

ADMINISTERING A LINUX-BASED SERVER
FOR STUDENT PROGRAMMERS

John Phillips
Mansfield University

Nicholas Andre
Mansfield University

Joo Tan
Mansfield University

Matthew Phillips
Mansfield University

April 2, 2003

ABSTRACT

Administering a multi-user Linux server offers many challenges. One server configuration that has been useful in web-programming courses taught at Mansfield University is based around Red Hat Linux 8.0 and includes an Apache web server, a MySQL database server, and support for the Perl, PHP, and Python programming languages. Each student has his or her own account allowing access to a private home directory and a private database. The student can access the server using Telnet, SSH, and FTP. The Vi text editor and numerous other Linux programs are supported as well. Overall, this gives a very flexible system that offers many teaching/learning opportunities.

This report will start by looking at the initial setup and configuration of the server. Scripts designed to automate the creation and deletion of bulk student accounts will be covered; specifically, given a roster from WebAdvisor, automatically parse it and generate the student Linux and database accounts. The report will conclude with a discussion of some of the other challenges faced by faculty system administrators.

1. INTRODUCTION

In the fall of 2000, the Mansfield University Department of Mathematics and Computer Information Science decided to offer a web-programming course sequence based around the Perl programming language running on a Linux server. The campus IT department was not anxious to support anything other than Microsoft Windows, therefore it was up to the instructors to setup and manage a Linux server in support of the new courses.

The desired Linux environment needed to support an HTTP web server, an SQL database server, an FTP server, an SSH server, a Telnet server, and web-programming languages such as Perl, PHP, and Python. We went with the popular choices and chose Apache for the web server and MySQL for the database server. [Netcraft03] A standard Red Hat 8.0 Linux distribution contained all of the required software. [Redhat02, Negus02] As an added benefit the software was open-source and freely available over the Internet.

Our current Linux configuration is running on an older Pentium III 450 MHz computer with 512 MB of RAM and an 80 GB hard disk drive. The server is housed in one of the instructor's offices. It connects to the campus network with a 100 Mbps Ethernet connection and is accessible to the Internet through the campus firewall.

The purpose of this paper is to document the current installation, configuration, and administration of our Linux server.

2. INSTALLATION

In this section, the steps we followed to install Red Hat Linux 8.0 are discussed. A complete listing of the steps followed can be found in Appendix A. The set of four CD images can be downloaded from one of the Red Hat Linux mirror servers listed at <http://www.redhat.com/download/mirror.html> or the CD set can be purchased from Red Hat.

2.1 Getting Started

We start by booting up the computer with the first Red Hat CD in the CDROM drive. Problems booting from the CD may indicate the need to change the computer's BIOS settings so that it boots from the CD first. A graphical install is selected and we answer a few questions about languages, keyboard, and mouse.

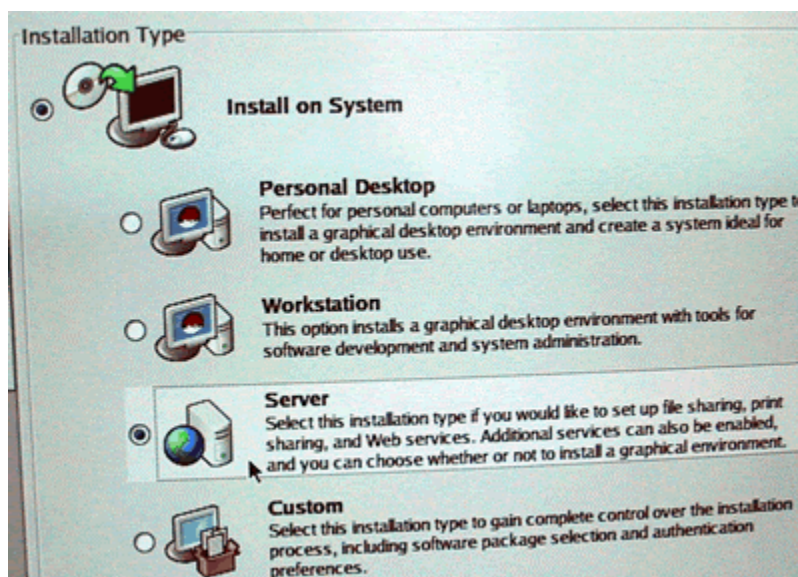


Figure 2.1 – Setting the installation type.

For the *Installation Type* we choose *Server* as seen in figure 2.1. This will tell the Red Hat installation program that we want to use this machine as a server. Based on this information, it will automatically allocate the hard disk space and choose the packages it thinks a server should have. We can and will override some of the choices, but this is a good starting point.

2.2 Disk Partitioning

For this step we tell the installer to automatically partition and then to remove all partitions of the system. We make sure to check the *Review the partitions created* checkbox.

Device	Mount Point/RAID/Volume	Type	Format	Size (MB)	Start	End
Hard Drives						
/dev/hda						
/dev/hda1	/boot	ext3	✓	102	1	13
/dev/hda2	/usr	ext3	✓	2989	14	394
/dev/hda3	/home	ext3	✓	1090	395	533
/dev/hda4		Extended		1969	534	784
/dev/hda5	/	ext3	✓	510	534	598
/dev/hda6	/var	ext3	✓	824	599	703
/dev/hda7		swap	✓	635	704	784

Hide RAID device/LVM Volume Group members

Figure 2.2 – Adjusting the partition table.

A partitioning screen (see figure 2.2) is presented showing us the choices Red Hat has made. We can adjust the partitions as necessary. When a large hard disk is present we consider increasing the size of the /var partition to 5 GB or more. /var contains the error logs that can grow in size very quickly with a large class of Perl programmers. In addition, it contains the MySQL databases.

Network Devices

Active on Boot	Device	IP/Netmask
<input checked="" type="checkbox"/>	eth0	157.62.24.202/255.255.255.0

Edit

Hostname

Set the hostname:

automatically via DHCP

manually

Miscellaneous Settings

Gateway:	157	62	24	155
Primary DNS:	157	62	2	5
Secondary DNS:				
Tertiary DNS:				

Figure 2.3 – Setting the network IP addresses.

2.3 Network Configuration

Clicking the *Edit* button when the *Network Configuration* screen appears allows us to enter the server's IP address and netmask (see figure 2.3). We set a name for the server and fill in the *Miscellaneous Settings* including the *Gateway* and *Primary DNS* values. These values were obtained from our campus network administrator.

