Experiment 1

To Study basic & User status Unix/Linux Commands

man command.

This is help command, and you can use man in conjunction with any command to learn more about that command for example.

- man ls will explain about the ls command and how you can use it.
- To quit man command page **press q**.

cal command

cal [month] [year]

cal command will print the calander on current month by default. If you want to print calander of august of 1965. That's eightht month of 1965.

cal 8 1965 will print following results.

```
    August 1965
 S  M Tu  W Th  F  S
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

clear command

clear command clears the screen and puts cursor at beginning of first line.

pwd command.

pwd command will print your home directory on screen, pwd means print working directory.

```
/home/stdt/c067890
```

ls command

ls command is most widely used command and it displays the contents of directory.

- ls will list all the files in your home directory, this command has many options.
- ls -l will list all the file names, permissions, group, etc in long format.
- ls –d will list directories.

Optıons of ls command

-a	Shows you all files, even files that are hidden (these files begin with a dot.)
-A	List all files including the hidden files. However, does not display the working directory (.) or the parent directory (..).
-b	Force printing of non-printable characters to be in octal \ddd notation.

-c	Use time of last modification of the i-node (file created, mode changed, and so forth) for sorting (-t) or printing (-l or -n).
-C	Multi-column output with entries sorted down the columns. Generally this is the default option.
-d	If an argument is a directory it only lists its name not its contents.
-f	Force each argument to be interpreted as a directory and list the name found in each slot. This option turns off -l, -t, -s, and -r, and turns on -a; the order is the order in which entries appear in the directory.
-F	Mark directories with a trailing slash (/), doors with a trailing greater-than sign (>), executable files with a trailing asterisk (*), FIFOs with a trailing vertical bar (|), symbolic links with a trailing at-sign (@), and AF_Unix address family sockets with a trailing equals sign (=).
-g	Same as -l except the owner is not printed.
-i	For each file, print the i-node number in the first column of the report.
-l	Shows you huge amounts of information (permissions, owners, size, and when last modified.)
-L	If an argument is a symbolic link, list the file or directory the link references rather than the link itself.
-m	Stream output format; files are listed across the page, separated by commas.
-n	The same as -l, except that the owner's UID and group's GID numbers are printed, rather than the associated character strings.
-o	The same as -l, except that the group is not printed.
-p	Displays a slash (/) in front of all directories.
-q	Force printing of non-printable characters in file names as the character question mark (?).
-r	Reverses the order of how the files are displayed.
-R	Includes the contents of subdirectories.
-s	Give size in blocks, including indirect blocks, for each entry.
-t	Shows you the files in modification time.
-u	Use time of last access instead of last modification for sorting (with the -t option) or printing (with the -l option).

-x	Displays files in columns.
-1	Print one entry per line of output.
pathnames	File or directory to list.

The * wildcard

The character * is called a wildcard, and will match against none or more character(s) in a file (or directory) name. For example, in your unixstuff directory, type

% ls list*

This will list all files in the current directory starting with list....

Try typing

% ls *list

This will list all files in the current directory ending with ....list

The ? wildcard

The character ? will match exactly one character.

So ?ouse will match files like house and mouse, but not grouse.

Try typing

% ls ?list

ls ~

List the contents of your home directory by adding a tilde after the ls command.

ls /

List the contents of your root directory.

ls ../

List the contents of the parent directory.

ls */

List the contents of all sub directories.

ls -d */

Only list the directories in the current directory.

date command.

Date displays todays date, to use it type date at prompt.

date '+DATE: %m/%d/%y%nTIME:%H:%M:%S' - Would list the time and date in the below format:

DATE: 02/08/01
TIME:16:44:55

write command will initiate an interactive conversation with user. Syntax is ----- write *username*

who command

who command displays information about the current status of system.
Who as default prints login names of users currently logged in.

whoami --- returns your username. Sounds useless, but isn't. You may need to find out who it is who forgot to log out somewhere, and make sure you have logged out.

passwd --- lets you change your password

logout or **exit** **---** ends terminal session.

Input Redirection--- Input can be given from the command line, or from a file

a.out < inputfile

Output Redirection--- goes to your screen, or you can "redirect" it to a file

echo "hi there" > out.file

To append, use >>

echo "hi back at you" >> out.file

mkdir command.----- mkdir bag will create new directory, i.e. here bag directory is created.

cd command.

cd bag will change directory from current directory to bag directory.
Use pwd to check your current directory and ls to see if bag directory is there or not.
You can then use cd bag to change the directory to this new directory.

rmdir command.

rmdir command will remove directory or directories if a directory is empty.

Options:

- rm -r directory_name will remove all files even if directory is not empty.
- rmdir direc1 is how you use it to remove direc1 directory.
- rmdir -p will remove directories and any parent directories that are empty.
- rmdir -s will suppress standard error messages caused by -p.

cat command

cat cal.txt cat command displays the contents of a file here cal.txt on screen (or standard out).

This is one of the most flexible Unix commands. We can use to create, view and concatenate files. For our first example we create a three-item English-Turkish dictionary in a file called "dict."

```
$ cat >dict
  red kirmizi
  green yesil
  blue mavi
<control-D>
```

<control-D> stands for "hold the control key down, then tap 'd'". The symbol `>` tells the computer that what is typed is to be put `into` the file `dict`. To view a file we use `cat` in a different way:

```
$ cat dict
  red kirmizi
  green yesil
  blue mavi
$
```

If we wish to add text to an existing file we do this:

```
$ cat >>dict
  White beyaz
  Black siyah
  <control-D>
$
```

Now suppose that we have another file tmp that looks like this:

```
$ cat tmp
  cat kedi
  dog kopek
$
```

Then we can join `dict` and `tmp` like this:

