled binaries. The source for this server consists of a massive 20,000 lines of code, so expect a proportionate time for compilation.

The downloaded file is obtained as a zipped file in .gz format. The file is to be first unzipped and then untared.

```
>gunzip postgresql-7.0.tar.gz
>tar -xvf postgresql-7.0.tar
```

The tar -xvf command is for unpacking the tar file. Suppose the downloaded file is stored in /postgres/tars. Now the downloaded file is of the format postgresql-7.0.tar.gz. Now after gunzip and the tar -xvf command, the files are unpacked in the same directory /postgres/tars in a directory within the present directory with name postgresql-7.0.

The files contained are listed below

```
-rw-r--r--    1 1005    wheel        1169   COPYRIGHT
-rw-r--r--    1 1005    wheel      108336   HISTORY
-rw-r--r--    1 1005    wheel       74632   INSTALL
-rw-r--r--    1 1005    wheel        1925   README
drwxr-xr-x  26 1005    wheel        1024   contrib
drwxr-xr-x   4  root    root         1024   doc
-rw-r--r-- 1    1005    wheel         742   register.txt
drwxr-xr-x  16 1005    wheel        1024   src
```

The src directory has all the configuration file like configure, which is an executable file. It also consists of files like config.guess, config.status, config.sub etc. These are all excutable files. There are other non-executable files and directories also. The first step towards installation is checking out the INSTALL file. This document contains a brief introduction to PosgreSQL and contains all the setup instructions.

Installation Steps

- Create the PostgreSQL superuser account. This is the user the server will run as. For production use you should create a separate, unprivileged account (postgres is commonly used). If you do not have root access or just want to play around, your own user account is enough. Running PostgreSQL as root, bin, or any other account with special access rights is a security risk; don't do it. The postmaster will in fact refuse to start as root. You need not do the building and installation itself under this account (although you can). You will be told when you need to login as the database superuser.
- Configure the source code for your system. It is this step at which you can specify your actual installation path for the build process and make choices about what gets installed. Change into the src subdirectory and type:

```
> ./configure
```

followed by any options you might want to give it. For a first installation you should be able to do fine without any. For a complete list of options, type:

```
> ./configure --help
```

Some of the more commonly used ones are:

--prefix=BASEDIR
Selects a different base directory for the installation of PostgreSQL. The default is /usr/local/pgsql.

--enable-locale
If you want to use locales.

--enable-multibyte
Allows the use of multibyte character encodings. This is primarily for languages like Japanese, Korean, or Chinese.

--with-perl
Builds the Perl interface and plperl extension language. Please note that the Perl interface needs to be installed into the usual place for Perl modules (typically under /usr/lib/perl), so you must have root access to perform the installation step. (It is often easiest to leave out --with-perl initially, and then build and install the Perl interface after completing the installation of PostgreSQL itself.)

--with-odbc

Builds the ODBC driver package.

--with-tcl
Builds interface libraries and programs requiring Tcl/Tk, including libpgtcl, pgtclsh, and pgtksh.

Compile the program. Type

```
> gmake
```

The compilation process can take anywhere from 10 minutes to an hour. Your mileage will most certainly vary. Remember to use GNU make. The last line displayed will hopefully be All of PostgreSQL be successfully made. Ready to install.

Install the PostgreSQL executable files and libraries. Type

```
> gmake install
```

You should do this step as the user that you want the installed executables to be owned by. This does not have to be the same as the database superuser; some people prefer to have the installed files be owned by root.

- If necessary, tell your system how to find the new shared libraries. How to do this varies between platforms. The most widely usable method is to set the environment variable.

```
LD_LIBRARY_PATH:
> LD_LIBRARY_PATH=/usr/local/pgsql/lib
> export LD_LIBRARY_PATH
```

on sh, ksh, bash, zsh or

```
> setenv LD_LIBRARY_PATH /usr/local/pgsql/lib
```

on csh or tcsh. You might want to put this into a shell startup file such as /etc/profile. On some systems the following is the preferred method, but you must have root access. Edit file /etc/ld.so.conf to add a line

```
/usr/local/pgsql/lib
```

Then run command /sbin/ldconfig.

If in doubt, refer to the manual pages of your system. If you later on get a message like psql: error in loading shared libraries libpq.so.2.1: cannot open shared object file: No such file or directory then the above was necessary.

Simply do this step.

- Create the database installation (the working data files).

To do this you must log in to your PostgreSQL superuser account. It will not work as root.

```
> mkdir /usr/local/pgsql/data
> chown postgres /usr/local/pgsql/data
> su - postgres
> /usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data
```

The -D option specifies the location where the data will be stored. You can use any path you want, it does not have to be under the installation directory. Just make sure that the superuser account can write to the directory (or create it, if it doesn't already exist) before starting initdb. (If you have already been doing the installation up to now as the PostgreSQL superuser, you may have to log in as root temporarily to create the data directory underneath a root-owned directory.)

- The previous step should have told you how to start up the database server. Do so now. The command should look something like

```
> /usr/local/pgsql/bin/postmaster -D /usr/local/pgsql/data
```

This will start the server in the foreground. To make it detach to the background, you can use the -S option, but then you won't see any log messages the server produces. A better way to put the server in the background is

```
> nohup /usr/local/pgsql/bin/postmaster -D /usr/local/pgsql/data/
                                          </dev/null>  >server.log 2>>1 &
```

- You probably want to install the man and HTML documentation.

Type

```
> cd /usr/src/pgsql/postgresql-7.0/doc
> gmake install
```

This will install files under /usr/local/pgsql/doc and /usr/local/pgsql/man. To enable your system to find the man documentation, you need to add a line like the following to a shell startup file:

```
> MANPATH=$MANPATH:/usr/local/pgsql/man
```

The documentation is also available in Postscript format. If you have a Postscript printer, or have your machine already set up to accept Postscript files using a print filter, then to print the User's Guide simply type

```
> cd /usr/local/pgsql/doc
> gunzip -c user.ps.tz | lpr
```

Here is how you might do it if you have Ghostscript on your system and are writing to a LaserJet printer.

```
> gunzip -c user.ps.gz \
| gs -sDEVICE=laserjet -r300 -q -dNOPAUSE -sOutputFile=- \ | lpr
```

Printer setups can vary wildly from system to system. If in doubt, consult your manuals or your local expert. The Administrator's Guide should probably be your first reading if you are completely new to PostgreSQL, as it contains information about how to set up database users and authentication.

- Usually, you will want to modify your computer so that it will automatically start the database server whenever it boots. This is not required; the PostgreSQL server can be run successfully from non-privileged accounts without root intervention. Different systems have different conventions for starting up daemons at boot time, so you are advised to familiarize yourself with them. Most systems have a file /etc/rc.local or /etc/rc.d/rc.local which is almost certainly no bad place to put such a command. Whatever you do, postmaster must be run by the PostgreSQL superuser (postgres) and not by root or any other user. Therefore you probably always want to form your command lines along the lines of su -c '...' postgres. It might be advisable to keep a log of the server output. To start the server that way try:

```
> nohup su -c 'postmaster -D /usr/local/pgsql/data >
server.log 2>&1' postgres &
```

This is all there is for the successful compilation and installation of the PostgreSQL server. Now we will follow up with a brief explanation of the compile time directives required.

If you look at the configuration stage, you will notice that there are some options required along with ./configure. Among all the options available, for our purpose the most suitable ones are --with-odbc and with-perl. This is because ODBC is an industry standard for database access. It stands for Open Data Base Connectivity. As the name suggests, it provides support for the remote connectivity of applications with the database. And perl is a scripting language used everywhere on the web. The use of this option enables the combination of our server with the interactivity of the Perl language.

Thus the typical command entered by us for configuration was :-

```
            ./configure --prefix=/usr1/postgreSQL/ --with-odbc
--with-perl
```

Using the Database

The first step towards the use of the PostgreSQL database server is the creation of a database. The database is created by the createdb command. The only pre-requisite for the use of this command is that the user must be authorised by the Postgres super user to create new databases in the current database system. If the user is unauthorised, an error message is flashed on the screen. Also the server must be up and running. The server is brought up by giving the following command at the command prompt (It is assumed that the installation is at /usr/local/pgsql and that the
db system has been created at /usr/local/pgsql/data) :-

```
> /usr/local/pgsql/bin/postmaster -D /usr/local/pgsql/data
```

To create a database,

```
> /usr/local/psql/bin/createdb test
```

This will initiate the server (back-end) and it will wait for queries at the standard unix port of 5432 (default). Additionally, TCP/IP support can be initiated by adding an option -i in the command above.

The second step is to select a front end which will interact with the server and show the results to the user. In other words, the front-end is required for user interaction with the server. The easiest to learn front-end is the pgsql interactive SQL monitor distributed with the PostgreSQL source. Now, to start working on your database, just fire up the pgsql SQL monitor by giving the following instruction: -

```
> psql test
```

This instruction will show a screen output similar to the one given below :-

```
Welcome to psql, the PostgreSQL interactive terminal.

Type: \copyright for distribution terms
      \h for help with SQL commands
      \? for help on internal slash commands
      \g to terminate with semicolon to execute query
      \q to quit

test=>
```

Now that both the back-end and the front-ends are set-up, all that remains is to test fire the installation by issuing some simple database creation commands. The most basic commands are referenced in the Appendix B of this manual.

The output shown above is that of the front-end. A very simple command for first time users is select current_user. This will return the name of the current user. It is essential to follow each command by a semicolon(;). The commands can spwan to multiple lines. To quit the interactive monitor simply enter \d.

The best guide to learning PostgreSQL is the accompanying documentation and its web-site, which houses some of the best references for learning SQL and the PostgreSQL server concepts.

Now that we have successfully deployed a database server for our services platform, all that remains is to integrate these resources and to initiate them. The next step is to setup a interactive scripting language which has the ability to integrate our web server files(pages) with the database so that we may commit tasks like the maintenance of user accounts, statistics etc.

The PHP Scripting Language

What is PHP ?

PHP is an HTML embedded scripting language. Its an acronym for PHP: Hypertext Preprocessor. Much of its syntax is borrowed from C, Java and Perl with a couple of PHP-unique features thrown in. The goal of the language is to allow web developers to write dynamically generated web-pages quickly. An example of PHP scripting is given below :

```
< html >
      <head>
            <title>Example</title>
      </head>
      <body>
            <?php echo "Hi, I'm a PHP script!"; ?>
      </body>
</html>
```

Notice how this is different from a CGI script written in other languages like Perl or C - instead of writing a program with lots of commands to output HTML, you write an HTML script with some embedded code to do something. The PHP code is enclosed in special start and end tags that alow you to jump into and out of PHP mode.

What distinguishes PHP from something like client-side JavaScript is that the code is executed on the server. If you were to have a script similar to the above on your server, the client would receive the results of running that script, with no way of determining what the underlying code may be.

What can PHP do ?

At the most basic level, PHP can do anything any other CGI program can do, such as collect form data, generate dynamic page content, or send and receive cookies.

Perhaps the strongest and most significant feature in PHP is its support for a wide range of databases. Writing a database enabled web-page is incredibly simple. Support for databases includes, but is not limited to Adabase D, dBase, Oracle, MySQL, PostgreSQL etc.