2.4 Firewall Configuration

Configuring the firewall is an important step in controlling the security and accessibility of the server. When the *Firewall Configuration* screen appears as shown in figure 2.4, we choose a security level of *Medium* and allow incoming connections on *WWW*, *FTP*, *SSH*, *DHCP*, and *Telnet*. We do not choose *Mail (SMTP)* as we did not want to risk our server being turned into an email spam machine. Users are allowed to send email to other users within the confines of the server itself, just not external to the server.

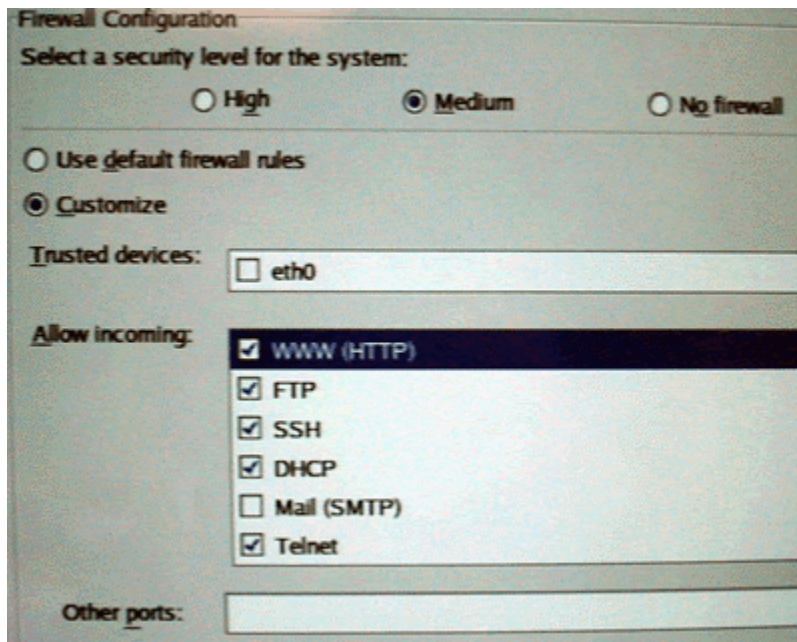


Figure 2.4 – Configuring the firewall.

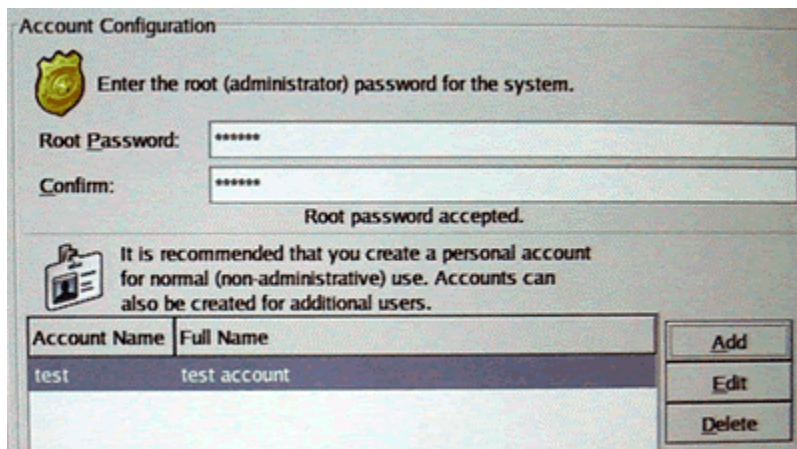


Figure 2.5 – Adding a user test account.

2.5 Account Configuration

It will be useful later in the setup process to have a standard user account available. Figure 2.5 shows the creation of an account named "test".

2.6 Package Group Selection

We now choose the packages that we want to install. Packages contain software applications. The packages are organized by categories and subcategories. Selecting an entire subcategory will select a popular subset of packages.

2.6.1 Desktops Category

Under the *Desktops* category we choose *X Window System* and *GNOME Desktop Environment*. These are not strictly necessary but they often help simplify our system administration chores.

2.6.2 Applications Category

Under the *Applications* category the *Editors* checkbox is selected and then we click the *Details* hyperlink on the right. We select *vim enhanced* only and deselect the other options. We select the *Graphical Internet* option so that a web browser will be installed. We choose *Text-based Internet* and within *Details* choose *lynx*. Lynx is a text-based web browser that is optional but can be useful at times.

2.6.3 Server Category

Under the *Servers* category we select *Server Configuration Tools*, *Web Server*, *FTP Server*, *SQL Database Server*, and *Network Servers*. Under the details of *SQL Database Server* we make sure to select *mysql-server*. Under the details of *Network Servers* we select *telnet-server*. Telnet is less secure but is more convenient for our students in many cases.

2.6.4 Development Category

Under the *Development* category we select *Development Tools*. This will give us several programming languages including Perl, C, C++, and Python.

2.6.5 System Category

Under the *System* category select *Administration Tools*. We do not have a printer connected to the server so we do not choose *Printing Support*.

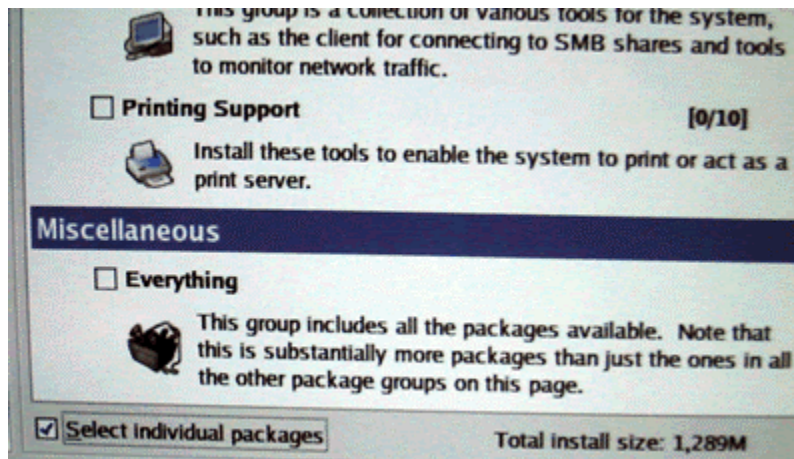


Figure 2.6 – Check the select individual packages checkbox.

2.6.6 Select Individual Packages Checkbox

It is very important to check the *Select individual packages* checkbox at the bottom of the screen as seen in figure 2.6.

2.7 Individual Package Selection

We now choose a few individual packages. As we plan on using the PHP programming language with MySQL, it is very important to check *php-mysql* under the *Development* category and in the *Languages* subcategory as shown in figure 2.7. Under the *Development* category and the *System* subcategory we check *rpmdb-redhat*. Under the *Documentation* category we check the *php-manual*. Other interesting packages can be selected now as well.