```
$ cat dict tmp >dict2
```

Options for cat command

filename	The name of the file or files that you wish to look at or perform tasks on.
-n	Precede each line output with its line number.
-b	Number the lines, as -n, but omit the line numbers from blank lines.
-u	The output is not buffered. (The default is buffered output.)
-s	cat is silent about non-existent files.
-v	Non-printing characters (with the exception of tabs, new-lines and form-feeds) are printed visibly. ASCII control characters (octal 000 - 037) are printed as ^n, where n is the corresponding ASCII character in the range octal 100 - 137 (@, A, B, C, . . ., X, Y, Z, [, \,], ^, and _); the DEL character (octal 0177) is printed ^?. Other non-printable characters are printed as M-x, where x is the ASCII character specified by the low order seven bits.
-e	A $ character will be printed at the end of each line (prior to the new-line).
-t	Tabs will be printed as ^I's and formfeeds to be printed as ^L's.

more command.-----is used to display contents of the file one page at a time.

$ more tmp

----More---(25%)

% will give the percentage of file that has been viewed.

cp command.
cp command copies a file. If I want to copy a file named oldfile in a current directory to a file named newfile in a current directory.
cp oldfile newfile
If I want to copy oldfile to other directory for example /tmp then
cp oldfile /tmp/newfile.

mv command.

mv command is used to move a file from one directory to another directory or to rename a file.

Some examples:

- mv oldfile newfile will rename oldfile to newfile.
- mv -i oldfile newfile for confirmation prompt.
- mv -f oldfile newfile will force the rename even if target file exists.

- mv * /usr/bag/ will move all the files in current directory to /usr/bag directory.

rm command.

To delete files use rm command.

Options:

- rm oldfile will delete file named oldfile.
- rm -f option will remove write-protected files without prompting.
- rm -r option will delete the entire directory as well as all the subdirectories, very dangerous command.

head command.--------- head [-number | -n number] filename

head filename by default will display the first 10 lines of a file.
If you want first 50 lines you can use head -50 filename or for 37 lines head -37 filename and so forth.

tail command.------ tail filename by default will display the last 10 lines of a file.

If you want last 50 lines then you can use tail -50 filename.

tail [+ number] [-l] [-b] [-c] [-r] [-f] [-c number | -n number] [file]

+number	
-l	Units of lines.
-b	Units of blocks.
-c	Units of bytes.
-r	Reverse. Copies lines from the specified starting point in the file in reverse order. The default for r is to print the entire file in reverse order.
-f	Follow. If the input-file is not a pipe, the program will not terminate after the line of the input-file has been copied, but will enter an endless loop, wherein it sleeps for a second and then attempts to read and copy further records from the input-file. Thus it may be used to monitor the growth of a file that is being written by some other process.
-c number	The number option-argument must be a decimal integer whose sign affects the location in the file, measured in bytes, to begin the copying: + Copying starts relative to the beginning of the file. - Copying starts relative to the end of the file. none Copying starts relative to the end of the file.

	The origin for counting is 1; that is, -c+1 represents the first byte of the file, -c-1 the last.
-n number	Equivalent to -c number, except the starting location in the file is measured in lines instead of bytes. The origin for counting is 1; that is, -n+1 represents the first line of the file, -n-1 the last.
file	Name of the file you wish to display

chmod -- is used to change permissions on a file.

for example if we have a text file with calender in it called cal.txt.initially when this file will be created the permissions for this file depends upon ***umask*** set in your profile files. As you can see this file has **666** or **-rw-rw-rw** attributes.

ls -l cal.txt

-rw-rw-rw- 1 ssb dxidev 135 Dec 3 16:14 cal.txt

In this line above I have -rw-rw-rw- meaning respectively that owner can read and write file, member of the owner's group can read and write this file and anyone else connected to this system can read and write this file., next ssb is owner of this file dxidev is the group of this file, there are 135 bytes in this file, this file was created on December 3 at time16:14 and at the end there is name of this file. Learn to read these permissions in binary, like this for example Decimal 644 which is 110 100 100 in binary meand rw-r--r-- or user can read,write this file, group can read only, everyone else can read only. Similarly, if permissions are 755 or 111 101 101 that means rwxr-xr-x or user can read, write and execute, group can read and execute, everyone else can read and execute. All directories have d in front of permissions. So if you don't want anyone to see your files or to do anything with it use chmod command and make permissions so that only you can read and write to that file, i.e.

chmod 600 filename

u: user g: group o: other a: all r: read w: write x: execute +: add permission -: take away permission

chmod go-rwx biglist

Experiment 2

Study & use of commands for performing arithmetic operations with Unix/Linux

bc is the basic calculator. If you enter bc , it will only do integer calculations. However, if you call it with the -l flag (i.e., bc -l), it will do floating point calculations.

dc is the desk calculator. It is very similar to bc, with two major differences: It does floating point calculations by default, and it uses reverse Polish notation to enter numbers and operators. For example, to add 1 + 2 and to see the result, you would have to type 1 2 + p . Also, dc does not have as many operations available as bc.

To exit bc or dc, press **Ctrl-d** , or enter quit

expr command is used to perform arithmetic calculation .

$ x=2 y=8; z= expr $x + $y

$ echo $z

10

Experiment 3

Create a file called wlcc.txt with some lines and display how many lines, words and characters are present in that file.

wc command

wc command counts the characters, words or lines in a file depending upon the option.

Options

- wc -l filename will print total number of lines in a file.
- wc -w filename will print total number of words in a file.
- wc -c filename will print total number of characters in a file.

Experiment

$ **cat >wlcc.txt**

red kirmizi

green yesil

blue mavi

<control-D>

$ wc wlcc.txt

3 6 32

Note: 3 Lines, 6 Words, 32 Characters

Experiment 4

Append ten more simple lines to the wlcc.txt file created above and split the appended file into 3 parts. What will be the names of these split files? Display the contents of each of these files. How many lines will be there on the last file?

split [-linecount | -l linecount] [-a suffixlength] [file [name]]

split -b n [k | m] [-a suffixlength] [file [name]]

-linecount | -l linecount	Number of lines in each piece. Defaults to 1000 lines.
-a suffixlength	Use suffixlength letters to form the suffix portion of the filenames of the split file. If -a is not specified, the default suffix length is 2. If the sum of the name operand and the suffixlength option-argument would create a filename exceeding NAME_MAX bytes, an error will result; split will exit with a diagnostic message and no files will be created.
-b n	Split a file into pieces n bytes in size.
-b n k	Split a file into pieces n*1024 bytes in size.
-b n m	Split a file into pieces n*1048576 bytes in size.
file	The path name of the ordinary file to be split. If no input file is given or file is -, the standard input will be used.
name	The prefix to be used for each of the files resulting from the split operation. If no name argument is given, x will be used as the prefix of the output files. The combined length of the basename of prefix and suffixlength cannot exceed NAME_MAX bytes; see OPTIONS.

Examples

split -b 22 newfile.txt new - would split the file "newfile.txt" into three separate files called newaa, newab and newac each file the size of 22.

split -l 300 file.txt new - would split the file "newfile.txt" into files beginning with the name "new" each containing 300 lines of text each

Experiment 5

Given two files each of which contains names of students. Create a program to display only those names that are found on both the files.

comm [-1] [-2] [-3] file1 file2

-1	Suppress the output column of lines unique to file1.
-2	Suppress the output column of lines unique to file2.
-3	Suppress the output column of lines duplicated in file1 and file2.
file1	Name of the first file to compare.
file2	Name of the second file to compare.

Examples

comm myfile1.txt myfile2.txt

The above example would compare the two files myfile1.txt and myfile2.txt.

Experiment 6

Create a program to find out the inode number of any desired file.

When a file system is created, data structures are created that contain information about files. Each file is associated with an inode that is identified by an inode number (often referred to as an "i-number" or "inode") in the file system where it resides.

Inodes store information on files, such as user and group ownership, access mode (read, write, execute permissions) and type of file. On many types of file systems the number of inodes available is fixed at file system creation, limiting the maximum number of files the file system can hold. A typical fraction of space allocated for inodes in a file system is 1% of total size.

Each file on a Unix filesystem has a inode number associated with it; knowing the inode number of the bad file allows us to search for and delete it.

In computing, an inode is a data structure on a traditional Unix-style file system such as UFS. An inode stores basic information about a regular file, directory, or other file system object.

ls -li a.out

8230 -rwxr-xr-x 1 root sys 24576 Jun 19 08:10 a.out

8230 is the Inode number of file a.out.

Experiment 7

Study & use of the Command for changing file permissions.

The chmod/chgrp/chown commands are used to change the permissions/ownership of files and/or directories. Linux is often used as a multi-user system and it is not desirable that all users have access to all files and directories.

For eg. : On a multi-user environment in a corporate office using a central server running linux , it might be required the accounts documents be shared between employees of the acccounts department . At the same time, it might be undesirable and indeed dangerous if anyone having access to the server is able to read/edit them.

It is for such situations that Linux has a 3X3 permission system.

There are 3 levels of security for a file :

Read Permission : Permission to read a file (r)

Write Permission : Permission to edit a file (w)

Execute Permission : Permission to execute a file if it is executable (x)

and 3 different levels for a directory :

Enter Permission : Permission to Enter into the Directory

Show Entry : Permission to see the contents of the Directory

Write Entry : Permission to make a new file or subdirectory in the Directory

For granting the above permissions, users are divided into 3 different sets

User : The owner of the file/directory - mostly the person who created the file/directory

Group : Linux users can be divided in groups and one user can be a member of more than one group.

A Group denotes all users who are members of group(s) to which the owner of a file/directory belongs

Others : All users not in the group(s) of the owner.

For eg :

A user level r/w/x permission means only the owner can read, write and execute the file

A group level r/w/x permission means only the members of group(s) to which the owner belongs can read, write and execute the file

An other level r/w/x permission means Everyone can read/write/execute the file.

Below is an example of how a file may be listed when typing (ls -l) at the prompt as well as information on how to interpret it.

-rw-rw-r-- 1 hope 123 Feb 03 15:36 file.txt

-	rw	rw-	r--	1	hope	123	Feb 03 15:36	file.txt
File	owner	group	everyone else	links	owner	size	mod date	file name

The *chmod* Command

The chmod command is used to change the permissions of files/directories in linux. It\\\'s syntax is as follows :

chmod -R/c/f/v [u / g / o / a] [+ / - / =] [rwxXstugo..]

for eg. if u want to give all users in the group of the owner just read permission to a file called foo.txt, the command is

chmod g+r /home/aarjav/foo.txt

here g stands for group, + stands for giving permission (as against - for taking permission away), r stands for read permission. So g+r means ?give group read permission?. All users for the owners group now have read permission to foo.txt

Now if they misbehave and u want to take their read permission away. The command is the same as above, just substituting the + sign with a minus sign

chmod g-r /home/aarjav/foo.txt

As shown the general format of the command is

chmod -R/c/f/v [u / g / o / a] [+ / - / =] [rwxXstugo]

here

u : user
g: group
o : others
a : all
+ : give permission
- : take permission away
= : cause the permissions given to be the only permissions of the file
r : read permission
w: write permission
x : execute permission
X: execute only if it is a directory or already has execute permission for some user
s : set user or group ID on execution
t : save program text on swap device
u : the permissions that the user who owns the file has for it
g : the permissions that the owner?s group has for a file
o : the permissions that users not in the owner?s group have for it

(X, s, t, u, g and o are not required for common tasks)

the initial options -R/c/f/v are explained as follows :

-c : Displays names of only those files whose permissions are being changed

(--changes can also be used instead of -c)

-f : Suppresses display of error messages when a file?s permissions cannot be changed

(--silent of --quiet can also be used instead of -f)

-R: Recursively changes the permission of all files in all subdirectories of the directory whose

permissions are being changed

(--recursive can also be used)

-v : Displays results of all permission changes

(--verbose can also be used)

Numeric Permissions:
chmod can also to attributed by using Numeric Permissions:

400 read by owner
040 read by group
004 read by anybody (other)
200 write by owner
020 write by group
002 write by anybody
100 execute by owner
010 execute by group
001 execute by anybody

Here are some more examples:

chmod o-r pig_info Remove read access from all others.

chmod g+rw pig_info Grant read and write access to group.

chmod ugo+x zippity Grant execute access to everybody.

The chown command

The chown command is used to change the user and/or group which owns one or more files or directories. Its general format is :

chown [-Rcfv] [username][:.][groupname] foo.txt

The flags used above are same as those used in the chmod command . The following are the different ways in which this command can be used :

- The username followed by a dot or colon followed by a groupname changes both the user and group ownerships to those specified.
- The username followed by a dot or colon and no groupname changes the user ownership as specified and changes the group ownership to the specified user?s login group.
- If the colon or dot and groupname are specified without a username, then only the groupownership is changed. This is effectively the same as the chgrp command.
-
- If the username is not followed by a dot or a colon, then only the user ownership is changed.

The chgrp command

The chgrp command is used to change the group ownership of one or more files or directories. Its general syntax is :

chgrp [-Rcfv] groupname foo.txt

The flags used here are also the same as those in the chmod command. The changes in ownership are applied to the groupname and the filename specified.

Experiment 8

Write a pipeline of commands, which displays on the monitor as well as saves the information about the number of users using the system at present on a file called usere.ux.

Many unix commands take text-like input and/or produce text-like output. It's sometimes useful to be able to control where the input comes from and output goes (via redirection), or even pass the output from one command to another's input (via pipes). This can be used to do fairly complex things, for example the following command will print a list of mispelled words in the file fnord.txt (it's sort of a primitive spellchecker):