PHP also has support for talking to other services using the protocols such as IMAP, SNMP, NNTP, POP3, or even HTTP. We can also open raw network sockets and interact using other protocols.

Why PHP ?

The basic question that arises whenever any new programming paradigm arrives is the need for it. Before PHP made its inroads into the Internet world, languages like Perl and Java were already having a strong user base. They are still widely used on the Internet, although a recent estimate puts the number of sites running on PHP to about 1,000,000.

Simply said, PHP is a major substitute for other scripting languages like Perl. The first major advantage of using PHP is that it is embedded right into the HTML document, unlike in Perl where you have to write a script at some other executable location. Embedded means that now the server will take considerable shorter time to execute the code and display the results.

Secondly, PHP has seamless integration with a number of database systems. Whereas Java supports ODBC too, PHP has inbuilt functions simple enough to be used by a novice to create sites with exemplary results. PHP can also be integrated with XML and ASP, even though ASP is aptly described as a new technology rather than a scripting language (which is VBScript).

Installation

The first step as with all UNIX software is getting the sources for compilation. The PHP sources can easily be downloaded from < http://www.php.net > . This is the official web-site of the developers. It is advisable to download the sources rather than the precompiled binaries.

For compilation, all that is required is some basic UNIX skills, an ANSI C Compiler (like gcc) and a web server (we have already installed Apache).

Quick Installation Instructions

```
        $ gunzip -c apache_1.3.x.tar.gz | tar xf -
        $ cd apache_1.3.x
        $ ./configure
        $ cd ..

        $ gunzip -c php-4.0.x.tar.gz | tar xf -
        $ cd php-4.0.x
        $ ./configure --with-mysql --with-apache=../apache_1.3.x --enable-track-vars
        $ make
        $ make install

        $ cd ../apache_1.3.x
        $ ./configure --prefix=/www --activate-module=src/modules/php4/libphp4.a
            (The above line is correct!  Yes, we know libphp4.a does not exist at this
                                    stage.  It isn't supposed to.  It will be created.)
        $make
            (you should now have an httpd binary which you can copy to your Apache bin dir)

$cd ../php-4.0.x

$cp php.ini-dist /usr/local/lib/php.ini
            You can edit /usr/local/lib/php.ini file to set PHP options.
```

Edit your httpd.conf or srm.conf file and add:

```
AddType application/x-httpd-php .php
```

The preferred directory to do this installation is /usr/local. This is because all the programs lie in this directory, which serves as a repository.

Note : -

At this point, it is important to stress that the PHP module can be installed as a dynamic module for Apache. This means that there should be no need to recompile Apache. As this is always the preffered method, you should always try this. The support for dynamic modules was experimental in previous versions of Apache and the installation buggy. The same problem appeared before us, and we were unable to install the dynamic module, so we had to take the primitive route to installation, which was to build a static module and then include it in the Apache installation by recompiling the whole of Apache.

Details regarding the ./configure command

It is a regular feature in compiling UNIX programs that we have to configure the sources/installation file before actually beginning compilation. This is done to set up various machine specific options and to avoid the compilation of unnecessary options. Here too, the configuration is very important for successful installation.

At this stage, various choices have to be made, making the availability of a setup plan an absolute necessity. Given below are some of the configure options: -

The PHP configuration involves the choice of the database server and then the database environment to be used. The options for configuration are given below:-

(a) Apache module

This module is to build PHP as an apache module. To build PHP with this module, --with-apache=DIR option should be included while installation.

(b) fhttpd module

This module is used to build PHP as fhttpd. The option--with-fhttpd=DIR should be included during installation.

(c) CGI version

The default is to build PHP as a CGI version. The PHP is built with this option if no other option is given during compilation.

(d) Database Support Option

The database options are listed below.

(1) Adabas D

```
--with-adabas=DIR
```

(2) dBase

```
--with-dbase=DIR
```

(3) filePro
    --with-filepro=DIR

(4) mSQL
    --with-msql=DIR

(5) MySQL
    --with-mysql=DIR

(6) iODBC
    --with-iodbc=DIR

(7) OpenLink ODBC
    --with-openlink=DIR

(8) Oracle
    --with-oracle=DIR

(9) PostgreSQL
    --with-pgsql=DIR

(10) Solid
    --with-solid=DIR

(11) Sybase
    --with-sysbase=DIR

(12) Sysbase-CT
    --with-sysbase-ct=DIR

(13) Velocis
    --with-velocis=DIR

(14) A custom ODBC library
    --with-custom-odbc=DIR

(15) Unified ODBC
    --disable-unified-odbc
    This disables the unified ODBC module, which is a common interface to all the databases with ODBC-based interface. This option is only applicable if one of the following option is used: --with-iodbc, --with-solid, --with-adabas, --with-velocis, --with-custom-odbc.

(16) LDAP
    --with-ldap=DIR

(e) Other configure options

--with-mcrypt=DIR

--enable-sysvsem

--enable-sysvshm

--with-xml

--enable-maintainer-mode
    Turns on extra dependencies and compiler warnings.

--with-system-regex
    Uses the system`s regular expression library rather than the bundled one.

--with-config-file-path=DIR
    The path used to look for the configuration file when PHP starts up.

--with-exec-dir=DIR
    Only allows the running of executables in DIR when in safe mode.

--enable-debug

--enable-safe-mode

--enable-track-vars
    Makes PHP keep track of where GET/POST/cookie variables come from in the arrays. This option only sets the default, it may be enabled or disabled in track_vars directive in the configuration file later.

--enable-magic-quotes

--enable-debugger

--enable-discard-path
    If this option is enabled, the PHP CGI binary can safely be placed outside of the web tree and peoples will not be able to circumvert .htaccess security.

--enable-bcmath
    This option enables bc style arbitrary math functions.

--enable-force-cgi-redirect

This option enables the security check for the internal server redirects. This option should be used if running the CGI version with apache. Not enabling this option disables the check and enables bypassing httpd security and authentication settings.

--disable-short-tags

This disables the short form <? ?> PHP tags. The short form must be disabled if it is desired to use PHP with XML.

--enable-url-includes

This option makes it possible to run code on other HTTP or FTP servers directly from PHP with include().

--disable-syntax-hl

This option turns off syntax highlighting.

The configuration is done by the ./configure file which is obtained by unpacking the PHP source code. The other option which is to be supplied while configuration is the directory in which all the configuration files are to be stored. This is given by the prefix command. A sample command which can be given for configuration is shown below:

```
./configure --prefix=/usr1/iPHP/ --with-pqsql=/usr1/local/pgsql
```

After the successful completion of the above step, a new static module with a .o extension will be created in the Apache modules directory, which has to be activated during the recompilation of the Apache sources.

The PHP interpreter is thus installed and ready for work.