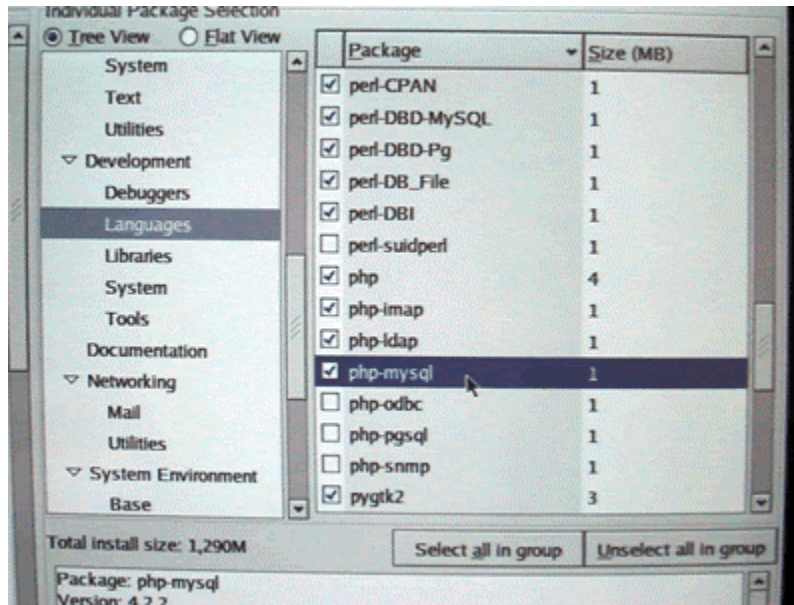


Figure 2.7 – Individual package selection.

2.8 Finishing Up the Install

Red Hat starts installing the software using all three CDs. Next a boot disk is created and the graphical interface is configured. When customizing the graphical configuration, we choose a login type of Text as shown in figure 2.8. When Linux boots up, it will start off in the text-based console mode. We can start up the graphical environment at any time by typing *startx* at the command prompt.

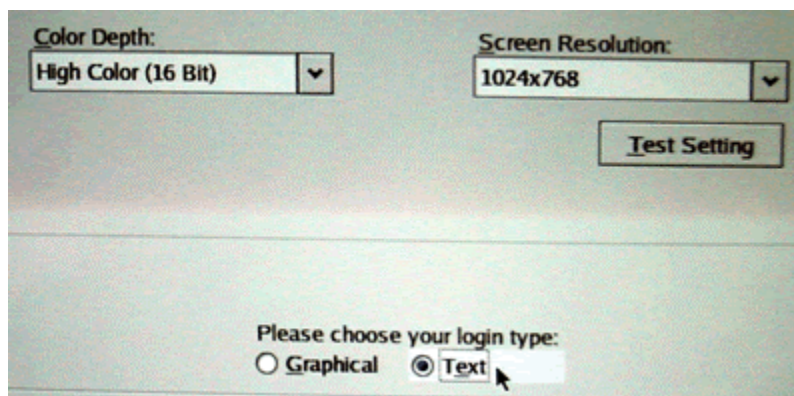


Figure 2.8 – Setting the login type to text.

3. CONFIGURATION

At this point the server is somewhat operational, yet many of the key services are not running. In addition, many critical security updates may have occurred and should be applied to the new installation. This section will discuss procedures to make the server fully operational.

3.1 Red Hat Alert Notification Tool

A great feature of Red Hat Linux 8.0 is the Red Hat Alert Notification Tool. This tool shows up in the lower right hand corner of the x window screen as a red circle with an explanation point whenever there is a critical update available. The service is free, however registration with Red Hat Network is required. When setting up a production server (as opposed to a test server), we click the red circle and follow the directions to register.

After registering, we check the *Select all packages* checkbox to install a new kernel and apply all of the critical updates available. The red circle becomes a blue circle with a check mark indicating the installation is up to date. We reboot the machine to apply the new kernel.

3.2 What is Working?

It is interesting to check and see just what is working and what isn't. We log in as root and type *startx* to enter the graphical environment. Clicking the earth/mouse icon at the bottom left side of the screen launches a web browser. We enter the URL of a site that we know is working, for example, <http://www.slashdot.org>. If the page comes up then we know that the Internet connection is working correctly.

On another machine on our network, we launch a web browser and type in the IP address of our Linux server. A message is received stating that the page cannot be displayed. This is because we have not yet configured our Apache web server.

Next we try to Telnet into our Linux server from a secondary network machine. We get a message saying that the remote connection was refused. That is because we need to turn on the Telnet service along with the FTP service and a few others.

Finally we try to SSH into our Linux server from the secondary network machine using VanDyke's SecureCRT program. [VanDyke03] We log in as the test user. This works correctly as SSH is enabled by default. We cannot log in from a remote machine as the root user. However, after logging in as our test user we can become root by using the *su* command and then entering the root password when prompted.

3.3 Using chkconfig

We can use a program called *chkconfig* to turn on and off any services that have been installed. In general, we only turn on services that are needed. We begin by logging in as root either staying in the console mode or using x windows. If using x windows then we open up a terminal screen by clicking on the red hat GNOME menu icon and then choosing *System Tools* and *Terminal*.

We type *chkconfig - list | more* to see the current settings and press the space bar to move through the listing. Sshd and xinetd are on for run levels 3, 4, and 5, whereas httpd and mysqld are off. Under xinetd based services we see that vsftpd and telnet are both off. We need to turn all of these services on.

To turn on a service we type *chkconfig servicename on* replacing servicename with the name of the service to activate. On our server we typed the following:

```
# chkconfig telnet on
# chkconfig talk on
# chkconfig ntalk on
```

```
# chkconfig vsftpd on
# chkconfig mysqld on
# chkconfig httpd on
```

Typing `chkconfig --list | more` again allows us to verify that these services are on for run levels 3, 4, and 5. At this point Telnet and FTP should work. However, `mysqld` and `httpd` need to be started to get them going. The easiest way to do this is to restart the computer by typing `# shutdown -r now`. Or we could start each service explicitly. Apache `httpd` can be started by typing `# httpd -k start` and `mysqld` can be started by typing `# /etc/init.d/mysqld start`. We still need to configure `mysqld` and `httpd` for them to work properly.

3.4 Configure MySQL

In this section we will configure and test the MySQL database server.

3.4.1 Setting the MySQL Root Password

The MySQL password for root is set as shown below. [Widenius03]

```
# mysqladmin -u root password "mu1234"
```

Note that MySQL passwords can be different from the Linux user account passwords. We can try out the new password by giving the following command:

```
# mysqladmin -u root -p version
```

When prompted we type in our mysql password which is `mu1234` in this case. The MySQL version information is displayed.