```
tr 'A-Z' 'a-z' <fnord.txt | tr -cs 'a-z' '\n' | sort | uniq | comm -23 - /usr/share/dict/words
```

What this does is pass the contents of fnord.txt to the `tr` command to translate uppercase letters to lowercase; the output from that is piped to another `tr` command that turns everything except lowercase letters into line breaks (effectively putting each word on a seperate line); that's piped to `sort` which puts the lines (words) in alphabetical order; that's piped to `uniq`, which gets rid of duplicate words; that's piped to `comm`, which compares the words to the dictionary, and prints whatever's not in the dictionary. (Unfortunately, this tends to include plurals, past tense verbs, words that came into use after 1934... it's really not a very good spellchecker.)

Most of the times pipes and redirects are used aren't nearly that complicated. In fact, there are fairly few idioms that get used over and over. I've tried to illustrate them in the examples below.

Note: pipes and redirects are actually done by the shell interpreter you're running. The simple pipes and redirects described here work essentially the same in all shells, but most shells can also perform other, more obscure, kinds of pipes and redirects (e.g. redirecting error messages along with/instead of standard output). For more details, read the `man` page for the shell you're using. BTW, the default shell under OS-X is `tcsh` (except in single-user mode, where it's `zsh`).

> - Redirect output from a command to a file on disk. Note: if the file already exist, it will be erased and overwritten without warning, so be careful.

Example:

```
ps -ax >processes.txt
```

> Use the `ps` command to get a list of processes running on the system, and store the output in a file named processes.txt

>> - Append output from a command to an existing file on disk.

Example:

```
ps -ax >>processes.txt
```

> Tack the current process list onto the end of the file processes.txt

< - Read a command's input from a disk file, rather than the user. Be careful not to type "`>`" by mistake, or you'll erase the contents of the file you're trying to read from.

Example:

```
niload -d -r / . </var/backups/local.nidump
```

Use the `niload` command to load data from the file /var/backups/local.nidumpinfo into the current NetInfo domain. Without input redirection, you'd have to type the NetInfo data into the terminal by hand (and God help you if you got the syntax slightly wrong...)

| - Pass the output of one command to another for further processing.

Examples:

```
ps -ax | grep Finder
```

Use the `ps` command to get a list of processes running on the system, and pass the list to `grep` to search for lines containing "Finder". (Usually, it'll find two: the Finder, and the processes executing `grep Finder`.)

```
lsof | more
```

Use the `lsof` command to list all open files in use on the system, and pass the list to `more` to display it one screen at a time (rather than just spewing the whole thing directly to the terminal).

tee - Used in the middle of a pipeline, this command allows you to both redirect output to a file, *and* pass it to further commands in the pipeline.

Examples:

```
ps -ax | tee processes.txt | more
```

Use the `ps` command to get a list of processes running on the system, store it in the file processes.txt, and also pass it to `more` to display it one screen at a time. Note that you could get the same result with the two commands:

```
ps -ax >processes.txt
more processes.txt
```

Experiment 9

Execute shell commands through vi editor

Introduction

The VI editor is a screen-based editor used by many Unix users. The VI editor has powerful features to aid programmers, but many beginning users avoid using VI because the different features overwhelm them. This tutorial is written to help beginning users get accustomed to using the VI editor, but also contains sections relevant to regular users of VI as well. Examples are provided, and the best way to learn is to try these examples, and think of your own examples as well... There's no better way than to experience things yourself.

EX Commands

Conventions

In this tutorial, the following convention will be used:

^X denotes a control character. For example, if you see: **^d** in the tutorial, that means you hold down the control key and then type the corresponding letter. For this example, you would hold down the **control** key and then type **d**.

Before You Begin

The VI editor uses the full screen, so it needs to know what kind of terminal you have. When you log in, wiliki should ask you what terminal you have. The prompt looks like this:

```
TERM = (vt100)
```

If you know your terminal is a vt100 (or an emulator that can do vt100), just hit return for the terminal type when you log in. If you have an hp terminal, type "`hp`" for the terminal type and hit return. If you are not sure what kind of terminal you have, ask a lab monitor, or have someone help you set the correct terminal type.

If you make an error when you log in and type the wrong terminal type, don't panic and log out. You can type the following commands to fix the settings:

First, tell your shell what type of terminal you have. (If you're not sure what your shell is, type this command to see what shell you have: `echo $SHELL`.) For the examples given, the terminal type is "vt100". Substitute it with whatever terminal type you have. For C shell (/bin/csh), the command is this:

```
set term=vt100
```

For Bourne Shell (/bin/sh) or Korn Shell (/bin/ksh), the commands are the following:

```
export TERM
TERM=vt100
```

Next, reset your terminal with this command:

```
tset
```

Now that the terminal type is (hopefully) correctly set, you are ready to get started with VI.

Starting the VI Editor

The VI editor lets a user create new files or edit existing files. The command to start the VI editor is `vi`, followed by the filename. For example to edit a file called *temporary*, you would type `vi temporary` and then return. You can start VI without a filename, but when you want to save your work, you will have to tell VI which filename to save it into later.

When you start VI for the first time, you will see a screen filled with tildes (A tilde looks like this: ~) on the left side of the screen. Any blank lines beyond the end of the file are shown this way. At the bottom of your screen, the filename should be shown, if you specified an existing file, and the size of the file will be shown as well, like this:

```
"filename" 21 lines, 385 characters
```

If the file you specified does not exist, then it will tell you that it is a new file, like this:

```
"newfile" [New file]
```

If you started VI without a filename, the bottom line of the screen will just be blank when VI starts. If the screen does not show you these expected results, your terminal type may be set wrong. Just type `:q` and return to get out of VI, and fix your terminal type. If you don't know how, ask a lab monitor.

Getting Out of VI

Now that you know how to get into VI, it would be a good idea to know how to get out of it. The VI editor has two modes and in order to get out of VI, you have to be in *command* mode. Hit the key labeled "**Escape**" or "**Esc**" (If your terminal does not have such a key, then try ^[, or control-[.) to get into *command* mode. If you were already in the command mode when you hit "**Escape**", don't worry. It might beep, but you will still be in the *command* mode.

The command to quit out of VI is `:q`. Once in *command* mode, type colon, and 'q', followed by return. If your file has been modified in any way, the editor will warn you of this, and not let you quit. To ignore this message, the command to quit out of VI without saving is `:q!`. This lets you exit VI without saving any of the changes.

Of course, normally in an editor, you would want to save the changes you have made. The command to save the contents of the editor is `:w`. You can combine the above command with the quit command, or `:wq`. You can specify a different file name to save to by specifying the name after the `:w`. For example, if you wanted to save the file you were working as another filename called *filename2*, you would type: `w filename2` and return.

Another way to save your changes and exit out of VI is the `ZZ` command. When in *command* mode, type `ZZ` and it will do the equivalent of `:wq`. If any changes were made to the file, it will be saved. This is the easiest way to leave the editor, with only two keystrokes.

The Two Modes of VI

The first thing most users learn about the VI editor is that it has two modes: *command* and *insert*. The *command* mode allows the entry of commands to manipulate text. These commands are usually one or two characters long, and can be entered with few keystrokes. The *insert* mode puts anything typed on the keyboard into the current file.

VI starts out in *command* mode. There are several commands that put the VI editor into *insert* mode. The most commonly used commands to get into insert mode are `a` and `i`. These two commands are described below. Once you are in *insert* mode, you get out of it by hitting the **escape** key. If your terminal does not have an **escape** key, ^[should work (control-[). You can hit escape two times in a row and VI would definitely be in *command* mode. Hitting **escape** while you are already in *command* mode doesn't take the editor out of *command* mode. It may beep to tell you that you are already in that mode.

How to Type Commands in Command Mode

The command mode commands are normally in this format: (Optional arguments are given in the brackets)