Appendix A

Sample httpd configuration file

```
##
## httpd.conf -- Apache HTTP server configuration file
##

# After this file is processed, the server will look for and process
# /usr1/httpd/conf/srm.conf and then /usr1/httpd/conf/access.conf
# unless you have overridden these with ResourceConfig and/or
# AccessConfig directives here.
#
# The configuration directives are grouped into three basic sections:
#  1. Directives that control the operation of the Apache server process as a
#     whole (the 'global environment').
#  2. Directives that define the parameters of the 'main' or 'default' server,
#     which responds to requests that aren't handled by a virtual host.
#     These directives also provide default values for the settings
#     of all virtual hosts.
#  3. Settings for virtual hosts, which allow Web requests to be sent to
#     different IP addresses or hostnames and have them handled by the
#     same Apache server process.

### Section 1: Global Environment
#
# ServerType is either inetd, or standalone.  Inetd mode is only supported on
# Unix platforms.
#
ServerType standalone

#
# ServerRoot: The top of the directory tree under which the server's
# configuration, error, and log files are kept.
#
ServerRoot "/usr1/httpd"

#
# The LockFile directive sets the path to the lockfile
#
#LockFile /usr1/httpd/logs/httpd.lock

#
# PidFile: The file in which the server should record its process
# identification number when it starts.
#
```

```
PidFile /usr1/httpd/logs/httpd.pid

#
# ScoreBoardFile: File used to store internal server process information.
# Not all architectures require this.  But if yours does (you'll know because
# this file will be  created when you run Apache) then you *must* ensure that
# no two invocations of Apache share the same scoreboard file.
#
ScoreBoardFile /usr1/httpd/logs/httpd.scoreboard

#
# In the standard configuration, the server will process this file,
# srm.conf, and access.conf in that order.  The latter two files are
# now distributed empty, as it is recommended that all directives
# be kept in a single file for simplicity.  The commented-out values
# below are the built-in defaults.  You can have the server ignore
# these files altogether by using "/dev/null" (for Unix) or
# "nul" (for Win32) for the arguments to the directives.
#
#ResourceConfig conf/srm.conf
#AccessConfig conf/access.conf

#
# Timeout: The number of seconds before receives and sends time out.
#
Timeout 300

#
# KeepAlive: Whether or not to allow persistent connections (more than
# one request per connection). Set to "Off" to deactivate.
#
KeepAlive On

#
# MaxKeepAliveRequests: The maximum number of requests to allow
# during a persistent connection. Set to 0 to allow an unlimited amount.
# We recommend you leave this number high, for maximum performance.
#
MaxKeepAliveRequests 100

#
# KeepAliveTimeout: Number of seconds to wait for the next request from the
# same client on the same connection.
#
KeepAliveTimeout 15
```

```
#
# It does this by periodically checking how many servers are waiting
# for a request.  If there are fewer than MinSpareServers, it creates
# a new spare.  If there are more than MaxSpareServers, some of the
# spares die off.  The default values are probably OK for most sites.
#
MinSpareServers 5
MaxSpareServers 10

#
# Number of servers to start initially --- should be a reasonable ballpark
# figure.
#
StartServers 1

#
# Limit on total number of servers running, i.e., limit on the number
# of clients who can simultaneously connect --- if this limit is ever
# reached, clients will be LOCKED OUT, so it should NOT BE SET TOO LOW.
# It is intended mainly as a brake to keep a runaway server from taking
# the system with it as it spirals down...
#
MaxClients 150

#
# MaxRequestsPerChild: the number of requests each child process is
# allowed to process before the child dies.  The child will exit so
# as to avoid problems after prolonged use when Apache (and maybe the
# libraries it uses) leak memory or other resources.  On most systems, this
# isn't really needed, but a few (such as Solaris) do have notable leaks
# in the libraries. For these platforms, set to something like 10000
# or so; a setting of 0 means unlimited.
#
MaxRequestsPerChild 100

#
# Listen: Allows you to bind Apache to specific IP addresses and/or
# ports, in addition to the default. See also the <VirtualHost>
# directive.
#
#Listen 3000
#Listen 12.34.56.78:80

#
# BindAddress: You can support virtual hosts with this option. This directive
# is used to tell the server which IP address to listen to. It can either
# contain "*", an IP address, or a fully qualified Internet domain name.
# See also the <VirtualHost> and Listen directives.
#
#BindAddress *

#
# Dynamic Shared Object (DSO) Support
#
# To be able to use the functionality of a module which was built as a DSO you
# have to place corresponding `LoadModule' lines at this location so the
# directives contained in it are actually available _before_ they are used.
# Please read the file README.DSO in the Apache 1.3 distribution for more
# details about the DSO mechanism and run `httpd -l' for the list of already
# built-in (statically linked and thus always available) modules in your httpd
# binary.
#
# Note: The order is which modules are loaded is important.  Don't change
# the order below without expert advice.
#
# Example:
# LoadModule foo_module libexec/mod_foo.so
#LoadModule php_module /usr/lib/apache/mod_php.so

#
# ExtendedStatus controls whether Apache will generate "full" status
# information (ExtendedStatus On) or just basic information (ExtendedStatus
# Off) when the "server-status" handler is called. The default is Off.
#
#ExtendedStatus On

### Section 2: 'Main' server configuration
#
# Port: The port to which the standalone server listens. For
# ports < 1023, you will need httpd to be run as root initially.
#
Port 80

# User/Group: The name (or #number) of the user/group to run httpd as.
#  . On SCO (ODT 3) use "User nouser" and "Group nogroup".
#  . On HPUX you may not be able to use shared memory as nobody, and the
#    suggested workaround is to create a user www and use that user.
#
User nobody
Group nobody
```

```
#
# ServerAdmin: Your address, where problems with the server should be
# e-mailed.  This address appears on some server-generated pages, such
# as error documents.
#
ServerAdmin root@sws05.ee.iitk.ac.in

#
# ServerName allows you to set a host name which is sent back to clients for
# your server if it's different than the one the program would get (i.e., use
# "www" instead of the host's real name).
#
# If your host doesn't have a registered DNS name, enter its IP address here.
# You will have to access it by its address (e.g., http://123.45.67.89/)
# anyway, and this will make redirections work in a sensible way.
#
ServerName 172.31.44.14

#
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.
#
DocumentRoot "/usr1/httpd/htdocs"

#
# Each directory to which Apache has access, can be configured with respect
# to which services and features are allowed and/or disabled in that
# directory (and its subdirectories).
#
# First, we configure the "default" to be a very restrictive set of
# permissions.
#
<Directory />
   Options FollowSymLinks
   AllowOverride None
</Directory>

#
# This should be changed to whatever you set DocumentRoot to.
#
<Directory "/usr1/httpd/htdocs">

#
# This may also be "None", "All", or any combination of "Indexes",
# "Includes", "FollowSymLinks", "ExecCGI", or "MultiViews".
```

```
#
# Note that "MultiViews" must be named *explicitly* --- "Options All"
# doesn't give it to you.
#
    Options Indexes FollowSymLinks

#
# This controls which options the .htaccess files in directories can
# override. Can also be "All", or any combination of "Options", "FileInfo",
# "AuthConfig", and "Limit"
#
    AllowOverride None

#
# Controls who can get stuff from this server.
#
    Order allow,deny
     Allow from all
    </Directory>

#
# UserDir: The name of the directory which is appended onto a user's home
# directory if a ~user request is received.
#
UserDir public_html

#
# Control access to UserDir directories.  The following is an example
# for a site where these directories are restricted to read-only.
#
#<Directory /home/*/public_html>
#    AllowOverride FileInfo AuthConfig Limit
#    Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
#    <Limit GET POST OPTIONS PROPFIND>
#        Order allow,deny
#        Require valid-user
#    </Limit>
#    <Limit PUT DELETE PATCH PROPPATCH MKCOL COPY MOVE LOCK UNLOCK>
#        Order deny,allow
#        Deny from all
#    </Limit>
#</Directory>

#
# DirectoryIndex: Name of the file or files to use as a pre-written HTML
# directory index.  Separate multiple entries with spaces.
```

```
#
DirectoryIndex index.html


# AccessFileName: The name of the file to look for in each directory
# for access control information.
#
AccessFileName .htaccess

#
# The following lines prevent .htaccess files from being viewed by
# Web clients.  Since .htaccess files often contain authorization
# information, access is disallowed for security reasons.  Comment
# these lines out if you want Web visitors to see the contents of
# .htaccess files.  If you change the AccessFileName directive above,
# be sure to make the corresponding changes here.
#
<Files ~ "^\.ht">
   Order allow,deny
   Deny from all
</Files>

#
# CacheNegotiatedDocs: By default, Apache sends "Pragma: no-cache" with each
# document that was negotiated on the basis of content. This asks proxy
# servers not to cache the document. Uncommenting the following line disables
# this behavior, and proxies will be allowed to cache the documents.
#
CacheNegotiatedDocs

#
# UseCanonicalName:  (new for 1.3)  With this setting turned on, whenever
# Apache needs to construct a self-referencing URL (a URL that refers back
# to the server the response is coming from) it will use ServerName and
# Port to form a "canonical" name.  With this setting off, Apache will
# use the hostname:port that the client supplied, when possible.  This
# also affects SERVER_NAME and SERVER_PORT in CGI scripts.
#
UseCanonicalName On

#
# TypesConfig describes where the mime.types file (or equivalent) is
# to be found.
#
TypesConfig /usr1/httpd/conf/mime.types
```

```
#
# DefaultType is the default MIME type the server will use for a document
# if it cannot otherwise determine one, such as from filename extensions.
# If your server contains mostly text or HTML documents, "text/plain" is
# a good value.  If most of your content is binary, such as applications
# or images, you may want to use "application/octet-stream" instead to

# keep browsers from trying to display binary files as though they are
# text.
#
DefaultType text/plain

#
# The mod_mime_magic module allows the server to use various hints from the
# contents of the file itself to determine its type.  The MIMEMagicFile
# directive tells the module where the hint definitions are located.
# mod_mime_magic is not part of the default server (you have to add
# it yourself with a LoadModule [see the DSO paragraph in the 'Global
# Environment' section], or recompile the server and include mod_mime_magic
# as part of the configuration), so it's enclosed in an <IfModule> container.
# This means that the MIMEMagicFile directive will only be processed if the
# module is part of the server.
#
<IfModule mod_mime_magic.c>
   MIMEMagicFile /usr1/httpd/conf/magic
</IfModule>

#
# HostnameLookups: Log the names of clients or just their IP addresses
# The default is off . Enabling it means thateach client request will result in
#AT LEAST one lookup request to the nameserver.
#
HostnameLookups Off

#
# ErrorLog: The location of the error log file.
# If you do not specify an ErrorLog directive within a <VirtualHost>
# container, error messages relating to that virtual host will be
# logged here.
#
ErrorLog /usr1/httpd/logs/error_log

#
# LogLevel: Control the number of messages logged to the error_log.
# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.
```

```
#
LogLevel warn

#
# The following directives define some format nicknames for use with
# a CustomLog directive (see below).
#
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

#
# The location and format of the access logfile (Common Logfile Format).
#
CustomLog /usr1/httpd/logs/access_log common

#
# If you would like to have agent and referer logfiles, uncomment the
# following directives.
#
#CustomLog /usr1/httpd/logs/referer_log referer
#CustomLog /usr1/httpd/logs/agent_log agent

#
# If you prefer a single logfile with access, agent, and referer information
# (Combined Logfile Format) you can use the following directive.
#
#CustomLog /usr1/httpd/logs/access_log combined

#
# Optionally add a line containing the server version and virtual host
# name to server-generated pages (error documents, FTP directory listings,
# mod_status and mod_info output etc., but not CGI generated documents).
# Set to "EMail" to also include a mailto: link to the ServerAdmin.
# Set to one of:  On | Off | EMail
#
ServerSignature On

#
# Aliases: Add here as many aliases as you need (with no limit). The format is
# Alias fakename realname
#
Alias /icons/ "/usr1/httpd/icons/"

<Directory "/usr1/httpd/icons">
```

```
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

#
# ScriptAlias: This controls which directories contain server scripts.
# ScriptAliases are essentially the same as Aliases, except that
# documents in the realname directory are treated as applications and
# run by the server when requested rather than as documents sent to the client.
#
ScriptAlias /cgi-bin/ "/usr1/httpd/cgi-bin/"

#
# "/usr1/httpd/cgi-bin" should be changed to whatever your ScriptAliased
# CGI directory exists, if you have that configured.
#
<Directory "/usr1/httpd/cgi-bin">
    AllowOverride None
    Options None
   # Order allow,deny
   # Allow from all
    AuthType Basic
    AuthName "Restricted Directory"
    AuthDBUserFile /usr1/httpd/bin/users
    Require valid-user
</Directory>

#
# Redirect allows you to tell clients about documents which used to exist in
# your server's namespace, but do not anymore. This allows you to tell the
# clients where to look for the relocated document.
# Format: Redirect old-URI new-URL
#

#
# Directives controlling the display of server-generated directory listings.
#

#
# FancyIndexing is whether you want fancy directory indexing or standard
#
IndexOptions FancyIndexing

#
```

```
# AddIcon* directives tell the server which icon to show for different
# files or filename extensions.  These are only displayed for
# FancyIndexed directories.
#
AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip

AddIconByType (TXT,/icons/text.gif) text/*
AddIconByType (IMG,/icons/image2.gif) image/*
AddIconByType (SND,/icons/sound2.gif) audio/*
AddIconByType (VID,/icons/movie.gif) video/*

AddIcon /icons/binary.gif .bin .exe
AddIcon /icons/binhex.gif .hqx
AddIcon /icons/tar.gif .tar
AddIcon /icons/world2.gif .wrl .wrl.gz .vrml .vrm .iv
AddIcon /icons/compressed.gif .Z .z .tgz .gz .zip
AddIcon /icons/a.gif .ps .ai .eps
AddIcon /icons/layout.gif .html .shtml .htm .pdf
AddIcon /icons/text.gif .txt
AddIcon /icons/c.gif .c
AddIcon /icons/p.gif .pl .py
AddIcon /icons/f.gif .for
AddIcon /icons/dvi.gif .dvi
AddIcon /icons/uuencoded.gif .uu
AddIcon /icons/script.gif .conf .sh .shar .csh .ksh .tcl
AddIcon /icons/tex.gif .tex
AddIcon /icons/bomb.gif core

AddIcon /icons/back.gif ..
AddIcon /icons/hand.right.gif README
AddIcon /icons/folder.gif ^^DIRECTORY^^
AddIcon /icons/blank.gif ^^BLANKICON^^

#
# DefaultIcon is which icon to show for files which do not have an icon
# explicitly set.
#
DefaultIcon /icons/unknown.gif

#
# AddDescription allows you to place a short description after a file in
# server-generated indexes.  These are only displayed for FancyIndexed
# directories.
# Format: AddDescription "description" filename
#
#AddDescription "GZIP compressed document" .gz
```

```
#AddDescription "tar archive" .tar
#AddDescription "GZIP compressed tar archive" .tgz

#
# ReadmeName is the name of the README file the server will look for by
# default, and append to directory listings.
#
# HeaderName is the name of a file which should be prepended to
# directory indexes.
#
# The server will first look for name.html and include it if found.
# If name.html doesn't exist, the server will then look for name.txt
# and include it as plaintext if found.
#
ReadmeName README
HeaderName HEADER

#
# IndexIgnore is a set of filenames which directory indexing should ignore
# and not include in the listing.  Shell-style wildcarding is permitted.
#
IndexIgnore .??* *~ *# HEADER* README* RCS CVS *,v *,t

#
# AddEncoding allows you to have certain browsers (Mosaic/X 2.1+) uncompress
# information on the fly. Note: Not all browsers support this.
# Despite the name similarity, the following Add* directives have nothing
# to do with the FancyIndexing customization directives above.
#
AddEncoding x-compress Z
AddEncoding x-gzip gz tgz

#
# AddLanguage allows you to specify the language of a document. You can
# then use content negotiation to give a browser a file in a language
# it can understand.  Note that the suffix does not have to be the same
# as the language keyword --- those with documents in Polish (whose
# net-standard language code is pl) may wish to use "AddLanguage pl .po"
# to avoid the ambiguity with the common suffix for perl scripts.
#
AddLanguage en .en
AddLanguage fr .fr
AddLanguage de .de
AddLanguage da .da
AddLanguage el .el
AddLanguage it .it
```

```
#
# LanguagePriority allows you to give precedence to some languages
# in case of a tie during content negotiation.
# Just list the languages in decreasing order of preference.
#
LanguagePriority en fr de

#
# AddType allows you to tweak mime.types without actually editing it, or to
# make certain files to be certain types.
#
# For example, the PHP3 module (not part of the Apache distribution - see
# http://www.php.net) will typically use:
#
AddType application/x-httpd-php .php
#AddType application/x-httpd-php3-source .phps

AddType application/x-tar .tgz

#
# AddHandler allows you to map certain file extensions to "handlers",
# actions unrelated to filetype. These can be either built into the server
# or added with the Action command (see below)
#
# If you want to use server side includes, or CGI outside
# ScriptAliased directories, uncomment the following lines.
#
# To use CGI scripts:
#
#AddHandler cgi-script .cgi

#
# To use server-parsed HTML files
#
#AddType text/html .shtml
#AddHandler server-parsed .shtml

#
# Uncomment the following line to enable Apache's send-asis HTTP file
# feature
#
#AddHandler send-as-is asis

#
# If you wish to use server-parsed imagemap files, use
```

```
#
#AddHandler imap-file map

#
# To enable type maps, you might want to use
#
#AddHandler type-map var

#
# Action lets you define media types that will execute a script whenever
# a matching file is called. This eliminates the need for repeated URL
# pathnames for oft-used CGI file processors.
# Format: Action media/type /cgi-script/location
# Format: Action handler-name /cgi-script/location
#

#
# MetaDir: specifies the name of the directory in which Apache can find
# meta information files. These files contain additional HTTP headers
# to include when sending the document
#
#MetaDir .web

#
# MetaSuffix: specifies the file name suffix for the file containing the
# meta information.
#
#MetaSuffix .meta

#
# Customizable error response (Apache style)
#  these come in three flavors
#
#    1) plain text
#ErrorDocument 500 "The server made a boo boo.
#  n.b.  the (") marks it as text, it does not get output
#
#    2) local redirects
#ErrorDocument 404 /missing.html
#  to redirect to local URL /missing.html
#ErrorDocument 404 /cgi-bin/missing_handler.pl
#  N.B.: You can redirect to a script or a document using server-side-includes.
#
#    3) external redirects
#ErrorDocument 402 http://some.other_server.com/subscription_info.html
#
```

```
# The following directives modify normal HTTP response behavior.
# The first directive disables keepalive for Netscape 2.x and browsers that
# spoof it. There are known problems with these browser implementations.
# The second directive is for Microsoft Internet Explorer 4.0b2
# which has a broken HTTP/1.1 implementation and does not properly
# support keepalive when it is used on 301 or 302 (redirect) responses.
#
BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0 force-response-1.0

#
# The following directive disables HTTP/1.1 responses to browsers which
# are in violation of the HTTP/1.0 spec by not being able to grok a
# basic 1.1 response.
#
BrowserMatch "RealPlayer 4\.0" force-response-1.0
BrowserMatch "Java/1\.0" force-response-1.0
BrowserMatch "JDK/1\.0" force-response-1.0

#
# Allow server status reports, with the URL of http://servername/server-status
# Change the ".your_domain.com" to match your domain to enable.
#
<Location /server-status>
   SetHandler server-status
   Order deny,allow
   Deny from all
   Allow from all
</Location>

#
# Allow remote server configuration reports, with the URL of
#  http://servername/server-info (requires that mod_info.c be loaded).
# Change the ".your_domain.com" to match your domain to enable.
#
<Location /server-info>
   SetHandler server-info
   Order deny,allow
   Deny from all
   Allow from all
</Location>

#
# There have been reports of people trying to abuse an old bug from pre-1.1
# days.  This bug involved a CGI script distributed as a part of Apache.
# By uncommenting these lines you can redirect these attacks to a logging
# script on phf.apache.org.  Or, you can record them yourself, using the script
# support/phf_abuse_log.cgi.
#
#<Location /cgi-bin/phf*>
#    Deny from all
#    ErrorDocument 403 http://phf.apache.org/phf_abuse_log.cgi
#</Location>

#
# Proxy Server directives. Uncomment the following lines to
# enable the proxy server:
#
#ProxyRequests On
<IfModule mod_proxy.c>
ProxyRequests On
ProxyRemote http://www.hotmail.com http://www.iitk.ac.in
#
<Directory proxy:/usr1/httpd/proxy>
  AllowOverride None
  Options None
 #  Order deny,allow
 #  Deny from none
 #  Allow from all
   AuthType Basic
   AuthName "Restricted Access"
   AuthDBUserFile /usr1/httpd/bin/users
   Require valid-user
</Directory>

#
# Enable/disable the handling of HTTP/1.1 "Via:" headers.
# ("Full" adds the server version; "Block" removes all outgoing Via: headers)
# Set to one of: Off | On | Full | Block
#
ProxyVia On

#
# To enable the cache as well, edit and uncomment the following lines:
# (no cacheing without CacheRoot)
#
CacheRoot "/usr1/httpd/proxy"
CacheSize 5
CacheGcInterval 4
CacheMaxExpire 24
CacheLastModifiedFactor 0.1
CacheDefaultExpire 1
```

```
#NoCache a_domain.com another_domain.edu joes.garage_sale.com
# NoCache 172.31.44.14
</IfModule>
# End of proxy directives.

### Section 3: Virtual Hosts
#
# VirtualHost: If you want to maintain multiple domains/hostnames on your
# machine you can setup VirtualHost containers for them.
# Please see the documentation at <URL:http://www.apache.org/docs/vhosts/>
# for further details before you try to setup virtual hosts.
# You may use the command line option '-S' to verify your virtual host
# configuration.

#
# If you want to use name-based virtual hosts you need to define at
# least one IP address (and port number) for them.
#
#NameVirtualHost 12.34.56.78:80
#NameVirtualHost 12.34.56.78

#
# VirtualHost example:
# Almost any Apache directive may go into a VirtualHost container.
#
#<VirtualHost ip.address.of.host.some_domain.com>
#    ServerAdmin webmaster@host.some_domain.com
#    DocumentRoot /www/docs/host.some_domain.com
#    ServerName host.some_domain.com
#    ErrorLog logs/host.some_domain.com-error_log
#    CustomLog logs/host.some_domain.com-access_log common
#</VirtualHost>

#<VirtualHost _default_:*>
#</VirtualHost>
```