3.4.2 Setting Up a User Database

It is useful to see how to set up a user's database by hand. In general, we will name each database the same as the username. However, MySQL comes with a database named `test`. So user `test` will get a database named `testuser` instead. The following commands are entered:

```
# mysql -p
password: mu1234
mysql> create database testuser;
mysql> grant all on testuser.* to test@localhost
identified by "test";
mysql> quit;
```

These commands create a database named `testuser`. Next we grant all privileges on the database `testuser` to the user `test@localhost`. `test@localhost` needs to be a valid Linux user name (recall that we created a Linux account for a user name `test`). The `identified by "test"` clause sets the MySQL database password for this user. We can use any password that we want here. However, for our simple educational database we will give each user a password that is the same as their user name.

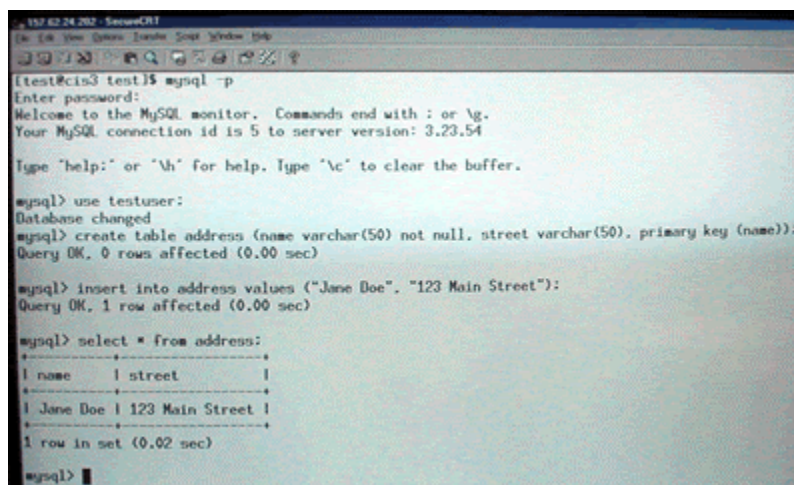
3.4.3 Testing the User Database

To test the user account we just created, we log in as that user and create a simple database table. We can login as user `test` either directly on the Linux server or by connecting using Telnet

or SSH from another computer on the network. The mysql query tool is launched and the following commands are entered:

```
$ mysql -p
password: test
mysql> use testuser;
mysql> create table address (name varchar(50) not null,
street varchar(50), primary key (name));
mysql> insert into address
values ("Jane Doe", "123 Main Street");
mysql> select * from address;
mysql> quit;
```

The session should look something like what is shown in figure 3.4.3.



```
mysql -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5 to server version: 3.23.54

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use testuser;
Database changed
mysql> create table address (name varchar(50) not null, street varchar(50), primary key (name));
Query OK, 0 rows affected (0.00 sec)

mysql> insert into address values ("Jane Doe", "123 Main Street");
Query OK, 1 row affected (0.00 sec)

mysql> select * from address;
+-----+-----+
| name | street |
+-----+-----+
| Jane Doe | 123 Main Street |
+-----+-----+
1 row in set (0.02 sec)

mysql>
```

Figure 3.4.3 – Using mysql monitor to test user database.

3.4.4 MySQL Password Shortcut

To save time, the Vi text editor (see Appendix E) can be used to create a special file to hold our password. The password will no longer need to enter it each time we give a MySQL command. At the prompt we type *vi .my.cnf* and press enter. Enter the following:

```
[client]
password=mu1234
```

3.5 Configure Apache httpd

The Apache web server has a large configuration file. However, it is very well commented. Table 3.5 shows the changes we make to our configuration file. We start by changing into the Apache httpd configuration directory by entering *cd /etc/httpd/conf*. A backup copy of the configuration file is made by typing *cp httpd.conf httpd.conf.bak*. Then we edit the configuration file by typing *vi httpd.conf*. Using the Vi text editor, we can quickly jump to any line by typing a colon followed by the line number, i.e. *: 266*.

Line #	Change	Comment
266	ServerName 157.62.24.202:80	Uncomment & set the host name. We are using our ip address for the host name.
365	# UserDir disable	Comment out this line with # symbol.
372	UserDir public_html	Uncomment this line.
380 – 391		Uncomment these lines.
382		Change IncludesNoExec to ExecCGI.
510	ServerSignature Off	
827	AddHandler cgi-script .cgi .pl .py	Uncomment and add .pl and .py.

Table 3.5 – Changes to made to the Apache httpd.conf file.

Line 372 is what lets each user have their own website. For example, we can reach our test user's web site by typing `http://157.62.24.202/~test/` in any web browser. Note however, that each user must first create a `public_html` directory and set up the proper permissions as shown in section 3.6 and later automated in section 4. Lines 380 to 391 control access to the `public_html` directories. By changing line 382 to `ExecCGI` we are telling Apache that it is okay to execute CGI programs in this directory (see figure 3.5). Line 510 tells Apache not to display our email address on a page that does not load properly. Finally, line 827 tells Apache that files ending in `.cgi`, `.pl`, or `.py` are to be treated as CGI programs.

```
# Control access to UserDir directories. The following is an
# for a site where these directories are restricted to read-on
#
<Directory /home/*/public_html>
    AllowOverride FileInfo AuthConfig Limit
    Options MultiViews Indexes SymLinksIfOwnerMatch ExecCGI
    <Limit GET POST OPTIONS>
        Order allow,deny
        Allow from all
    </Limit>
    <LimitExcept GET POST OPTIONS>
        Order deny,allow
        Deny from all
    </LimitExcept>
</Directory>
```

Figure 3.5 – A section of the httpd.conf file.

After making changes to the `httpd.conf` file it is necessary to restart the Apache web server with the command: `# httpd -k restart`.

3.6 Creating a Web Site Directory for a User

To find out if the Apache web server is working correctly, we connect to it from another computer by entering the Linux server IP address into the web browser URL field. A web page titled Test Page should appear.

Next we need to try a user's directory. However, first we must create one by logging in as the test user. To create the necessary directory and set the permissions we type in the following:

```
$ mkdir public_html
$ chmod 755 public_html
$ cd ..
$ chmod 711 test
```

```
$ cd
```

The above commands create the `public_html` in the user's home directory. The `chmod 755` command sets the directory to have read, write, and execute permissions for the user and read and execute permissions for everyone else. The `cd ..` command moves us up a directory level so that we are in the `/home` directory. The `chmod 711` command sets permissions so that others can execute programs found in the user's account but they cannot read or write any files there. The final `cd` command brings us back to the home directory. Using a web browser from another machine, we should now be able to access any files in our `public_html` directory by typing a URL of `http://157.62.24.202/~test/`.