```
[count] command [where]
```

Most commands are one character long, including those which use control characters. The commands described in this section are those which are used most commonly the VI editor.

The *count* is entered as a number beginning with any character from 1 to 9. For example, the `x` command deletes a character under the cursor. If you type `23x` while in *command* mode, it will delete 23 characters.

Some commands use an optional *where* parameter, where you can specify how many lines or how much of the document the command affects, the *where* parameter can also be any command that moves the cursor.

Some Simple VI Commands

Here is a simple set of commands to get a beginning VI user started. There are many other convenient commands, which will be discussed in later sections.

a

enter *insert* mode, the characters typed in will be inserted after the current cursor position. If you specify a count, all the text that had been inserted will be repeated that many times.

h

move the cursor to the left one character position.

i

enter *insert* mode, the characters typed in will be inserted before the current cursor position. If you specify a count, all the text that had been inserted will be repeated that many times.

j

move the cursor down one line.

k

move the cursor up one line.

l

move the cursor to the right one character position.

r

replace one character under the cursor. Specify *count* to replace a number of characters

u

undo the last change to the file. Typing `u` again will re-do the change.

x

delete character under the cursor. *Count* specifies how many characters to delete. The characters will be deleted after the cursor.

Text Buffers in VI

The VI editor has 36 buffers for storing pieces of text, and also a general purpose buffer. Any time a block of text is deleted or yanked from the file, it gets placed into the general purpose buffer. Most users of VI rarely use the other buffers, and can get along without the other buffers. The block of text is also stored in another buffer as well, if it is specified. The buffer is specified using the `"` command. After typing `"`, a letter or digit specifying the buffer must be entered. For example, the command: `"mdd` uses the buffer **m**, and the last two characters stand for delete current line. Similarly, text can be pasted in with the `p` or `P` command. `"mp` pastes the contents of buffer **m** after the current cursor position. For any of the commands used in the next two sections, these buffers can be specified for temporary storage of words or paragraphs.

Cutting and Yanking

The command commonly used command for cutting is `d`. This command deletes text from the file. The command is preceded by an optional *count* and followed by a movement specification. If you double the command by typing `dd`, it deletes the current line. Here are some combinations of these:

d^

deletes from current cursor position to the beginning of the line.

d$

deletes from current cursor position to the end of the line.

dw

deletes from current cursor position to the end of the word.

3dd

deletes three lines from current cursor position downwards.

There is also the `y` command which operates similarly to the `d` command which take text from the file without deleting the text.

Pasting

The commands to paste are `p` and `P`. The only differ in the position relative to the cursor where they paste. `p` pastes the specified or general buffer after the cursor position, while `P` pastes the specified or general buffer before the cursor position. Specifying *count* before the paste command pastes text the specified number of times.

Indenting Your Code and Checking

The VI editor has features to help programmers format their code neatly. There is a variable that to set up the indentation for each level of nesting in code. In order to set it up, see the customization section of this tutorial. For example, the command to set the shift width to 4 characters is `:set sw=4`.

The following commands indent your lines or remove the indentation, and can be specified with *count*:

<<

Shifts the current line to the left by one shift width.

>>

Shifts the current line to the right by one shift width.

The VI editor also has a helpful feature which checks your source code for any hanging parentheses or braces. The `%` command will look for the left parenthesis or brace corresponding to a particular right parenthesis or brace and vice versa. Place the cursor onto a parenthesis or brace and type **%** to move the cursor to the corresponding parenthesis or brace. This is useful to check for unclosed parentheses or braces. If a parenthesis or brace exists without a matching parenthesis or brace, VI will beep at you to indicate that no matching symbol was found.

Word and Character Searching

The VI editor has two kinds of searches: string and character. For a string search, the `/` and `?` commands are used. When you start these commands, the command just typed will be shown on the bottom line, where you type the particular string to look for. These two commands differ only in the direction where the search takes place. The `/` command searches forwards (downwards) in the file, while the `?` command searches backwards (upwards) in the file. The `n` and `N` commands repeat the previous search command in the same or opposite direction, respectively. Some characters have special meanings to VI, so they must be preceded by a backslash (\) to be included as part of the search expression.

Special characters:

^

Beginning of the line. (At the beginning of a search expression.)

.

Matches a single character.

*

Matches zero or more of the previous character.

$

End of the line (At the end of the search expression.)

[

Starts a set of matching, or non-matching expressions... For example: /f[iae]t matches either of these: fit fat fet In this form, it matches anything except these: /a[^bcd] will not match any of these, but anything with an a and another letter: ab ac ad

<

Put in an expression escaped with the backslash to find the ending or beginning of a word. For example: /\<the\> should find only word the, but not words like these: there and other.

>

See the '<' character description above.

The character search searches within one line to find a character entered after the command. The `f` and `F` commands search for a character on the current line only. `f` searches forwards and `F` searches backwards and the cursor moves to the position of the found character.

The `t` and `T` commands search for a character on the current line only, but for `t`, the cursor moves to the position before the character, and `T` searches the line backwards to the position after the character.

These two sets of commands can be repeated using the `;` or `,` command, where `;` repeats the last character search command in the same direction, while `,` repeats the command in the reverse direction.

Settings for VI (and EX)

You can customize the way VI behaves upon start up. There are several edit options which are available using the :set command, these are the VI and EX editor options available on Wiliki: (You can get this list by typing `:set all` and then **return** in command mode)

```
noautoindent        magic                              noshowmatch
autoprint           mesg                               noshowmode
noautowrite         nomodelines                        noslowopen
nobeautify          nonumber                           tabstop=8
directory=/tmp      nonovice                           taglength=0
nodoubleescape      nooptimize                         tags=tags /usr/lib/tags
noedcompatible      paragraphs=IPLPPPQPP LIpplpipnpbp   term=xterm
noerrorbells        prompt                             noterse
noexrc              noreadonly                         timeout
flash               redraw                             timeoutlen=500
hardtabs=8          remap                              ttytype=xterm
noignorecase        report=5                           warn
keyboardedit        scroll=11                          window=23
keyboardedit!       sections=NHSHH HUuhsh+c            wrapscan
nolisp              shell=/bin/csh                     wrapmargin=0
nolist              shiftwidth=8                       nowriteany
```

Some of these options have values set with the equals sign '=' in it, while others are either set or not set. (These on or off type of options are called **Boolean**, and have "no" in front of them to indicate that they are not set.) The options shown here are the options that are set without any customization. Descriptions of some of these are given below, with an abbreviation. For example, the command set autoindent, you can type `:set autoindent` or `:set ai`. To unset it, you can type `:set noautoindent` or `:set noai`.

autoindent (ai)

> This option sets the editor so that lines following an indented line will have the same indentation as the previous line. If you want to back over this indentation, you can type `^D` at the very first character position. This `^D` works in the *insert* mode, and not in *command* mode. Also, the width of the indentations can be set with **shiftwidth**, explained below.

exrc

> The *.exrc* file in the current directory is read during startup. This has to be set either in the environment variable **EXINIT** or in the *.exrc* file in your home directory.

mesg

> Turn off messages if this option is unset using `:set nomesg`, so that nobody can bother you while using the editor.

number (nu)

> Displays lines with line numbers on the left side.

shiftwidth (sw)

> This option takes a value, and determines the width of a software tabstop. (The software tabstop is used for the << and >> commands.) For example, you would set a shift width of 4 with this command: `:set sw=4`.

showmode (smd)

> This option is used to show the actual mode of the editor that you are in. If you are in *insert* mode, the bottom line of the screen will say **INPUT MODE**.

warn

This option warns you if you have modified the file, but haven't saved it yet.

window (wi)

This option sets up the number of lines on the window that VI uses. For example, to set the VI editor to use only 12 lines of your screen (because your modem is slow) you would use this: `:set wi=12`.

wrapscan (ws)

This option affects the behavior of the word search. If `wrapscan` is set, if the word is not found at the bottom of the file, it will try to search for it at the beginning.

wrapmargin (wm)

If this option has a value greater than zero, the editor will automatically "word wrap". That is, if you get to within that many spaces of the left margin, the word will wrap to the next line, without having to type return. For example, to set the wrap margin to two characters, you would type this: `:set wm=2`.

Abbreviations and Mapping Keys to Other Keys

One EX editor command that is useful in the VI editor is the **abbreviate** command. This lets you set up abbreviations for specific strings. The command looks like this: `:ab string thing to substitute for`. For example, if you had to type the name, "**Humuhumunukunukuapua`a**" but you didn't want to type the whole name, you could use an abbreviation for it. For this example, the command is entered like this:

```
:ab 9u Humuhumunukunukuapua`a
```

Now, whenever you type `9u` as a separate word, VI will type in the entire word(s) specified. If you typed in 9university, it will not substitute the word.

To remove a previously defined abbreviation, the command is unabbreviate. To remove the previous example, the command would be ":una 9u" To get your listing of abbreviations, simply just type :ab without any definitions.

Another EX editor command that is useful for customization is the mapping command. There are two kinds of mapping commands. One for command mode, and the other for insert mode. These two commands are `:map` and `:map!` respectively. The mapping works similarly to the abbreviation, and you give it a key sequence and give it another key sequence to substitute it with. (The substituted key sequences are usually VI commands.)

The EXINIT Environment Variable and the **.exrc** *file*

There are two ways to customize the VI editor. If you create a file called *.exrc* in your home directory, all the commands in there will be read when VI starts up. The other method is to set an environment variable called **EXINIT**. The options will be set in your shell's setup file. If you use /bin/csh (C-Shell), the command is as follows, and is put in the *.cshrc* file:

```
setenv EXINIT '...'
```

If you use /bin/sh or /bin/ksh, the command is as follows, and is put into the *.profile* file:

```
export EXINIT
EXINIT='...'
```

Don't put in ... as the example says. In this space put the commands that you want to set up. For example, if you want to have auto indent, line numbering, and the wrap margin of three characters, then the setenv command (for C shell) looks like this:

```
setenv EXINIT 'set ai nu wm=3'
```

If you want to put more than one command in the `setenv EXINIT` thing, separate the commands with a vertical bar (|). For example, to map the 'g' command to the 'G' character in command mode, the command is `:map g G`, and combined with the above command, you get this:

```
setenv EXINIT 'set ai nu wm=3|map g G'
```

If you want to create the file called *.exrc*, you can put exactly the same things in the file as shown in the quotes after the **EXINIT**.

Recovering Your Work When Something Goes Wrong with Your Terminal

The VI editor edits a temporary copy of your file, and after the editing is complete, or when you tell it to save, it puts the contents of the temporary copy into the original file. If something goes wrong while you are editing your file, the VI editor will attempt to save whatever work you had in progress, and store it for later recovery. (Note: If VI dies while you were working on any file, it sends you an email message on how to recover it. The **-r** option stands for recovery. If you were editing the file *vitalinfo*, and you accidentally got logged out, then the **-r** option of the 'vi' editor should help. The command would look somewhat like this: `vi -r vitalinfo` After using the **-r** option once, though, you MUST save what you have recovered to the actual file... The **-r** option only works once per failed VI session.

Warning About Using VI on the Workstations

There are two things to be aware of when using the workstations: Editing the same file many times at once, and changing the size of the screen.

Because VI edits a copy of your original file and saves the contents of that copy into the original file, if you are logged on more than once and are editing the same file more than once using VI, if you save on one window and then you save on the other window, the changes made to the file on the first save would be overwritten. Make sure that you only run one copy of VI per file.

If you use a terminal program from a workstation, you can change the size of the screen by dragging the sides of the window. If the size is not working properly, the command to type is this:

```
    eval `resize`
```

If that doesn't work the command would be this:

```
    eval `/usr/bin/X11/resize`
```

If the size is wrong, the editor will not operate correctly. If you have any problems with the screen size, ask the monitors in the computer lab for help setting the sizes correctly.

Summary of VI commands

This list is a summary of VI commands, categorized by function. There may be other commands available, so check the on-line manual on VI. For easy reference, you can save this file as text and delete any commands you don't think you would use and print out the resulting shorter file.

Cutting and Pasting/Deleting text

"
: Specify a buffer to be used any of the commands using buffers. Follow the " with a letter or a number, which corresponds to a buffer.

D
: Delete to the end of the line from the current cursor position.

P
: Paste the specified buffer before the current cursor position or line. If no buffer is specified (with the " command.) then 'P' uses the general buffer.

X

Delete the character before the cursor.

Y

Yank the current line into the specified buffer. If no buffer is specified, then the general buffer is used.

d

Delete until *where*. "dd" deletes the current line. A count deletes that many lines. Whatever is deleted is placed into the buffer specified with the " command. If no buffer is specified, then the general buffer is used.

p

Paste the specified buffer after the current cursor position or line. If no buffer is specified (with the " command.) then 'p' uses the general buffer.

x

Delete character under the cursor. A count tells how many characters to delete. The characters will be deleted after the cursor.

y

Yank until , putting the result into a buffer. "yy" yanks the current line. a count yanks that many lines. The buffer can be specified with the " command. If no buffer is specified, then the general buffer is used.

Inserting New Text

A

Append at the end of the current line.

I

Insert from the beginning of a line.

O

(letter oh) Enter *insert* mode in a new line above the current cursor position.

a

Enter *insert* mode, the characters typed in will be inserted after the current cursor position. A count inserts all the text that had been inserted that many times.

i

Enter *insert* mode, the characters typed in will be inserted before the current cursor position. A count inserts all the text that had been inserted that many times.

o

Enter *insert* mode in a new line below the current cursor position.

Moving the Cursor Within the File

^B

Scroll backwards one page. A count scrolls that many pages.

^D

Scroll forwards half a window. A count scrolls that many lines.

^F

Scroll forwards one page. A count scrolls that many pages.

^H

Move the cursor one space to the left. A count moves that many spaces.

^J

Move the cursor down one line in the same column. A count moves that many lines down.

^M

Move to the first character on the next line.

^N

Move the cursor down one line in the same column. A count moves that many lines down.

^P

Move the cursor up one line in the same column. A count moves that many lines up.

^U

Scroll backwards half a window. A count scrolls that many lines.

$

Move the cursor to the end of the current line. A count moves to the end of the following lines.

%

Move the cursor to the matching parenthesis or brace.

^

Move the cursor to the first non-whitespace character.

(
: Move the cursor to the beginning of a sentence.

)
: Move the cursor to the beginning of the next sentence.

{
: Move the cursor to the preceding paragraph.

}
: Move the cursor to the next paragraph.

|
: Move the cursor to the column specified by the count.

+
: Move the cursor to the first non-whitespace character in the next line.

-
: Move the cursor to the first non-whitespace character in the previous line.

_
: Move the cursor to the first non-whitespace character in the current line.

0
: (Zero) Move the cursor to the first column of the current line.

B
: Move the cursor back one word, skipping over punctuation.

E
: Move forward to the end of a word, skipping over punctuation.

G
: Go to the line number specified as the count. If no count is given, then go to the end of the file.

H
: Move the cursor to the first non-whitespace character on the top of the screen.

L
: Move the cursor to the first non-whitespace character on the bottom of the screen.

M
: Move the cursor to the first non-whitespace character on the middle of the screen.

W
: Move forward to the beginning of a word, skipping over punctuation.

b
: Move the cursor back one word. If the cursor is in the middle of a word, move the cursor to the first character of that word.

e
: Move the cursor forward one word. If the cursor is in the middle of a word, move the cursor to the last character of that word.

h
: Move the cursor to the left one character position.

j
: Move the cursor down one line.

k
: Move the cursor up one line.

l
: Move the cursor to the right one character position.

w
: Move the cursor forward one word. If the cursor is in the middle of a word, move the cursor to the first character of the next word.

Moving the Cursor Around the Screen

^E
: Scroll forwards one line. A count scrolls that many lines.

^Y
: Scroll backwards one line. A count scrolls that many lines.

z

Redraw the screen with the following options. "z<return>" puts the current line on the top of the screen; "z." puts the current line on the center of the screen; and "z-" puts the current line on the bottom of the screen. If you specify a count before the 'z' command, it changes the current line to the line specified. For example, "16z." puts line 16 on the center of the screen.

Replacing Text

C

Change to the end of the line from the current cursor position.

R

Replace characters on the screen with a set of characters entered, ending with the Escape key.

S

Change an entire line.

c

Change until . "cc" changes the current line. A count changes that many lines.

r

Replace one character under the cursor. Specify a count to replace a number of characters.

s

Substitute one character under the cursor, and go into insert mode. Specify a count to substitute a number of characters. A dollar sign ($) will be put at the last character to be substituted.

Searching for Text or Characters

,

Repeat the last f, F, t or T command in the reverse direction.

/

Search the file downwards for the string specified after the /.

;

Repeat the last f, F, t or T command.

?

Search the file upwards for the string specified after the ?.

F

Search the current line backwards for the character specified after the 'F' command. If found, move the cursor to the position.

N

Repeat the last search given by '/' or '?', except in the reverse direction.

T

Search the current line backwards for the character specified after the 'T' command, and move to the column after the if it's found.

f

Search the current line for the character specified after the 'f' command. If found, move the cursor to the position.

n

Repeat last search given by '/' or '?'.

t

Search the current line for the character specified after the 't' command, and move to the column before the character if it's found.

Manipulating Character/Line Formatting

~

Switch the case of the character under the cursor.

<

Shift the lines up to *where* to the left by one shiftwidth. "<<" shifts the current line to the left, and can be specified with a count.

>

Shift the lines up to *where* to the right by one shiftwidth. ">>" shifts the current line to the right, and can be specified with a count.

J

Join the current line with the next one. A count joins that many lines.

Saving and Quitting

^\

Quit out of "VI" mode and go into "EX" mode. The EX editor is the line editor VI is build upon. The EX command to get back into VI is ":vi".

Q

Quit out of "VI" mode and go into "EX" mode. The ex editor is a line-by-line editor. The EX command to get back into VI is ":vi".

ZZ

Exit the editor, saving if any changes were made.

Miscellany

^G

Show the current filename and the status.

^L

Clear and redraw the screen.

^R

Redraw the screen removing false lines.

^[

Escape key. Cancels partially formed command.

^^

Go back to the last file edited.

!

Execute a shell. If a is specified, the program which is executed using ! uses the specified line(s) as standard input, and will replace those lines with the standard output of the program executed. "!!" executes a program using the current line as input. For example, "!4jsort" will take five lines from the current cursor position and execute sort. After typing the command, there will be a single exclamation point where you can type the command in.

&

Repeat the previous ":s" command.

.

Repeat the last command that modified the file.

:

Begin typing an EX editor command. The command is executed once the user types return. (See section below.)

@

Type the command stored in the specified buffer.

U

Restore the current line to the state it was in before the cursor entered the line.

m

Mark the current position with the character specified after the 'm' command.

u

Undo the last change to the file. Typing 'u' again will re-do the change.

EX Commands

The VI editor is built upon another editor, called EX. The EX editor only edits by line. From the VI editor you use the : command to start entering an EX command. This list given here is not complete, but the commands given are the more commonly used. If more than one line is to be modified by certain commands (such as ":s" and ":w") the range must be specified before the command. For example, to substitute lines 3 through 15, the command is ":3,15s/from/this/g".

:ab string strings

Abbreviation. If a word is typed in VI corresponding to string1, the editor automatically inserts the corresponding words. For example, the abbreviation ":ab usa United States of America" would insert the words, "United States of America" whenever the word "usa" is typed in.

:map keys new_seq

Mapping. This lets you map a key or a sequence of keys to another key or a sequence of keys.

:q

Quit VI. If there have been changes made, the editor will issue a warning message.

:q!
: Quit VI without saving changes.

:s/*pattern*/*to_pattern*/*options*
: Substitute. This substitutes the specified pattern with the string in the to_pattern. Without options, it only substitutes the first occurence of the pattern. If a 'g' is specified, then all occurences are substituted. For example, the command ":1,$s/Dwayne/Dwight/g" substitutes all occurences of "Dwayne" to "Dwight".

:set [all]
: Sets some customizing options to VI and EX. The ":set all" command gives all the possible options. (See the section on customizing VI for some options.)

:una string
: Removes the abbreviation previously defined by ":ab".

:unm keys
: Removes the remove mapping defined by ":map".

:vi filename
: Starts editing a new file. If changes have not been saved, the editor will give you a warning.

:w
: Write out the current file.

:w filename
: Write the buffer to the filename specified.

:w >> filename
: Append the contents of the buffer to the filename.

:wq
: Write the buffer and quit.

Experiment No. 10

Write a shell script that accept any number of argument and prints them in the reverse order.