Appendix B

Simple SQL Queries

(a) CREATE TABLE [database name]

This commands is used for creating the tables used in database. In this command the field which are to be fed in the database are supplied such as for personal information database, it may be firstname, lastname, age etc which can be supplied to the database. This is the first step of creating a database in which fields are decided.

(b) INSERT INTO [database name] VALUES

This command is used for inserting the data according to the fields. With this command, data in the database can be modified by inserting new data in it.

(c) SELECT * FROM [database name]

This command shows the contents of the database. The contents are first stored by insert command. The * can be any of the desired field from the database. If * is given in the command then it will show all the contents of the database.

(d) SELECT * FROM [database name] WHERE [field]=[contents]

This command shows only a specified field of the database with a certain condition. It is basically a command for database query. The other options may be <,> or <> that is less then, greater than and not equal to.

(e) DELETE FROM [database name] WHERE [field]=[contents]

This command is used to delete data from a certain row. The greater than, less than and not equal to options are also present here. This command basically deletes the data of the field with matching contents.

(f) UPDATE [database name] SET [field]=[change] WHERE [field]=[content]

This command updates or changes a specified data where it finds a match with the field. This command is used to modify previously stored data.

(g) SELECT * FROM [database name] ORDER BY [field]

This command sorts and orders the database query according to the given field. If the field is of characters as in the case of state or name, then the query is ordered alphabetically.

(h) SELECT * FROM [database name] WHERE [field] IS NULL

This commands sorts out the database entries with some specified field left vacant either by mistake or intentionally. This helps in finding the incorrect entries.

(i) SELECT * FROM [database name] WHERE [field]=[content] AND [other content]

This command is basically for conditional search with two or more quantities at the same time. The AND command is in the case in which the query is shown when both the expression matches with the query. OR can also be used which will show the query when either or both the expression matches with the database.

Like Comparison

(1) SELECT * FROM [databasename] WHERE [field] BETWEEN [value] AND [value]

This command basically chosses the data stored in the database with the value of the field in between some limit.

(2) SELECT * FROM [databasename] WHERE [field] LIKE '[string]%'

This is also a comparison command in which query is all the names of the field beginning with a particular string. For example if the string is D, then it will search the database for all the field names starting with D.

Comparison	Operation
begins with D	LIKE 'D%'
contains a D	LIKE '%D%'
has D in second position	LIKE '_D%'
begins with D and contains e	LIKE 'D%e%'
begins with D, contains e, f	LIKE "D%e%f%'
begins with non-D	NOT LIKE 'D%'

Regular Expression

Comparison	Operator
regular expression	~
regular expression-case insensitive	~*
not equal to regular expression	!~
not equal to regular expression,case insensitive	!~*

Test	Special Characters
start	^
end	$
any single character	.
set of characters	[ccc]
set of characters not equal	[^ccc]
range of characters	[c-c]
range of characters not equal	[^c-c]
zero or one of the previous character	?
zero or multiple of previous characters	*
one or multiple of previous character	+
OR operator	|

Regular expression operators

Test	Operation
begins with D	~'^D'
contains D	~'D'
D in second position	~'^.D'
begins with D and contains e	~'D.*e'
begins with D, contains e, and then f	~'D.*e.*f'
contains A, B, C or D	~'[A-D]' or ~'[ABCD]'
contains A or a	~*'a' or ~'[Aa]'
does not contain D	!~'D'
does not begin with D	!~'^D' or ~'^[^D]'
begins with D, with one optional space	~'^?D'
begins with D, with optional spaces	~'^*D'
begins with D, with at least one space	~'^+D'
ends with G, with optional spaces	~'G*$'

Regular expressions special characters

Regular expresiion examples

Case Clause

(1) SELECT [field1],[field2] CASE WHEN [field2]>=[value] THEN '[strin g]' Else '[string]' END

This command is for deciding which of the data belongs to one class and which ones belong to the others according to certain criteria given in the command.