3.7 Log files

The Apache `httpd` error log is a very useful tool for debugging CGI programs. However, it is only available to the root user by default. To give other users access to the error log, we log in as root and give the following commands:

```
# cd /var/log/  
# chmod 755 httpd  
# cd
```

Now when you log in as a regular user you can access the error log by typing:

```
$ tail -25 /var/log/httpd/error_log
```

3.8 Changing Network Information

The current network settings can be viewed by typing the command `ifconfig`. We can temporarily change a value by entering commands such as `ifconfig eth0 157.62.24.202` or `ifconfig eth0 netmask 255.255.255.0`. The settings can be permanently changed by editing the `/etc/sysconfig/network-scripts` file. It should look something like figure 3.8. Running the graphical application found at *GNOME - System Settings - Network* can also set these values.

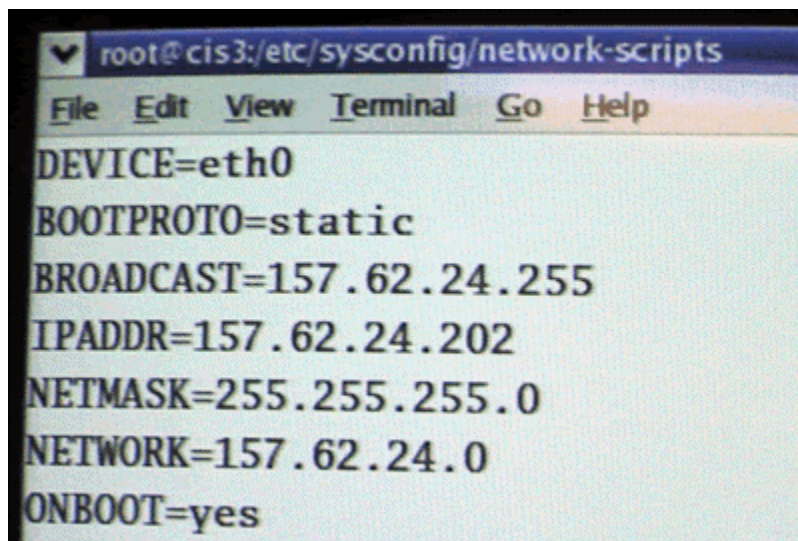


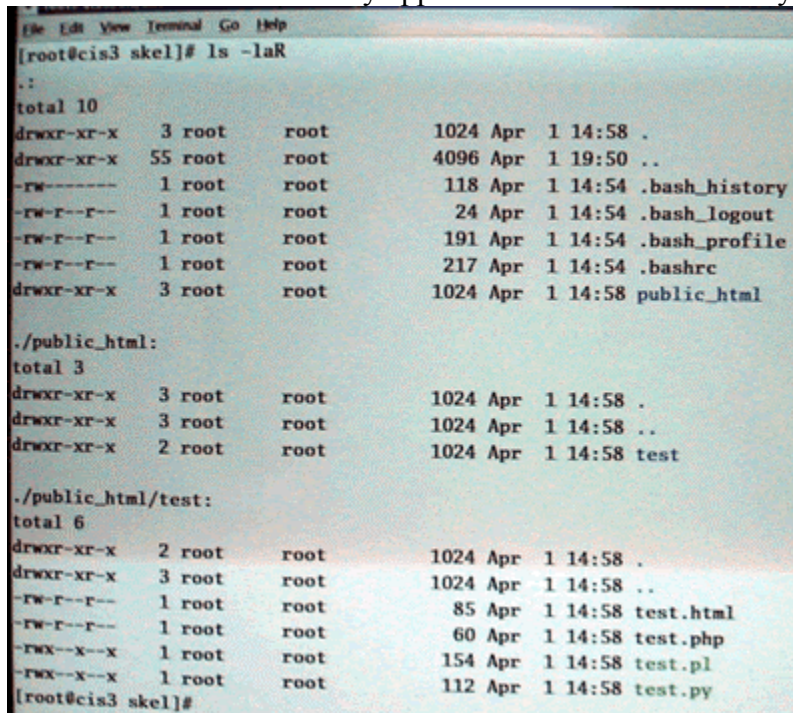
Figure 3.8 – The network-scripts file.

4. AUTOMATING STUDENT ACCOUNT CREATION

This section will discuss several ways we have reduced the system administrator workload. One key to making this system successful is to reduce any system administration burdens the instructor may face. The largest time burden is the setup and management of student accounts. These accounts include both the Linux login account and the SQL database account.

4.1 The /etc/skel Directory

As each student account is created, the contents of the /etc/skel directory are automatically copied into the student's home directory. In the home directory, a public_html directory is created to store files that will be made available to the web. A few sample programs are placed in the student's public_html directory. Finally, we find it useful to place some common alias commands in the student's .bashrc file. For example, the following alias command is useful for reviewing the Apache error log file: *alias log='tail -50 /var/log/httpd/error_log'*. Now the student can just type the word log and then see the last 50 lines of the error log without typing in the full command. The /etc/skel directory as seen in figure 4.1 allow these and other similar tasks to be setup ahead of time and be automatically applied to new accounts as they are created.



```
[root@cis3 skel]# ls -laR
.:
total 10
drwxr-xr-x  3 root  root    1024 Apr  1 14:58 .
drwxr-xr-x 55 root  root   4096 Apr  1 19:50 ..
-rw-----  1 root  root    118 Apr  1 14:54 .bash_history
-rw-r--r--  1 root  root    24 Apr  1 14:54 .bash_logout
-rw-r--r--  1 root  root   191 Apr  1 14:54 .bash_profile
-rw-r--r--  1 root  root   217 Apr  1 14:54 .bashrc
drwxr-xr-x  3 root  root   1024 Apr  1 14:58 public_html

./public_html:
total 3
drwxr-xr-x  3 root  root    1024 Apr  1 14:58 .
drwxr-xr-x  3 root  root    1024 Apr  1 14:58 ..
drwxr-xr-x  2 root  root    1024 Apr  1 14:58 test

./public_html/test:
total 6
drwxr-xr-x  2 root  root    1024 Apr  1 14:58 .
drwxr-xr-x  3 root  root    1024 Apr  1 14:58 ..
-rw-r--r--  1 root  root    85 Apr  1 14:58 test.html
-rw-r--r--  1 root  root    60 Apr  1 14:58 test.php
-rwx--x--x  1 root  root   154 Apr  1 14:58 test.pl
-rwx--x--x  1 root  root   112 Apr  1 14:58 test.py
[root@cis3 skel]#
```

Figure 4.1 – Contents of the /etc/skel directory.