```
[~/temp]$ cat deepak.sh

#! /bin/sh


# reverse a string


[[ -z "$@" ]] && STR=" a b c d e f g h i j k l m n o p q r s t u v w x y z " ||
STR="$@"


len=${#STR}


REV=""

for (( i=$len ; i>0 ; i-- ))

do

        REV=$REV""${STR:$i-1:$i}

        STR=${STR%${STR:$i-1:$i}}

done


echo "Reversed string"




echo $REV
```

Experiment No.11

Write a shell script to find the smallest of three numbers that are read from the keyboard.

```
echo first_num
read first_num
echo second_num
read second_num
echo third_num
read third_num

if (first_num>second_num) then
if (first_num>third_num) then
echo $first_num is the biggest
else
echo $third_num is the biggest
elif
(second_num>third_num) then
echo $second_num is the biggest
else
echo $third_num is the biggest
```

Experiment No.12

Write a shell script that report the logging in of a specified user within one minute after he/she logs in. The script automatically terminates if the specified user does not login

During a specifiedperiod of time.

```
#! /bin/sh

echo "The current users are:"
who | awk '{print $1}' | sort > temp1
cp temp1 temp2
more temp1

while true
do
    who | awk '{print $1}' | sort > temp2
    cmp -s temp1 temp2

    case "$?" in


   0)
        echo "No user has logged in/out in the last 5 seconds."
        ;;
         1)
        user=`comm -23 temp1 temp2`
        file=`grep $user temp1 temp2 | cut -c 1-5`

        if [ $file = "temp1" ]
            echo "User "$user" has logged out."


        if [ $file = "temp2" ]
            echo "User "$user" has logged in."
        ;;
    esac

    rm temp1
    mv temp2 temp1

    sleep 5
done
```

Experiment.13

Write a shell script that reports the logging in of a specified user within one minute after he/she logs in. The script automatically terminates if the specified user does not login during a specified period of time

```
echo ‘Enter the login name of the user:”
read user
period=0
while [ true]
do
var=`who | grep –w “$user”`
len=`echo “$var | wc –c`
if [ $len –gt 1 ]
then
echo “$user logged in $tm seconds”
exit
else
sleep 1
tm=`expr $tm + 1`
fi
if [ $tm –eq 61 ]
then
echo “$user did not login within 1 minute”
exit
fi
done
```

Output :

```
Enter the login name of the user :mca219
mca219 logged in 25 seconds
Enter the login name of the user :mca250
mca250 did not login within 1 minute
```

Description

sleep command : Using this command the user can make the system sleep, that is , pause for some fixed period of time.

Experiment No.14

Study of installation of SAMBA, APACHE,TOMCAT.

Configuring Samba (smb.conf)

Samba's configuration is stored in the `smb.conf` file, which usually resides in `/etc/samba/smb.conf` or `/usr/local/samba/lib/smb.conf`. You can either edit this file yourself or do it using one of the many graphical tools that are available, such as the Web-based interface SWAT, that is included with Samba.

Configuration File Syntax

The `smb.conf` file uses the same syntax as the various old `.ini` files in Windows 3.1: Each file consists of various sections, which are started by putting the section name between brackets (`[]`) on a new line. Each contains zero or more key/value pairs separated by an equality sign (=). The file is just a plaintext file, so you can open and edit it with your favorite editing tool.

Each section in the `smb.conf` file represents either a share or a meta-service on the Samba server. The section `[global]` is special, since it contains settings that apply to the whole Samba server. Samba supports a number of meta-services, each of which serves its own purpose. For example, the `[homes]` share is a meta-service that causes Samba to provide a personal home share for each user. The `[printers]` share is a meta-service that establishes print queue support and that specifies the location of the intermediate spool directory into which print jobs are received from Windows clients prior to being dispatched to the UNIX/Linux print spooler.

The `printers` meta-service will cause every printer that is either specified in a `printcap` file, via the `lpstat`, or via the CUPS API, to be published as a shared print queue. The `printers` stanza in the `smb.conf` file can be set as not browseable. If it is set to be browseable, then it will be visible as if it is a share. That makes no sense given that this meta-service is responsible only for making UNIX system printers available as Windows print queues. If a `comment` parameter is specified, the value of it will be displayed as part of the printer name in Windows Explorer browse lists.

Each section of the `smb.conf` file that specifies a share, or a meta-service, is called a stanza. The `global` stanza specifies settings that affect all the other stanzas in the `smb.conf` file. Configuration parameters are documented in the `smb.conf` man page. Some parameters can be used only in the `global` stanza, some only in share or meta-service stanzas, and some can be used globally or just within a share or meta-service stanza.

A minimal smb.conf contains a very minimal `smb.conf`.

Example 1.1. A minimal smb.conf