Functions and operators

There are large nember of functions available in POSTGRESQL.
Operators differ from functions in following ways:

- Operators are symbols, not names
- Always take two arguments
- Arguments appear to the left and right of the operator symbol

The standard arithmetic operators are: addition (+), subtraction(-), multiplication (*), division (/), modulo/remainder (%), and exponentiation (^) according to the precedence i.e. the exponentiation is done first and so on.

Set Show and Reset

Various parameters can be controlled by SET. DATESTYLE controls the appearence of dates. \df command at database prompt shows all the functions. \df int shows all int functions. TIMEZONE controls the timezone. SHOW DATESTYLE command may be used for viewing the existing datestyle.

SQL Aggregates

Aggregate	Function
COUNT(*)	count of rows
SUM(colname)	column total
MAX(colname)	column maximum
MIN(colname)	column minimum
AVG(colname)	column average

Aggregates

(1) SELECT [field], COUNT(*) FROM [databasename] GROUP BY [field]

This command counts number of rows of the field and sorts or groups the field in alphabetical order.

(2) SELECT [field], COUNT(*) FROM [databasename] GROUP BY [field] HAVING COUNT(*)>1

This command counts the number of rows of the field and shows only those data in the database query, which has more than one data with same field.

Joining Tables

In some records, there is requirement of storing variety of data in different fields. It becomes very difficult to store such a database with large number of fields in a single database. In that case, multiple tables are created and they are later joined. The advantages of using multiple tables are following:

- Easier data modification
- Easier data lookup
- Data stored in only one space
- Less storage space required.

```
(1) SELECT [database1].[field1]                --returns
   FROM [table1], [table2]                   --tables involved
   WHERE [database1].[field2]=[database2].[field2] AND     --join
   [database2].[field3]=[value]              --restriction
```

Sequences

Sequences are named counters created by users. After creation, the sequence can be assigned to a table as a column default. Using sequences, unique numbers can be automatically assigned during INSERT. The advantage of using sequences is that there is no gaps in numeric assignment. These are ideal as user-visible identification numbers.

Function	Action
nextval('name')	Returns the next available sequence number, and updates the counter
currval('name')	Returns the sequence number from the previous nextval() call
setval('name', newval)	Sets the sequence number counter to the specific value.