4.2 Parsing a WebAdvisor Roster Using PHP

At our institution we are provided with a web interface to our class rosters. To create the student accounts, the instructor simply copies the online class roster and pastes it into a web form. Upon submitting the form, our PHP program (see Appendix B) parses the roster and produces a nicely formatted list of the essential student data. The format we use is the student's school assigned email name followed by a tab and then the student's name. The email name is unique and is used as the student's account name. This is saved as a text file.

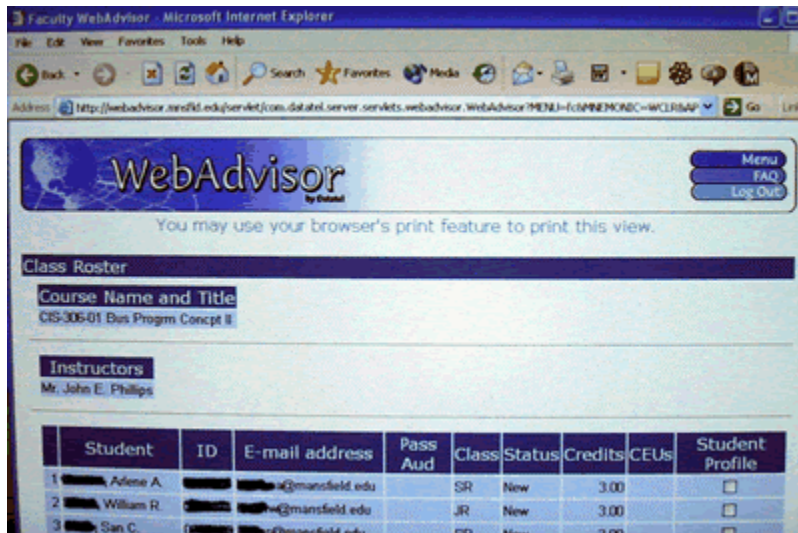


Figure 4.2 – WebAdvisor class roster.

Figure 4.2 shows a WebAdvisor class roster. We use this as a starting point when creating our student Linux accounts. We start by selecting the table starting with the title bar (Student, ID, E-mail address, etc.). We then copy and paste this into rosterform2.html running on a web server as shown in figure 4.2b.

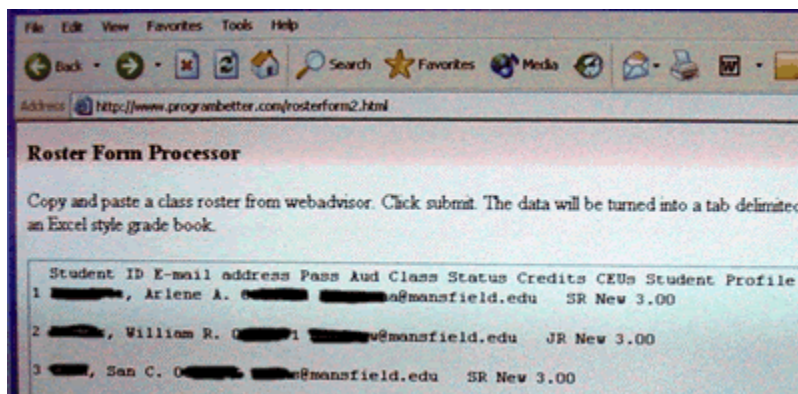


Figure 4.2b – Pasting roster data into rosterform2.html.

Clicking submit passes the raw WebAdvisor table to rosterprocess2.php which parses the form data and returns a list of student info. After clicking View - Source on the browser, the results appear in a file similar to those shown in figure 4.2c.

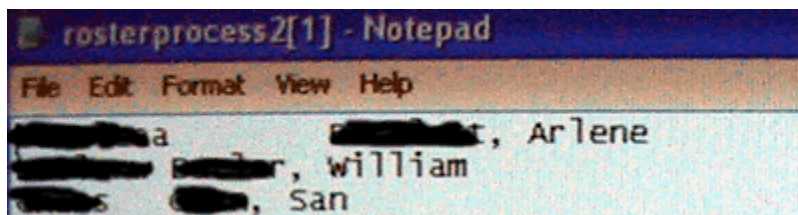


Figure 4.2c – Results from rosterprocess2.php.

4.3 Create Linux and MySQL Accounts using Perl

The second step is to feed this text file into a Perl program (see Appendix C) that automatically creates the Linux and MySQL accounts. The entire process to add a class of students takes about a minute. Figure 4.3 shows the results when creating batch accounts using a data file containing John, Paul, George, and Ringo of The Beatles.

```
***** Creating Linux account for Paul McCartney with a use
Changing password for user mccartnp.
passwd: all authentication tokens updated successfully.
Changing permissions on /home/mccartnp/ to 711
Creating MySQL database using: create database mccartnp;
Granting database permissions using: grant all on mccartnp.
cartnp";

***** Creating Linux account for George Harrison with a use
Changing password for user harrisog.
passwd: all authentication tokens updated successfully.
Changing permissions on /home/harrisog/ to 711
Creating MySQL database using: create database harrisog;
Granting database permissions using: grant all on harrisog.
rrisog";

***** Creating Linux account for Ringo Starr with a user id
Changing password for user starr
```

Figure 4.3 – Perl program creating batch Linux and MySQL accounts.

5. AUTOMATING STUDENT PROGRAM COUNTS

In one of our web programming classes students are required to complete a large number of small Perl programs. To help track these programs and each student's progress, we developed a Perl script (see Appendix D) that will scan each student's directory and produce a report showing what programs have been completed so far. We then do spot checks on certain problems to make sure the students are really doing them and that the quality is acceptable. Figure 5.3 shows a sample report produced by the program.

```
lennonj
a1: a1, a2, a3,
a2: a2, a5,
a3: a1,
Total = 6

mccartnp
a1: a1, a2,
Total = 2

starr
a1: a1, a2,
a2: a1, a2, a3,
a3: a1,
Total = 6
[root@cis3 admin]#
```

Figure 5.3 – Program to count student assignments.