```
[global]
workgroup = WKG
netbios name = MYNAME

[share1]
path = /tmp

[share2]
path = /my_shared_folder
comment = Some random files
```

TDB Database File Information

This section contains brief descriptions of the databases that are used by Samba-3.

The directory in which Samba stores the tdb files is determined by compile-time directives. Samba-3 stores tdb files in two locations. The best way to determine these locations is to execute the following command:

```
root#  smbd -b | grep PRIVATE_DIR
   PRIVATE_DIR: /etc/samba/private
```

This means that the confidential tdb files are stored in the `/etc/samba/private` directory. Samba-3 also uses a number of tdb files that contain more mundane data. The location of these files can be found by executing:

```
root#  smbd -b | grep LOCKDIR
   LOCKDIR: /var/lib/samba
```

Therefore the remaining control files will, in the example shown, be stored in the `/var/lib/samba` directory.

The persistent tdb files are described in the Persistent TDB File Descriptions table. All persistent tdb files should be regularly backed up. Use the `tdbbackup` utility to backup the tdb files. All persistent tdb files must be preserved during machine migrations, updates and upgrades.

The temporary tdb files do not need to be backed up, nor do they need to be preseved across machine migrations, updates or upgrades. The temporary tdb files are described in the Temporary TDB File Descriptions.

Table 1.1. Persistent TDB File Descriptions

Name	Description
account_policy	Samba/NT account policy settings, includes password expiration settings.
group_mapping	Mapping table from Windows groups/SID to UNIX groups.
ntdrivers	Stores per-printer installed driver information.
ntforms	Stores per-printer installed forms information.
ntprinters	Stores the per-printer devmode configuration settings.
passdb	Exists only when the tdbsam passwd backend is used. This file stores the SambaSAMAccount information. Note: This file requires that user POSIX account information is available from either the /etc/passwd file, or from an alternative system source.
registry	Read-only Samba database of a Windows registry skeleton that provides support for exporting various database tables via the winreg RPCs.
secrets	This file stores the Workgroup/Domain/Machine SID, the LDAP directory update password, and a further collection of critical environmental data that is necessary for Samba to operate correctly. This file contains very sensitive information that must be protected. It is stored in the PRIVATE_DIR directory.
share_info	Stores per-share ACL information.
winbindd_idmap	Winbindd's local IDMAP database.

Table 1.2. Temporary TDB File Descriptions

Name	Description	Backup
brlock	Byte-range locking information.	No
connections	A temporary cache for current connection information used to enforce max connections.	no
eventlog/*tdb	Records of eventlog entries. In most circumstances this is just a cache of system logs.	no
gencache	Generic caching database for dead WINS servers and trusted domain data.	no
login_cache	A temporary cache for login information, in particular bad password attempts.	no
messages	Temporary storage of messages being processed by smbd.	no
netsamlogon_cache	Caches user net_info_3 structure data from net_samlogon requests (as a domain member).	no
perfmon/*.tdb	Performance counter information.	no
printing/*.tdb	Cached output from lpq command created on a per-print-service basis.	no
schannel_store	A confidential file, stored in the PRIVATE_DIR, containing crytographic connection information so that clients that have temporarily disconnected can reconnect without needing to renegotiate the connection setup process.	no
sessionid	Temporary cache for miscellaneous session information and for utmp handling.	no
unexpected	Stores packets received for which no process is actively listening.	no
winbindd_cache	Cache of Identity information received from an NT4 domain or from ADS. Includes user lists, etc.	yes

Starting Samba

Samba essentially consists of two or three daemons. A daemon is a UNIX application that runs in the background and provides services. An example of a service is the Apache Web server for which the daemon is called `httpd`. In the case of Samba there are three daemons, two of which are needed as a minimum.

The Samba server is made up of the following daemons:

nmbd

This daemon handles all name registration and resolution requests. It is the primary vehicle involved in network browsing. It handles all UDP-based protocols. The `nmbd` daemon should be the first command started as part of the Samba startup process.

smbd

This daemon handles all TCP/IP-based connection services for file- and print-based operations. It also manages local authentication. It should be started immediately following the startup of `nmbd`.

This daemon should be started when Samba is a member of a Windows NT4 or ADS domain. It is also needed when Samba has trust relationships with another domain. The `winbindd` daemon will check the `smb.conf` file for the presence of the *`idmap uid`* and *`idmap gid`* parameters. If they are are found, `winbindd` will use the values specified for for UID and GID allocation. If these parameters are not specified, `winbindd` will start but it will not be able to allocate UIDs or GIDs.

When Samba has been packaged by an operating system vendor, the startup process is typically a custom feature of its integration into the platform as a whole. Please refer to your operating system platform administration manuals for specific information pertaining to correct management of Samba startup.

Example Configuration

There are sample configuration files in the examples subdirectory in the source code distribution tarball package. It is suggested you read them carefully so you can see how the options go together in practice. See the man page for all the options. It might be worthwhile to start out with the `smb.conf.default` configuration file and adapt it to your needs. It contains plenty of comments.

The simplest useful configuration file would contain something like that shown in Another simple smb.conf File.

Example 1.2. Another simple smb.conf File

[global]
workgroup = MIDEARTH

[homes]
guest ok = no
read only = no

This will allow connections by anyone with an account on the server, using either their login name or *`homes`* as the service name. (Note: The workgroup that Samba should appear in must also be set. The default workgroup name is WORKGROUP.)

Make sure you put the `smb.conf` file in the correct place. Note, the correct location of this file depends on how the binary files were built. You can discover the correct location by executing from the directory that contains the `smbd` command file:

```
root# smbd -b | grep smb.conf
```

For more information about security settings for the *`[homes]`* share, please refer to Securing Samba.

Test Your Config File with testparm

It's important to validate the contents of the `smb.conf` file using the testparm program. If testparm runs correctly, it will list the loaded services. If not, it will give an error message. Make sure it runs correctly and that the services look reasonable before proceeding. Enter the command:

```
root# testparm /etc/samba/smb.conf
```

Testparm will parse your configuration file and report any unknown parameters or incorrect syntax. It also performs a check for common misconfigurations and will issue a warning if one is found.

Always run testparm again whenever the `smb.conf` file is changed!

The `smb.conf` file is constantly checked by the Samba daemons `smbd` and every instance of itself that it spawns, `nmbd` and `winbindd`. It is good practice to keep this file as small as possible. Many administrators prefer to document Samba configuration settings and thus the need to keep this file small goes against good documentation wisdom. One solution that may be adopted is to do all documentation and configuration in a file that has another name, such as `smb.conf.master`. The `testparm` utility can be used to generate a fully optimized `smb.conf` file from this master configuration and documentation file as shown here:

```
root# testparm -s smb.conf.master > smb.conf
```

This administrative method makes it possible to maintain detailed configuration change records while at the same time keeping the working `smb.conf` file size to the minimum necessary.

SWAT

SWAT is a Web-based interface that can be used to facilitate the configuration of Samba. SWAT might not be available in the Samba package that shipped with your platform, but in a separate package. If you need to build SWAT please read the SWAT man page regarding compilation, installation, and configuration of SWAT from the source code.

To launch SWAT, just run your favorite Web browser and point it to http://localhost:901/. Replace *`localhost`* with the name of the computer on which Samba is running if that is a different computer than your browser.

SWAT can be used from a browser on any IP-connected machine, but be aware that connecting from a remote machine leaves your connection open to password sniffing because passwords will be sent over the wire in the clear.

Please note that re-writing the configuration file using SWAT will remove all comments! More information about SWAT can be found in The Samba Web Administration Tool.

List Shares Available on the Server

To list shares that are available from the configured Samba server, execute the following command:

```
$ smbclient -L yourhostname
```

You should see a list of shares available on your server. If you do not, then something is incorrectly configured. This method can also be used to see what shares are available on other SMB servers, such as Windows 2000.

you choose user-level security, you may find that Samba requests a password before it will list the shares. See the `smbclient` man page for details. You can force it to list the shares without a password by adding the option `-N` to the command line.

Connect with a UNIX Client

Enter the following command:

```
$ smbclient //yourhostname/aservice
```

Typically *yourhostname* is the name of the host on which smbd has been installed. The *aservice* is any service that has been defined in the `smb.conf` file. Try your username if you just have a *[homes]* section in the `smb.conf` file.

Example: If the UNIX host is called *bambi* and a valid login name is *fred*, you would type:

```
$ smbclient //bambi/fred
```

Connect from a Remote SMB Client

Now that Samba is working correctly locally, you can try to access it from other clients. Within a few minutes, the Samba host should be listed in the Network Neighborhood on all Windows clients of its subnet. Try browsing the server from another client or "mounting" it.

Mounting disks from a DOS, Windows, or OS/2 client can be done by running a command such as:

```
C:\> net use m: \\servername\service
```

Where the drive letter m: is any available drive letter. It is important to double-check that the service (share) name that you used does actually exist.

Try printing, for example,

```
C:\> net use lpt1:  \\servername\spoolservice
```

The `spoolservice` is the name of the printer (actually the print queue) on the target server. This will permit all print jobs that are captured by the lpt1: port on the Windows client to be sent to the printer that owns the spoolservice that has been specified.