Sequence number access functions

Appendix C

Sample CGI script

```
#!/usr/bin/perl
use CGI qw (:standard);
use CGI::Carp qw(fatalsToBrowser);
#################################################
###              Set Configurations Below
#################################################

### Specify your database fields in the array below
@fields=("Roll_No","Category","Rank");

### Set URL and email fields to be hyperlinks
### The number used corresponds to its placement in the array above
### Remeber to start your count at zero
### Leave the variables empty to remove the hyperlink
$index_of_url = "8";
$index_of_email = "5";

### Specify the required fields below. This may be left empty
@required = ("Roll_No","Rank");

### Specify the filtered words below.  This may be left empty
@filter = ("word1","word2");

$database = "data.txt";                        # Data file name and path
$script_name = "dbspace.cgi";                  # This script name
$db_name = "IETK Result DB";                   # The database name
$table_width = "90%";                            # The display width of the entire table
$record_width = "80%";                         # The display width of the database records
$max_rows_returned = "10";             # Max number of rows displayed per page
$filter_html_tags = "yes";                     # Removes HTML tags from record entries
$user_file   = "users.txt";                    # User file name and path
$check_user_duplicates = "no";                 # Checks for duplicate user names in
#user file
@extra_user_fields = ("Email");                # Additional info needed when registering users
$use_external_html = "no";                     # Set to "yes" to use your own html template
$external_html_header = "header.html";           # File name and path to html template header
$external_html_footer = "footer.html";         # File name and path to html template footer
$authenticate = "yes";                 # Prompts users for a user name and password
$register         = "no";                      # Allows new user registration
$record_ownership = "no";              # Users can modify/delete only their
#records
```

```
$admin_user_name = "admin";                    # This is the default Administrator user name
$default_user_permission_search = "yes";       # Gives newly registered users Search permission
$default_user_permission_add = "no";           # Gives newly registered users
#Add permission
$default_user_permission_modify = "no";          # Gives newly registered users Modify
permission
$default_user_permission_delete = "no";          # Gives newly registered users Delete permission

##################################################
###          Change Nothing Below This Line
##################################################


############## Global Viariables ##################

$q = new CGI;
$field_count = @fields;
$colspan = $field_count+1;
$EXCLUSIVE = 2;
$UNLOCK    = 8;
$user = $q->param(user);
$user   = $q->cookie(user) if($user eq "");
$user_search = $q->param(user_search);
$user_search = "yes" if($authenticate eq "no");
$user_add = $q->param(user_add);
$user_add = "yes" if($authenticate eq "no");
$user_modify = $q->param(user_modify);
$user_modify = "yes" if($authenticate eq "no");
$user_delete = $q->param(user_delete);
$user_delete = "yes" if($authenticate eq "no");
$selected_user = $q->param(selected_user);
$admin_add = $q->param(admin_add);
$sort  = $q->param(sort_on);
$search_for   = $q->param(search_for);
$search_field = $q->param(search_field);
$action       = $q->param(action);
$display      = $q->param(display);
@keys         = $q->param(key);
$key          = $q->param(key);
$key_matches  = @keys;
$search_field = "all" if($search_field eq "");
$search_for   = '.'   if ($search_for eq "");
$action = $q->param(which_search) if($action =~ /^View/i);

################# Main Logic ##################
```

```
if($action =~ /add administrator/i){ &add_user; exit;}
if ($authenticate eq "yes"){ unless (-e $user_file){ $title="Set Administrative Password";
&create_admin; } }
if($register eq "yes" && $action =~ /register/i){ &register; exit;}
if($action =~ /add user/i){ &add_user; exit;}
if($action =~ /submit login/i){ &authenticate; exit;}
if ($authenticate eq "yes" && $user eq ""){ &login; exit;}
if($action =~ /logout/i){ &logout; exit;}

if($action =~ /add this record/i){ &add_record; $message="Record Added"; $title="Main
Menu"; &print_default($title, $message);}
elsif($action =~ /add/i){ &print_add_screen;}
elsif($action =~ /modify record/i){ $title="Modification Error"; $error_message="You forgot to
select a record to modify!"; &access_problem if($key < 1); &search_db($key);
&print_modify_page;}
elsif($action =~ /modify this record/i){ $title="Modify Record"; &delete_records;
$key=$keys[0]; &add_record; $message="Record Modified"; $title="Main Menu";
&print_default($title, $message);}
elsif($action =~ /delete record/i){ $title="Delete Record"; $error_message="You forgot to select
a record to delete!"; &access_problem if($key < 1); &delete_records; $message="Record
Deleted"; $title="Main Menu"; &print_default($title, $message);}
elsif($action =~ /modify/i){ &search_db($search_for); $title="Modify Which Record?";
$button_text="Modify Record"; $choose="modify"; &search_results;}
elsif($action =~ /delete/i){ &search_db($search_for); $title="Delete Which Record?";
$button_text="Delete Record"; $choose="delete"; &search_results;}
elsif($action =~ /search/i){ &search_db($search_for); $title="Search Results"; $button_text = "";
$choose="search"; &search_results;}
elsif($action =~ /user manager/i){ $title="User Manager"; &user_manager;}
elsif($action =~ /create new user/i){ $title="User Manager - Create New User"; $perform="add";
$button_text ="Add User"; &edit_user;}
elsif($action =~ /edit selected user/i){ $title="User Manager - Edit Selected User";
$perform="edit"; $button_text ="Edit User"; &edit_user;}
elsif($action =~ /edit user/i){ $old_password = $q->param(old_password); &delete_user;
$perform="edit"; &add_user;}
elsif($action =~ /remove selected user/i){ $perform="delete"; &delete_user; $title="User
Manager"; &user_manager;}
 else { $title="Main Menu"; &print_default($title); }
exit;

################# Add Record ##################

sub add_record {
  foreach $required (@required){
  if($q->param($required) eq "") {
  $title = "Add A Record"; $error_message = "You did not fill out the required information!";
&access_problem($error_message); exit; } }
```

```
  $when_added = &get_date;
  $key   = time();
  $record=$key;
  $record  .= "\|$user";
  $record  .= "\|$when_added";
  foreach $field (@fields){ ${$field}  = $q->param($field); ${$field}  = filter(${$field}); $record
.= "\|${$field}"; }
   unless (-e $database){ open (DB, ">$database") || die "Error creating database.  $!\n"; }
   else { open (DB, ">>$database") || die "Error opening database.  $!\n"; }
   flock DB, $EXCLUSIVE;
   seek DB, 0, 2;
   print DB "$record\n";
   flock DB, $UNLOCK;
  close(DB);
} # End of add_record subroutine.

############## Delete Records #################

sub delete_records
  {
  open (DATABASE, "$database") or die "Error opening file: $!\n";
  while (<DATABASE>)
    {
    $line = $_;
    chop $line;
    @dbfields = split (/\|/, $line);
    $deleted = "no";
    if ($dbfields[0] eq $q->param(key)) { $deleted = "yes"; if ($user eq $dbfields[1]) {
$security_satisfied = "yes"; } }
    elsif ($deleted ne "yes") { $new_data .= "$line\n"; }
    } #end of while
  close (DATABASE);
  if ($authenticate ne "yes") { $security_satisfied = "yes"; }
  if ($security_satisfied ne "yes" && $record_ownership eq "yes" && $admin_user_name ne
"$user")
    {
    $error_message = "You're not authorized to access the record selected!";
    &access_problem($error_message);
    exit;
    }  else  {
    flock DATABASE, $EXCLUSIVE;
    open (DATABASE, ">$database") or die "Error opening file: $!\n";
    print DATABASE "$new_data";
    close (DATABASE);
    flock DATABASE, $UNLOCK;
    }
  }

################## Add Screen ###################

sub print_add_screen{
$title="Add A Record";
&html_header($title);
  print<<HTML;
     <P>
      <TABLE BORDER=0 CELLSPACING=0>
HTML
  foreach $field (@fields){
       print<<HTML;
        <TR>
         <TD><B>\u$field:</B></TD>
         <TD><INPUT TYPE=TEXT NAME="$field"></TD>
        </TR>
HTML
  } # End of foreach.
      print<<HTML;
       <TR>
        <TD COLSPAN=2>
         <CENTER><BR>
          <INPUT TYPE=SUBMIT NAME=action VALUE="Add This Record">
         </CENTER>
        </TD>
       </TR>
      </TABLE>
     <P>
    </FONT>
HTML
&html_footer;
} # End of print_add_screen subroutine.

############### Modify Screen #################

sub print_modify_page{
  ($key,$dbuser,$when_added,@field_vals) = split(/\|/, $results[0]);
$title="Modify Record";
&html_header($title);
  print<<HTML;
   <INPUT TYPE=HIDDEN NAME=key value="$key">
    <TABLE BORDER=0 CELLSPACING=0>
HTML
  $x=0;
  foreach $field (@fields){
    print<<HTML;
     <TR>
      <TD><FONT SIZE=2 FACE=ARIAL><B>\u$field:</B></FONT></TD>
      <TD><INPUT TYPE=TEXT NAME="$field" VALUE="$field_vals[$x]" SIZE=40></TD>
     </TR>
HTML
    $x++;
  } # End of foreach.
print<<HTML;
          </TABLE>
          <BR>
       <INPUT TYPE=SUBMIT NAME=action VALUE="Modify This Record">
   <P>
HTML
&html_footer;
}

############## Results Screen ################

sub search_results{
&html_header($title);
  $rows_left_to_view = $total_row_count - $new_hits_seen;
  $start_hit = $hits_seen + 1;
  $end_hit = $new_hits_seen;
  $hits_displayed = $end_hit - $start_hit + 1;
  if ($total_row_count > 1) {
    print qq~<P><FONT SIZE=2 FACE=ARIAL>Total Found: <B>$total_row_count</B>
records<BR>~;
    if ($total_row_count > $max_rows_returned) {
      if ($hits_displayed == 2) {  print qq~<B>Last two</B> records shown below<BR>~;  }
      elsif ($hits_displayed == 1) {  print qq~<B>Last</B> record shown below<BR>~;  }
      else { print qq~<B>$start_hit</B> - <B>$end_hit</B> shown below<BR>~;  }
    } #end if
  } #end if ... > 1
  else { print qq~<P><FONT SIZE=2 FACE=ARIAL>Total Found: <B>1</B> record\n~; }
if($display =~/columns/i) {
 print<<HTML;
          </FONT><p><TABLE BORDER=0 CELLSPACING=2 CELLPADDING=2
WIDTH=$record_width>
    <TR BGCOLOR="#e0e0e0">
HTML
  if($choose =~ /(modify|delete)/){
    print "<TD ALIGN=CENTER><FONT SIZE=2
FACE=ARIAL><B>Select</B></FONT></TD>\n";
  }
  foreach $field (@fields){
    print "<TD ALIGN=CENTER><FONT SIZE=2
FACE=ARIAL><B>\u$field</B></FONT></TD>\n";
  } # End of foreach
  print "</TR>";
  foreach $record (@results){
    ($key,$dbuser,$when_added,@field_vals) = split(/\|/, $record);
    print "<TR BGCOLOR=\"#efefef\">\n";
    if($choose =~ /(modify|delete)/){
       print "<TD ALIGN=CENTER><FONT SIZE=2 FACE=ARIAL><INPUT TYPE=RADIO
NAME=key VALUE=$key></FONT></TD>\n";
    } # End of if.
      for($x=0;$x<$field_count;$x++){
      $item = &check_item($field_vals[$x], $x);
      print "<TD><FONT SIZE=2 FACE=ARIAL>$item</FONT></TD>\n";
    }
     print "</TR> ";
  } # End of foreach loop.
  print<<HTML;
</TR></TABLE>
HTML
} else {
  foreach $record (@results){
    ($key,$dbuser,$when_added,@field_vals) = split(/\|/, $record);
 print<<HTML;
            <P><TABLE BORDER=0 CELLSPACING=2 CELLPADDING=2
WIDTH=$record_width>
HTML
  if($choose =~ /(modify|delete)/){
    print "<TR BGCOLOR=\"\#e0e0e0\"><TD ALIGN=CENTER COLSPAN=2><FONT
SIZE=2 FACE=ARIAL COLOR=RED><B>Select Record Below</B> <INPUT TYPE=RADIO
NAME=key VALUE=$key></FONT></TD></TR>\n";
  }
  $x=0;
  foreach $field (@fields){
      $item = &check_item($field_vals[$x], $x);
    print<<HTML;
     <TR>
      <TD BGCOLOR="#e0e0e0"><B>\u$field</B></TD>
      <TD BGCOLOR="#efefef" WIDTH=100%>$item</TD>
     </TR>
HTML
    $x++;
            } # End of foreach field loop.
  print "</TABLE><P>";
  } # End of foreach record loop.
} # End of if display loop
```

```
if ($button_text ne ""){ print"<p><INPUT TYPE=submit NAME=action
VALUE=\"$button_text\"><p>"; }
$which_search = $choose;
  if ($rows_left_to_view > 0 && $show ne "no") {
    print "<P><INPUT TYPE=\"hidden\" NAME=\"new_hits_seen\"
VALUE=\"$new_hits_seen\">\n";