6. CONCLUSION

We have learned a great deal by operating our own Linux web server. There have been no major problems in nearly three years of operation. In fact, the server is so trouble free that we tend to do system administration work at the beginning of each semester and forget what we have done by the time the next semester starts. This document was written to document the steps we followed so that it will be easier for us to do next time. Hopefully, these instructions will be useful to others who want to set up their own server as well. We plan to update and further improve this document as we move on to new Linux distributions. [Phillips03]

REFERENCES

- [Apache03] The Apache Software Foundation. "Apache HTTP server version 2.0 documentation". Available at: <http://httpd.apache.org/docs-2.0/>. Accessed March 28, 2003.
- [Ipswitch03] Ipswitch, Inc. WS_FTP LE download web site. Available at: <http://ipswitch.com/downloads/index.html>. Accessed March 28, 2003.
- [Lemay02] Lemay, L. "Sams teach yourself Perl in 21 days, 2nd ed.", Sams, 2002.
- [Negus02] Negus, C. "Red Hat Linux 8 Bible". Wiley, 2002.
- [Netcraft03] Netcraft, Ltd. "March 2003 Web Server Survey". Apache has 62% market share. Available at: <http://news.netcraft.com/>. Accessed April 2, 2003.
- [Phillips03a] Phillips, J. "John Phillips Web Site". This site will contain the latest versions of this document. Available at: <http://www.mnsfld.edu/~jphillip/research.html>. Accessed April 2, 2003.
- [Phillips03b] Phillips, J. "Vi Reference Sheet". The HTML version of this document is available at: <http://www.mnsfld.edu/~jphillip/documents/vi.html>. Accessed April 2, 2003.
- [PHP03] The PHP Group. "PHP manual". 2003. Available at: <http://www.php.net/manual/en/>. Accessed March 22, 2003.
- [Ramirez03] Ramirez, C., "Perl Documentation Website". Available at: <http://perldoc.com/>. Accessed March 26, 2003.
- [Redhat02] Red Hat, Inc. "Red Hat Linux manuals". 2002. Available at: <http://www.redhat.com/docs/manuals/linux/>. Accessed March 28, 2003.
- [Tiobe03] Tiobe Software, "Tiobe software programming community index for March 2003". Available at: <http://www.tiobe.com/tpci.htm>. Accessed March 25, 2003.
- [VanDyke03] VanDyke Software, Inc. SecureCRT web site. Available at: <http://www.vandyke.com/products/securecrt/index.html>. Accessed March 28, 2003.
- [Widenius03] Widenius, M., Lentz, A., & DuBois, P. "MySQL reference manual." Available at: <http://www.mysql.com/documentation/mysql/bychapter/index.html>. Accessed March 22, 2003.

APPENDIX A – Step by Step Red Hat 8.0 Server Installation

1. Press Enter on boot screen to choose graphical install.
2. Skip CD test.
3. Welcome to Red Hat Linux – press next
4. English is selected – press next
5. Keyboard is U.S. English – press next
6. Generic 2 button ps/2 mouse is selected – press next
7. Installation type – Server is selected – press next
8. Disk partitioning setup—automatically partition is selected – press next
9. Automatic partitioning – remove all partitions on this system is selected – check the "Review the partitions created" checkbox
10. Warning "Are you sure?" – Yes
11. Partitioning – adjust as necessary; consider making /var several gigabytes if large drive
12. Boot loader configuration – press next
13. Network configuration – press edit – set IP address to 157.62.24.202 and netmask to 255.255.255.0 – press Ok -- Set host name to cis3 – set Gateway to 157.62.24.155 and set Primary DNS to 157.62.2.5. Note that these values are unique to this one setup on the MU network. Your values will be different. Use 192.168.0.1 for your IP address and 192.168.0.255 for your Gateway (as starting values) if you are setting up a private or home network and get your DNS number from your ISP.
14. Firewall configuration – choose Medium – customize choosing WWW(HTTP), FTP, SSH, DHCP, Telnet -- we do not choose Mail (SMTP) -- press next.
15. Additional language – English (USA) – press next
16. TimeZone – America/New_York – press next
17. root password – carefully enter password in both boxes – click Add – create a test account named test and choose a password of mu2003 – Ok – click next
18. Package group selection – choose:
 - a. X windows system
 - b. GNOME desktop environment
 - c. Editors – click details and choose vim enhanced only – Ok
 - d. Graphical Internet
 - e. Text based Internet – click details and choose lynx – Ok
 - f. Server configuration tools
 - g. Web server
 - h. Deselect Windows file server
 - i. FTP server

- j. SQL database server – click details and choose mysql-server – Ok
 - k. Network servers – click details and make sure telnet-server is selected – Ok
 - l. Development tools
 - m. Administration tools
 - n. Deselect printing support unless you plan to connect a printer to your server
 - o. Make sure to check the individual packages checkbox
19. Individual package selection – click development and select php-mysql – click system and choose rpmdb-redhat – click documentation and choose php-manual – click next
 20. Click next to install
 21. Insert disc 2 when prompted
 22. Insert disc 3 when prompted
 23. Boot disk creation – choose yes – click next – insert diskette – Ok – when finished remove disk and label properly.
 24. Graphical interface configuration – click next
 25. Monitor configuration – click next
 26. Customize graphical configuration – set login type to text
 27. Install complete – exit
 28. Red Hat reboots
 29. Login as either test or as root.

APPENDIX B – PHP Script to Parse WebAdvisor Roster

B.1 PHP Roster Parser

This PHP program will parse the WebAdvisor roster submitted from the form shown on the next page and produce output in the form of emailname tab lastname, firstname. This data can then be used to automate the creation of Linux and MySQL student accounts.

```
<?
# rosterprocess2.php by John Phillips on 8/23/2001 revised 4/1/2003

# Called by rosterform2.html

# Take an html roster from web advisor and convert it into tab delimited text data.
# This can be easily imported into Excel or used to create Linux accounts.
# The result is emailname tab lastname, firstname\n

$roster = $_REQUEST['roster'];

if( ! empty($roster) ) {

    # Search through web advisor form and replace everything through and
    # including Credits CEUs with ". This will get us near the student data.

    $result = ereg_replace( ".*Credits CEUs", "", $roster );

    # Split the file up into separate rows of student data.

    $rArray = explode( "\n", $result );

    # Get each row from the array and look at that student's data.

    while( list(, $value) = each ( $rArray ) ) {

        # Use a regular expression to extract the data that we want.
        # The data format is: 1 Lennon, John A. 1234567 lennonj@mansfield.edu SR New 3.00
        # ([0-9]{1,2}) matches 1 or 2 numbers; ([a-z]+, [a-z]+) matches lastname, firstname
        # [^0-9]* matches 0 or more characters that are not a number--gets rid of middle initial
        # ([0-9]{7}) matches the 7 digit student id number; a space and then
        # ([a-z]+)@mansfield.edu matches the unique email name of the student.

        if(eregi("([0-9]{1,2}) ([a-z]+, [a-z]+) [^0-9]*([0-9]{7}) ([a-z]+)@mansfield.edu",$value,$r)){

            # Print out the data in the format: lennonj\tLennon, John\n

            print "$r[4]\t$r[2]\n";

        }
    }
}
else {
    print "no roster data";
}
?>
```