    print "<INPUT TYPE=\"hidden\" NAME=\"which_search\" VALUE=\"$which_search\">";
    if ($rows_left_to_view > $max_rows_returned) {
            print "<INPUT TYPE=\"submit\" NAME=\"action\" VALUE=\"View Next
$max_rows_returned Records\">"; }
    elsif ($rows_left_to_view == 1){
            print "<INPUT TYPE=\"submit\" NAME=\"action\" VALUE=\"View Last Record\">"; }
    elsif ($rows_left_to_view <= $max_rows_returned){
            print "<INPUT TYPE=\"submit\" NAME=\"action\" VALUE=\"View Last
$rows_left_to_view Records\">"; }
   }
    print "<INPUT TYPE=\"hidden\" NAME=\"sort_on\" VALUE=\"$sort\">\n";
    print "<INPUT TYPE=\"hidden\" NAME=\"display\" VALUE=\"$display\">\n";
    print "<INPUT TYPE=\"hidden\" NAME=\"search_for\" VALUE=\"$search_for\">\n";
    print "<INPUT TYPE=\"hidden\" NAME=\"search_field\" VALUE=\"$search_field\">\n";
print "<p>";
&html_footer;
} # End of search_results subroutine.

################ Default Screen ##################

sub print_default {
&html_header;
 print<<HTML;
  <TABLE BORDER=0 CELLSPACING="0" CELLPADDING="3"><TR><TD
COLSPAN=3><FONT FACE="ARIAL" SIZE=2><B>Search
            For:</B></FONT><BR><INPUT TYPE="text" NAME="search_for"
SIZE="40"></TD>
   </TR><TR>
    <TD><FONT FACE="ARIAL" SIZE=2><B>Search By:</B></FONT><BR><SELECT
NAME="search_field"><OPTION VALUE="all">All
HTML
  $x=0;
  foreach $field (@fields){  print "<OPTION VALUE=$x>\u$field";  $x++;  }
  print<<HTML;
    </SELECT></FONT> </TD><TD><FONT FACE="ARIAL" SIZE=2><B>Sort
By:</B></FONT><BR><SELECT NAME="sort_on">
HTML
  $x=0;
  foreach $field (@fields){  print "<OPTION VALUE=$x>\u$field";  $x++;  }
  print<<HTML;
```

```
</SELECT></TD>
            <TD><FONT FACE="ARIAL" SIZE=2><B>Display
By:</B></FONT><BR><SELECT NAME="display">
HTML
if($display eq "rows") { print "<OPTION VALUE=\"rows\" SELECTED>Rows"; }
else { print "<OPTION VALUE=\"rows\">Rows"; }
if($display eq "columns") { print "<OPTION VALUE=\"columns\" SELECTED>Columns"; }
else { print "<OPTION VALUE=\"columns\">Columns"; }
  print<<HTML;
</SELECT></TD></TR></TABLE><br>
HTML
$footer="default";
&html_footer;
} # End of print_default subroutine.

########### Search & Sort Database #############

sub search_db{
  my $search_for = $_[0];
   unless (-e $database){ open (DB, ">$database") || die "Error creating database.  $!\n"; }
   else { open (DB, "$database") || die "Error opening database.  $!\n"; }
    while(<DB>){
      if($search_field =~ /all/i){
        if(/$search_for/oi){ push @results, $_};
      } else {
        ($key,$dbuser,$when_added,@field_vals) = split(/\|/, $_);
        if($field_vals[$search_field] =~ /$search_for/oi){ push @results, $_};
      } # End of else.
    } # End of while.
  close (DB);
  foreach $curr (@results){
    ($key,$dbuser,$when_added,@rest) = split(/\|/, $curr);
     $max = @fields;
     $code='$record{$key} = { key => "$key", ';
     $code .='dbuser => "$dbuser", ';
     $code .='when_added => "$when_added", ';
     for($x=0;$x<$max;$x++){
      $code .= "\$fields[$x] => \"\$rest[$x]\",\n";
    } # End of for
     $code .= '};';
    eval $code;
  } # End of foreach curr
   $sort_on = "$fields[$sort]";
   @results=();
  foreach $rp (sort { $a->{$sort_on} cmp $b->{$sort_on} } values %record){
    $new_rec = $rp->{key};
    $new_rec .= "\|$rp->{dbuser}";
    $new_rec .= "\|$rp->{when_added}";
    for($x=0;$x<$max;$x++){
      $new_rec .= "\|$rp->{$fields[$x]}";
    } # End of for
    push @results, $new_rec;
  } # End of foreach
$count = @results;
if($count < 1){ $message="No Matches Found For \"$search_for\""; $title="Main Menu";
&print_default($title, $message); exit;}
$total_row_count = @results;
if ($total_row_count > $max_rows_returned) {
$hits_seen = $q->param(new_hits_seen);
  for ($i = 1; $i <= $hits_seen; $i++)
    { $seen_row = shift (@results);  }
  $length_of_database_rows = @results;
  for ($i = $length_of_database_rows-1; $i >= $max_rows_returned; $i--)
    { $extra_row = pop (@results); }
  $new_hits_seen = $hits_seen + @results; } else { $show="no"; }
} # End of search_db subroutine.

################### Filter ####################

sub filter{
  $temp = $_[0];
  $temp =~ s/\|//g;
  $temp =~ s/\"/'/g;
  foreach $removed (@filter) { $temp =~ s/$removed/\?\$\%\&/gi; }
  if ($filter_html_tags eq "yes") { $temp =~ s/<[^>]*>//gs;  }
  return ($temp);
}

############### Check Item #################

sub check_item{
  $r_val = $_[0];
  $index = $_[1];
  if($r_val =~ /^\s*$/){$r_val="&nbsp;"; return($r_val);}
  if($index_of_url ne "" && $index eq $index_of_url) {
  if ($r_val =~ /http:\/\//i){ $link = "$r_val"; }
  else { $link = "http://"; $link .= "$r_val";  }
  $r_val2 = "<a href=\"$link\">$r_val</a>"; $r_val = $r_val2;
   }
  if($index_of_email ne "" && $index eq $index_of_email) { $r_val2 = "<a
href=\"mailto:$r_val\">$r_val</a>"; $r_val = $r_val2; }
  return($r_val);
```

```
}

################### Login ###################

sub login {
$message = $_[0];
$title = "Login";
&html_header($title,$message);
 print<<HTML;
<TABLE><TR>
<TD><FONT SIZE=2 FACE="ARIAL"><B>User Name</B></FONT><BR><INPUT
TYPE="text" NAME="user" SIZE="30"></TD></TR>
<TR><TD><FONT SIZE=2 FACE="ARIAL"><B>Password</B></FONT><BR><INPUT
TYPE="password" NAME="password" SIZE="30"></TD>
</TR>
<TR><TD ALIGN=MIDDLE><BR><INPUT TYPE="submit" NAME="action"
VALUE="Submit Login">
HTML
if($register eq "yes") { print "<INPUT TYPE=\"submit\" NAME=\"action\"
VALUE=\"Register\">"; }
print<<HTML;
</TD>
</TR>
</TABLE>
HTML
$footer = "login";
&html_footer;
}

################ Authenticate ################

sub authenticate {
   $form_password = $q->param(password);
   open (USERFILE, "$user_file") || die "Error opening user file.  $!\n";
   $user_matched = 0;
   while (<USERFILE>) {
           chop($_);
           ($password, $user, $u_search, $u_add, $u_modify, $u_delete, @extra_user_fields) =
split(/\|/, $_);
           if ($q->param(user) eq $user) {
               $user_matched = 1;
               $username = $user;
               $user_add = $u_add;
               $user_modify = $u_modify;
               $user_search = $u_search;
               $user_delete = $u_delete;
```

```
            if (&auth_encrypt ($form_password, $password) eq $password) {
               $cookie = cookie(-NAME=>'user',-VALUE=>"$user",-EXPIRES=>'+3d');
            } # End of if passwords match
            } # End of if username matches
   } # End of While
   close (USERFILE);
   if ($user_matched == 1 && $cookie eq "") {
               $footer = "login";
               $title = "Login";
               $error_message = "Your password did not match your user name!";
               &access_problem($error_message);
               exit;    }
   if ($user_matched == 0) {
               $footer = "login";
               $title = "Login";
               $error_message = "Your user name was not found!";
               &access_problem($error_message);
               exit;    }
print header(-cookie=>[$cookie]);
$set_cookie = "yes";
$message = "User \"$username\" is now logged in";
$title = "Main Menu";
$user = $username;
&print_default($title, $message);
exit;
}

############### Register Form ################

sub register {
$title = "Register A New User";
&html_header($title);
 print<<HTML;
<TABLE><TR>
<TD><FONT SIZE=2 FACE="ARIAL"><B>User Name</B></FONT><BR><INPUT
TYPE="text" NAME="user" SIZE="30"></TD></TR>
<TR><TD><FONT SIZE=2 FACE="ARIAL"><B>Password</B></FONT><BR><INPUT
TYPE="password" NAME="password" SIZE="30"></TD></TR>
<TR><TD><FONT SIZE=2 FACE="ARIAL"><B>Re-Type
Password</B></FONT><BR><INPUT TYPE="password" NAME="password2"
SIZE="30"></TD></TR>
HTML
  foreach $extra (@extra_user_fields){
    print "<TR><TD><FONT SIZE=2
FACE=\"ARIAL\"><B>\u$extra</B></FONT><BR><INPUT TYPE=\"text\"
NAME=\"$extra\" SIZE=\"30\"></TD></TR>";
```

```
  }
 print<<HTML;
<TR><TD ALIGN=MIDDLE><BR><INPUT TYPE="hidden" NAME="search"
value="$default_user_permission_search"><INPUT TYPE="hidden" NAME="add"
value="$default_user_permission_add"><INPUT TYPE="hidden" NAME="modify"
value="$default_user_permission_modify"><INPUT TYPE="hidden" NAME="delete"
value="$default_user_permission_delete"><INPUT TYPE="submit" NAME="action"
VALUE="Add User"> <INPUT TYPE="submit" NAME="action" VALUE="Back To
Login"></TD>
</TR>
</TABLE>
HTML
$footer = "login";
&html_footer;
}

################ Add User ##################

sub add_user {
$user = $q->param(user);
$password = $q->param(password);
$password2 = $q->param(password2);
if ($perform eq "edit" && $password eq "") { $use_old = "yes"; $password = "$old_password";
$password2 = "$old_password";}
        if ($password ne $password2 || $password eq "" || $password2 eq "") {
           $footer = "login";
           $title = "Register A New User";
           $error_message = "Your passwords didn't match!";
           &access_problem($error_message);
           exit; }
@form_vars = ('user', 'search', 'add', 'modify', 'delete', @extra_user_fields);
   foreach $x (@form_vars) {
        if ($q->param($x) eq "" || $q->param($x) =~ /\|/) {
         $footer = "login";
           $title = "Register A New User";
           $error_message = "You must fill in all of the fields in the registration form!";
           &access_problem($error_message);
           exit; }     }
   if ($check_user_duplicates eq "yes") {
        open (USERFILE, "$user_file");
        $user_matched = 0;
        while (<USERFILE>) {
           chop($_);
           @f = split(/\|/, $_);
           if ($f[1] eq $user) {
                 $user_matched = 1;
```

```
                    last;         }      } # End of while userfile open
            close (USERFILE);
            if ($user_matched == 1) {
               $footer = "login";
               $title = "Register A New User";
               $error_message = "That user name is already taken!";
               &access_problem($error_message);
               exit; } # End of user matched (found duplicate)
            } # End of if check duplicates on
      srand(time|$$);
      $random = "abcdefghijklmnopqrstuvwxyz1234567890";
      $real_password = $password;
      $salt = "";
      for (1..2) { $salt .= substr($random,int(rand(36)),1); }
    if ($use_old ne "yes") {  $password = &auth_encrypt ($password, $salt); }
    $auth=$password;
    foreach $field (@form_vars){
      ${$field}  = $q->param($field);
      $auth   .= "\|${$field}";  }
     unless (-e $user_file){
      open (AUTH_FILE, ">$user_file") || die "Error creating user file.  $!\n";
    } else {
      open (AUTH_FILE, ">>$user_file") || die "Error opening user file.  $!\n";
    }
     flock AUTH_FILE, $EXCLUSIVE;
     seek AUTH_FILE, 0, 2;
     print AUTH_FILE "$auth\n";
     flock AUTH_FILE, $UNLOCK;
    close(AUTH_FILE);
  if ($admin_add eq ""){
     $message = "New User Added<br>You may now login with your new user name below.";
     &login($message);
     } else {
     $title = "User Manager";
     $message = "User Added/Modified";
     &user_manager($message);
     }
  }

  ############### Create Admin ###############

  sub create_admin{
  &html_header($title);
   print<<HTML;
  <center><TABLE WIDTH=275>
  <TR>
```

```
<TD><FONT SIZE=2 FACE="ARIAL">Please start out by setting the Administrator
information below.  When logged in as the below user, you will be able to access additional
options which are only available to the database administrator.</FONT></TD>
</TR><TR>
<TD><br><FONT SIZE=2 FACE="ARIAL"><B>User Name:</B>
$admin_user_name</FONT><br><font size="-2">This can be changed in the script
configurations.</font></TD></TR>
<TR><TD><br><FONT SIZE=2 FACE="ARIAL"><B>Password</B></FONT><BR><INPUT
TYPE="password" NAME="password" SIZE="30"></TD></TR>
<TR><TD><FONT SIZE=2 FACE="ARIAL"><B>Re-Type
Password</B></FONT><BR><INPUT TYPE="password" NAME="password2"
SIZE="30"></TD></TR>
HTML
  foreach $extra (@extra_user_fields){
    print "<TR><TD><FONT SIZE=2
FACE=\"ARIAL\"><B>\u$extra</B></FONT><BR><INPUT TYPE=\"text\"
NAME=\"$extra\" SIZE=\"30\"></TD></TR>";
  }
 print<<HTML;
<TR><TD ALIGN=MIDDLE><BR><INPUT TYPE="hidden" NAME="user"
value="$admin_user_name"><INPUT TYPE="hidden" NAME="search" value="yes"><INPUT
TYPE="hidden" NAME="add" value="yes"><INPUT TYPE="hidden" NAME="modify"
value="yes"><INPUT TYPE="hidden" NAME="delete" value="yes"><INPUT TYPE="submit"
NAME="action" VALUE="Add Administrator"></TD>
</TR>
</TABLE></center>
HTML
$footer = "login";
&html_footer;
exit;
}

################ User Manager ###############

sub user_manager{
$message = $_[0];
&html_header($title,$message);
 print<<HTML;
<center>
<TABLE>
<TR>
<TD><select name="selected_user" size=8>
HTML
    open (USERFILE, "$user_file") || die "Error opening user file.  $!\n";
    while (<USERFILE>) {
            chop($_);
```

```
          ($password, $user, $u_search, $u_add, $u_modify, $u_delete, @extra_user_fields) =
split(/\|/, $_);
          print "<option value=\"$user\">$user</option>\n";
   } # End of While
   close (USERFILE);
  print<<HTML;
</select></TD>
<td valign=top><input type=submit name=action value="Create New User"><p><input
type=submit name=action value="Edit Selected User"><p><input type=submit name=action
value="Remove Selected User">
</TR>
</TABLE>
</center>
HTML
&html_footer;
exit;
}

############### Add/Edit User ################

sub edit_user {
if ($perform eq "edit" && $selected_user eq "" ) {
             $title = "User Manager - Edit Selected User";
             $error_message = "You forgot to select a user record to edit!";
             &access_problem($error_message);
             exit;
}
if ($perform eq "edit") {
   open (USERFILE, "$user_file") || die "Error opening user file.  $!\n";
   while (<USERFILE>) {
          chop($_);
          ($password, $userfound, $u_search, $u_add, $u_modify, $u_delete, @other_user_fields)
= split(/\|/, $_);
          if ($selected_user eq $userfound) {
             @found_other_fields = @other_user_fields;
             $edit_user = $userfound;
             $edit_user_add = $u_add;
             $edit_user_modify = $u_modify;
             $edit_user_search = $u_search;
             $edit_user_delete = $u_delete;
             $old_password = $password;
          } # End of if username matches
   } # End of While
   close (USERFILE);
}
&html_header($title);
```

```
 print<<HTML;
<TABLE><TR>
<TD><FONT SIZE=2 FACE="ARIAL"><B>User Name</B></FONT><BR><INPUT
TYPE="text" NAME="user" SIZE="30" value="$edit_user"></TD></TR>
HTML
if ($perform eq "edit") {
  print<<HTML;
<TR><TD><FONT SIZE=2 FACE="ARIAL"><B>New Password</B></FONT><br><font
size="-2">Leave blank if you aren't changing passwords.</font><BR><INPUT
TYPE="password" NAME="password" SIZE="30"></TD></TR>
HTML
} else {
  print<<HTML;
<TR><TD><FONT SIZE=2 FACE="ARIAL"><B>Password</B></FONT><BR><INPUT
TYPE="password" NAME="password" SIZE="30"></TD></TR>
HTML
}
if ($perform eq "edit") {
  print<<HTML;
<TR><TD><FONT SIZE=2 FACE="ARIAL"><B>Re-Type New
Password</B></FONT><br><font size="-2">Leave blank if you aren't changing
passwords.</font><BR><INPUT TYPE="password" NAME="password2"
SIZE="30"></TD></TR>
HTML
} else {
  print<<HTML;
<TR><TD><FONT SIZE=2 FACE="ARIAL"><B>Re-Type
Password</B></FONT><BR><INPUT TYPE="password" NAME="password2"
SIZE="30"></TD></TR>
HTML
}
$extra_fields_count = @found_other_fields;
      for($x=0;$x<$extra_fields_count;$x++){
print "<TR><TD><FONT SIZE=2
FACE=\"ARIAL\"><B>\u$extra_user_fields[$x]</B></FONT><BR><INPUT TYPE=\"text\"
NAME=\"$extra_user_fields[$x]\" SIZE=\"30\"
value=\"$found_other_fields[$x]\"></TD></TR>";
    }
if ($perform eq "add") {
  foreach $extra (@extra_user_fields){
    print "<TR><TD><FONT SIZE=2
FACE=\"ARIAL\"><B>\u$extra</B></FONT><BR><INPUT TYPE=\"text\"
NAME=\"$extra\" SIZE=\"30\"></TD></TR>";
  }
}
 print<<HTML;
```

```
<tr><td><br><FONT SIZE=2 FACE="ARIAL"><b>User Permissions</b></font></td></tr>
HTML

if ($edit_user_search eq "no"){
  print<<HTML;
<tr><td><FONT SIZE=2 FACE="ARIAL">Search: </font><input type="radio" name="search"
value="yes"> Yes &nbsp;&nbsp;&nbsp;<input type="radio" name="search" value="no"
checked> No</td></tr>
HTML
} else {
  print<<HTML;
<tr><td><FONT SIZE=2 FACE="ARIAL">Search: </font><input type="radio" name="search"
value="yes" checked> Yes &nbsp;&nbsp;&nbsp;<input type="radio" name="search"
value="no"> No</td></tr>
HTML
}
if ($edit_user_add eq "no"){
  print<<HTML;
<tr><td><FONT SIZE=2 FACE="ARIAL">Add: </font><input type="radio" name="add"
value="yes"> Yes &nbsp;&nbsp;&nbsp;<input type="radio" name="add" value="no" checked>
No</td></tr>
HTML
} else {
  print<<HTML;
<tr><td><FONT SIZE=2 FACE="ARIAL">Add: </font><input type="radio" name="add"
value="yes" checked> Yes &nbsp;&nbsp;&nbsp;<input type="radio" name="add" value="no">
No</td></tr>
HTML
}
if ($edit_user_modify eq "no"){
  print<<HTML;
<tr><td><FONT SIZE=2 FACE="ARIAL">Modify: </font><input type="radio"
name="modify" value="yes"> Yes &nbsp;&nbsp;&nbsp;<input type="radio" name="modify"
value="no" checked> No</td></tr>
HTML
} else {
  print<<HTML;
<tr><td><FONT SIZE=2 FACE="ARIAL">Modify: </font><input type="radio"
name="modify" value="yes" checked> Yes &nbsp;&nbsp;&nbsp;<input type="radio"
name="modify" value="no"> No</td></tr>
HTML
}
if ($edit_user_delete eq "no"){
  print<<HTML;
```

```
<tr><td><FONT SIZE=2 FACE="ARIAL">Delete: </font><input type="radio" name="delete"
value="yes"> Yes &nbsp;&nbsp;&nbsp;<input type="radio" name="delete" value="no"
checked> No</td></tr>
HTML
} else {
  print<<HTML;
<tr><td><FONT SIZE=2 FACE="ARIAL">Delete: </font><input type="radio" name="delete"
value="yes" checked> Yes &nbsp;&nbsp;&nbsp;<input type="radio" name="delete"
value="no"> No</td></tr>
HTML
}
  print<<HTML;
<TR><TD ALIGN=MIDDLE><BR><input type=hidden name="old_password"
value="$old_password"><input type=hidden name="selected_user" value="$edit_user"><input
type=hidden name="admin_add" value="yes"><INPUT TYPE="submit" NAME="action"
VALUE="$button_text"> <INPUT TYPE="submit" NAME="action" VALUE="User
Manager"></TD>
</TR>
</TABLE>
HTML
&html_footer;
}

################ Delete User ###############

sub delete_user{
if ($selected_user eq "" ) {
                $title = "User Manager - Remove Selected User";
                $error_message = "You forgot to select a user record to remove!";
                &access_problem($error_message);
                exit;
}
if ($selected_user eq "$admin_user_name" && $perform eq "delete") {
                $title = "User Manager - Remove Selected User";
                $error_message = "You are not allowed to remove the administrator!";
                &access_problem($error_message);
                exit;
}
  open (USERFILE, "$user_file") or die "Error opening file: $!\n";
  while (<USERFILE>)
    {
    $line = $_;
    chop $line;
    @dbfields = split (/\|/, $line);
    $deleted = "no";
    if ($dbfields[1] eq $selected_user) { $deleted = "yes"; }
    elsif ($deleted ne "yes") { $new_data .= "$line\n"; }
    } #end of while
  close (USERFILE);
    flock USERFILE, $EXCLUSIVE;
    open (USERFILE, ">$user_file") or die "Error opening file: $!\n";
    print USERFILE "$new_data";
    close (USERFILE);
}

################# Logout ###################

sub logout {
$cookie = cookie(-NAME=>'user',-VALUE=>"$user",-EXPIRES=>'-1d');
print header(-cookie=>[$cookie]);
$set_cookie = "yes";
$message = "User \"$user\" logged out";
&login($message);
 } # end of logout

############ Encrypt Password ###############

sub auth_encrypt {
    local ($field, $salt) = @_;
    $field = crypt ($field, $salt);
    $field;
 } # end of encrypt

############## Access Problem ##############

sub access_problem {
&html_header($title);
 print<<HTML;
<BR>
<FONT SIZE=4 FACE="ARIAL" COLOR=RED><B>Error!</B></FONT><P>
<FONT SIZE=4 FACE="ARIAL">$error_message</FONT><P>
<INPUT TYPE="button" VALUE="  Back  " onClick="history.go(-1)"><br><br>
HTML
&html_footer;
exit;
}

################ Get Date ##################

sub get_date  {
  local ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst,$date);
  local (@days, @months);
  @days = ('Sunday','Monday','Tuesday','Wednesday','Thursday','Friday','Saturday');
  @months =
('January','February','March','April','May','June','July','August','September','October','November','
December');
  ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) = localtime(time);
  if ($hour < 10)    {    $hour = "0$hour";    }
  if ($min < 10)     {    $min = "0$min";    }
  if ($sec < 10)    { $sec = "0$sec";    }
  $mon++;
 $year += 1900;
 $date = "$mon/$mday/$year at $hour\:$min\:$sec";
  return $date;
  }

############## Form Header #################

sub form_header {
print<<HTML;
<FORM METHOD="post" ACTION="$script_name">
HTML
}

############## Form Footer #################

sub form_footer {
if($footer eq "default") {
if ($user_search eq "yes") {print "<INPUT TYPE=\"submit\" NAME=\"action\"
VALUE=\"Search\"> "; }
if ($user_add eq "yes") {print "<INPUT TYPE=\"submit\" NAME=\"action\" VALUE=\"  Add
\"> "; }
if ($user_modify eq "yes") {print "<INPUT TYPE=\"submit\" NAME=\"action\"
VALUE=\"Modify\"> "; }
if ($user_delete eq "yes") {print "<INPUT TYPE=\"submit\" NAME=\"action\"
VALUE=\"Delete\"> "; }
if ($authenticate eq "yes") { print " <INPUT TYPE=\"submit\" NAME=\"action\"
VALUE=\"Logout\">\n"; }
}
elsif($footer eq "login") { print "\&nbsp\;"; }
 else { print "<INPUT TYPE=\"submit\" NAME=\"action\" VALUE=\"Main Menu\">"; if
($authenticate eq "yes") { print " <INPUT TYPE=\"submit\" NAME=\"action\"
VALUE=\"Logout\">";} }
print "<input type=hidden name=\"user_search\" value=\"$user_search\">\n";
print "<input type=hidden name=\"user_add\" value=\"$user_add\">\n";
print "<input type=hidden name=\"user_modify\" value=\"$user_modify\">\n";
print "<input type=hidden name=\"user_delete\" value=\"$user_delete\">\n";
}
```

```
############## HTML Header #################

sub html_header{
if($set_cookie eq "") { print $q->header; }
$set_cookie = "";
if($use_external_html eq "yes") {
%external_variables = (title=>$title, message=>$message,);
print html_template("$external_html_header", \%external_variables);
&form_header;
} else {
 print<<HTML;
<HTML><HEAD>
   <TITLE>$db_name - $_[0]</TITLE>
  </HEAD><BODY BGCOLOR="#FFFFFF">
  <CENTER><FONT SIZE=5 FACE="ARIAL"><B>
   $db_name
  </B></FONT><p><FONT SIZE=3 FACE="ARIAL"
COLOR=RED><B>$_[1]</B></FONT></CENTER><P>
   <CENTER>
   <TABLE BORDER=1 WIDTH="$table_width" CELLSPACING="0" CELLPADDING="3">
    <TR>
    <TD BGCOLOR="#e0e0e0">
     <CENTER>
         <FONT SIZE=3 FACE="ARIAL"><B>$_[0]</B></FONT>
     </CENTER>
    </TD>
   </TR>
    <TR>
     <TD align=middle><BR>
HTML
&form_header;
}
}

############## HTML Footer #################

sub html_footer{
if($use_external_html eq "yes") {
%external_variables = (title=>$title, message=>$message,);
&form_footer;
print html_template("$external_html_footer", \%external_variables);
if ($user eq "$admin_user_name" && $footer eq "default" && $authenticate eq "yes") {
print "<center><input type=\"submit\" name=\"action\" value=\"User Manager\"></center><p>";
}
 else {
```

```
 print<<HTML;
   <BR></TD></TR>
  <TR>
   <TD BGCOLOR="#e0e0e0"><CENTER>
HTML
&form_footer;
 print<<HTML;
    </CENTER>
   </TD>
  </TR>
</TABLE><P>
HTML
if ($user eq "$admin_user_name" && $footer eq "default" && $authenticate eq "yes") {
print "<center><input type=\"submit\" name=\"action\" value=\"User Manager\"></center><p>";
}
  print<<HTML;
</FORM>
 </CENTER>
</BODY></HTML>
HTML
          }
}

############ Use External HTML ###########################
##  Thank you Tom Christiansen & Nathan Torkington for this routine
##  Taken from the Perl Cookbook

sub html_template {
my ($filename, $fillings) = @_;
my $text;
local $/;
local *F;
open (F, "< $filename\0") || return;
$text = <F>;
close (F);
$text =~ s{%%( .*? )%%}
            {exists($fillings->{$1})
                    ?$fillings->{$1}
                    : ""
            }gsex;
return $text;
}
```

References

1. *Center for Satellite and Hybrid Communication Networks @* http://isr.umd.edu/CSHCN
2. *The Official Apache Developers' Sight @* http://www.apache.org
3. *The Unix vs. NT site @* http://www.unix-vs-nt.org
4. *The PostgreSQL official Site @* http://www.postgresql.org
5. *Momjian, Bruce (2000). PostgreSQL: Introduction and Concepts, Addison Wesley*
6. *The Official PHP developer's site @* http://www.php.net
7. *Bakken, Stig Saether(2000). PHP Manual, PHP Documentation Group*
8. *PHPbuilder.com*
9. *The Official Perl Site @* http://www.perl.org