B.2 HTML Form

Copy the WebAdvisor roster table of student data and paste it into this HTML form.

```
<html>
<head>
<title>Roster Processor 2</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body bgcolor="#FFFFFF" text="#000000">
<h3>Roster Form Processor</h3>
<p>Copy and paste a class roster from webadvisor. Click submit. The data will
  be turned into a tab delimited text file that can be used to set up an Excel
  style grade book or create Linux accounts.</p>
<form name="rosterform2" method="post" action="rosterprocess2.php">
  <p>
    <textarea name="roster" cols="100" rows="20" wrap="OFF"></textarea>
  </p>
  <p>
    <input type="submit" name="Submit" value="Submit">
  </p>
</form>
<p>&nbsp;</p>
</body>
</html>
```

APPENDIX C – Perl Script to Batch Create Linux and MySQL Accounts

```
#!/usr/bin/perl -w

# adduser2.pl by John Phillips on 01/16/2003
# Based on code by Nicholas Andre.

# This program will read in a textfile of user names and create
# both a Linux account and a MySQL account. The Linux account
# will have a default password as specified in the file pw.txt.
# The MySQL database will be named with the student's login name.
# The MySQL database password will be the login name as well.
# The students.txt file contains the students to add and
# is in the form:
#
# emailname (tab) lastname, firstname\n
# or
# emailname lastname, firstname\n
#
# Call as: perl adduser2.pl students.txt

use DBI;

$dbAdmin = "root";
$dbPassword = "mul234";

print "Connecting to MySQL using DBI\n";
$dbh = DBI->connect('DBI:mysql:mysql', $dbAdmin, $dbPassword)
    or die "Couldn't connect to database: " . $dbh->errstr;

while( <> ) {
    chomp;
    if( /^(\\w+)[ \\t](.+), (\\w+)/ ) {
        $user = $1; $first = $3; $last = $2;
        print "\n***** Creating Linux account for $first $last with a user id of $user\n";
        system("/usr/sbin/useradd -c '$first $last' $user");
        system("/usr/bin/passwd --stdin $user < pw.txt");

        print "Changing permissions on /home/$user/ to 711\n";
        chmod( 0711, "/home/$user/" );

        $q = "create database $user;";
        print "Creating MySQL database using: $q\n";
        $sth = $dbh->prepare($q) or die "Could not prepare statement: $dbh->errstr";
        $sth->execute();

        #Mysql command: grant all on dbname.* to user@localhost identified by "userpassword";
        $q = "grant all on $user.* to $user\\@localhost identified by \\\"$user\\\";";
        print "Granting database permissions using: $q\n";
        $sth = $dbh->prepare( $q ) or die "Could not prepare statement: $dbh->errstr";
        $sth->execute();
    }
}
$dbh->disconnect();
```

APPENDIX D – Perl Script to Automate Student Program Counts

This program assumes that all accounts in the /home directory are students in the course being examined. It assumes that assignments are placed in directories named a1, a2, and so on. It assumes that the programs in any given directory are named a1.pl, a2.pl, a3.pl, etc.

```
#!/usr/bin/perl -w

# a2.pl by John Phillips on 10/10/2002
# Program to count up student homework assignments.

@students = glob "/home/*";

foreach $student ( @students ) {
    $total = 0;
    @subdir = glob "$student/a?";
    if( $subdir[0] ) {
        if( $student =~ /home\/(\w+)/ ) {
            print "\n$1\n";
        }
        foreach $assignment ( @subdir ) {
            if( $assignment =~ /\home\/\w+\/(a\d)/ ) {
                print "$1: ";
            }
            @studentFiles = glob "$assignment/a*.pl";
            foreach $file ( @studentFiles ) {
                if( $file =~ /\home\/\w+\/\w+\/(\w+)\.pl/ ) {
                    print "$1, ";
                    $total++;
                }
            }
            print "\n";
        }
        print "Total = $total\n";
    }
    else {
        # print "$student does not have any assignments\n";
    }
}
```

APPENDIX E – Vi Reference Sheet

HTML version is online at <http://www.mansfield.edu/~jphillip/documents/vi.html>

\$ vi [filename] - start vi	:wq - write file and quit	:q! - quit without saving
:help - start help system	: - follow help reference	^T - back to prev. reference
U - undoes all edits on 1 line	u - undo	^L - redraw the screen
h, j, k, l - move cursor 1 space	4l - move cursor 4 to right	4k - move cursor up 4 lines
0 - move to beginning of line (zero)	\$ - move to end of line	w - move cursor forward 1 word
W - move cursor forward, ignore punct.	b - backward by word	B - by word ignore punct.
^f, ^b, ^d, ^u - scroll by screen		:44 - goto line 44
H - move cursor to top	M - move cursor to middle	L - move cursor to last line scrn
z <Enter> - move current line to top of screen	z. - move cur. line to center	z- - move cur. line to bottom
(- move to beginning of current sentence) - beginning of next sent.	d) - deletes to end of current sent.
{ - beginning of current paragraph	} - beginning of next par.	2y} - yanks 2 paragraphs ahead
44G - goto beginning of line 44	^G - reports total lines	d?move - del back through move
/pattern - searches forward for a match	?pattn - search backward	/,?,n,N - next match
:50,100s/old/new/g - change old to new	:1,\$s/old/new/g - all lines	:1,30s/his/the/gc - with confirm
i - insert text / Esc to exit insert mode	a - append text	c - change text
cw - change word (del & then insert)	cc - change entire line	C - from cursor to eol
r - replace a single letter & stay in com. mode	rW - replace a word	R - overstrike mode for 1 line
~ - change the case of a letter	x - delete single character	50i* - inserts 50 asterisks
dw - delete word	dd - delete entire line	d\$ or D - del to end of line
p - put deleted text after or below cursor	P - put above	yw, y\$, 4yy - yank (copy)
a7yy - yank 7 lines to buffer a	fyy - yank curr. line to buf-f	fp - put buffer f after cursor
!:date - execute UNIX command	:sh - start new shell (ctrl-D)	:r newfile - reads file into your file
:ab - list abbreviations	:ab abbr phrase - add abbr.	:unab abbr - remove abbr.
:set autoindent - turns on indenting	ctrl-T - add indent	ctrl-D - take indent away
:set shiftwidth=4	5>> - shift 5 lines right	<< - shift line left
J - joins 2 lines into 1	% - find matching bracket	. - repeats last command
\$ vi file1 file2 - edit multiple files	:e filename - edit another file	:w - write current file
:n - show next file	^^ - switch back to prev. file	:args - shows files open
:split [filename] - create a new window	^w^w - switch between win	^w=, ^w-, ^w+ - resize wins