```
C:\> print filename
```

Installation of APACHE

The following requirements exist for building Apache:

Disk Space

Make sure you have at least 50 MB of temporary free disk space available. After installation Apache occupies approximately 10 MB of disk space. The actual disk space requirements will vary considerably based on your chosen configuration options and any third-party modules.

ANSI-C Compiler and Build System

Make sure you have an ANSI-C compiler installed. The GNU C compiler (GCC) from the Free Software Foundation (FSF) is recommended (version 2.7.2 is fine). If you don't have GCC then at least make sure your vendor's compiler is ANSI compliant. In addition, your PATH must contain basic build tools such as make.

Accurate time keeping

Elements of the HTTP protocol are expressed as the time of day. So, it's time to investigate setting some time synchronization facility on your system. Usually the ntpdate or xntpd programs are used for this purpose which are based on the Network Time Protocol (NTP). See the Usenet newsgroup comp.protocols.time.ntp and the NTP homepage for more details about NTP software and public time servers.

Perl 5 [OPTIONAL]

For some of the support scripts like apxs or dbmmanage (which are written in Perl) the Perl 5 interpreter is required (versions 5.003 or newer are sufficient). If you have multiple Perl interpreters (for example, a systemwide install of Perl 4, and your own install of Perl 5), you are advised to use the --with-perl option (see below) to make sure the correct one is used by configure. If no Perl 5 interpreter is found by the configure script, you will not be able to use the affected support scripts. Of course, you will still be able to build and use Apache 2.0.

Download

Apache can be downloaded from the Apache HTTP Server download site which lists several mirrors. Most users of Apache on unix-like systems will be better off downloading and compiling a source version. The build process (described below) is easy, and it allows you to customize your server to suit your needs. In addition, binary releases are often not up to date with the latest source releases. If you do download a binary, follow the instructions in the INSTALL.bindist file inside the distribution.

After downloading, it is important to verify that you have a complete and unmodified version of the Apache HTTP Server. This can be accomplished by testing the downloaded tarball against the PGP signature. Details on how to do this are available on the download page and an extended example is available describing the use of PGP.

Extract

Extracting the source from the Apache HTTPD tarball is a simple matter of uncompressing, and then untarring:

```
$ gzip -d httpd-2_0_NN.tar.gz
$ tar xvf httpd-2_0_NN.tar
```

This will create a new directory under the current directory containing the source code for the distribution. You should cd into that directory before proceeding with compiling the server.

Configuring the source tree

The next step is to configure the Apache source tree for your particular platform and personal requirements. This is done using the script configure included in the root directory of the distribution. (Developers downloading the CVS version of the Apache source tree will need to have autoconf and libtool installed and will need to run buildconf before proceeding with the next steps. This is not necessary for official releases.)

To configure the source tree using all the default options, simply type ./configure. To change the default options, configure accepts a variety of variables and command line options.

The most important option is the location --prefix where Apache is to be installed later, because Apache has to be configured for this location to work correctly. More fine-tuned control of the location of files is possible with additional configure options.

Also at this point, you can specify which features you want included in Apache by enabling and disabling modules. Apache comes with a Base set of modules included by default. Other modules are enabled using the --enable-*module* option, where *module* is the name of the module with the mod_ string removed and with any underscore converted to a dash.

You can also choose to compile modules as shared objects (DSOs) -- which can be loaded or unloaded at runtime -- by using the option --enable-*module*=shared. Similarly, you can disable Base modules with the --disable-*module* option. Be careful when using these options, since configure cannot warn you if the module you specify does not exist; it will simply ignore the option.

In addition, it is sometimes necessary to provide the configure script with extra information about the location of your compiler, libraries, or header files. This is done by passing either environment variables or command line options to configure. For more information, see the configure manual page.

For a short impression of what possibilities you have, here is a typical example which compiles Apache for the installation tree /sw/pkg/apache with a particular compiler and flags plus the two additional modules mod_rewrite and mod_speling for later loading through the DSO mechanism:

```
$ CC="pgcc" CFLAGS="-O2" \
./configure --prefix=/sw/pkg/apache \
--enable-rewrite=shared \
--enable-speling=shared
```

When configure is run it will take several minutes to test for the availability of features on your system and build Makefiles which will later be used to compile the server.

Details on all the different configure options are available on the configure manual page.

Build

Now you can build the various parts which form the Apache package by simply running the command:

```
$ make
```

Please be patient here, since a base configuration takes approximately 3 minutes to compile under a Pentium III/Linux 2.2 system, but this will vary widely depending on your hardware and the number of modules which you have enabled.

Install

Now it's time to install the package under the configured installation *PREFIX* (see --prefix option above) by running:

```
$ make install
```

If you are upgrading, the installation will not overwrite your configuration files or documents.

Customize

Next, you can customize your Apache HTTP server by editing the configuration files under *PREFIX*/conf/.

```
$ vi PREFIX/conf/httpd.conf
```

Have a look at the Apache manual under docs/manual/ or consult http://httpd.apache.org/docs/2.0/ for the most recent version of this manual and a complete reference of available configuration directives.

Test

Now you can start your Apache HTTP server by immediately running:

```
$ PREFIX/bin/apachectl start
```

and then you should be able to request your first document via URL http://localhost/. The web page you see is located under the DocumentRoot which will usually be *PREFIX*/htdocs/. Then stop the server again by running:

```
$ PREFIX/bin/apachectl stop
```

Upgrading

The first step in upgrading is to read the release announcement and the file CHANGES in the source distribution to find any changes that may affect your site. When changing between major releases (for example, from 1.3 to 2.0 or from 2.0 to 2.2), there will likely be major differences in the compile-time and run-time configuration that will require manual adjustments. All modules will also need to be upgraded to accomodate changes in the module API.

Upgrading from one minor version to the next (for example, from 2.0.55 to 2.0.57) is easier. The make install process will not overwrite any of your existing documents, log files, or configuration files. In addition, the developers make every effort to avoid incompatible changes in the configure options, run-time configuration, or the module API between minor versions. In most cases you should be able to use an identical configure command line, an identical configuration file, and all of your modules should continue to work. (This is only valid for versions after 2.0.41; earlier versions have incompatible changes.)

To upgrade across minor versions, start by finding the file config.nice in the build directory of your installed server or at the root of the source tree for your old install. This will contain the exact configure command line that you used to configure the source tree. Then to upgrade from one version to the next, you need only copy the config.nice file to the source tree of the new version, edit it to make any desired changes, and then run:

```
$ ./config.nice
$ make
$ make install
$ PREFIX/bin/apachectl stop
$ PREFIX/bin/apachectl start
```

Installation of TOMCAT

Installing Java Runtime Environment

To run Tomcat, you need Java Standard Edition (Java SE), also known as the JDK.

For the Tomcat installation I used SUN's latest Java SE JDK that was available at the time of this writing: Java SE Development Kit (JDK) 6 Update 10 (6u10). Regarding Java SE 6, Platform Name and Version Numbers, see http://java.sun.com/javase/6/webnotes/version-6.html. And for the whole Java version history I recommend the Wiki article http://en.wikipedia.org/wiki/Java_version_history.

You can download SUN's latest Java JDKs at: http://java.sun.com/javase/downloads/index.jsp.

For my 64-bit Debian system I selected the 64-bit JDK multiplatform binary for Linux: `jdk-6u10-linux-x64.bin`.
I downloaded the binary file to `/tmp` and installed it as follows as root:

```
# mkdir -p /usr/java
# cd /usr/java
#
# chmod 700 /tmp/jdk-6u10-linux-x64.bin
# /tmp/jdk-6u10-linux-x64.bin
...
   creating: jdk1.6.0_10/
   creating: jdk1.6.0_10/db/
   creating: jdk1.6.0_10/db/bin/
  inflating: jdk1.6.0_10/db/bin/ij
  inflating: jdk1.6.0_10/db/bin/NetworkServerControl
  inflating: jdk1.6.0_10/db/bin/setNetworkClientCP.bat
  inflating: jdk1.6.0_10/db/bin/derby_common.sh
...
Done.
# export JAVA_HOME=/usr/java/jdk1.6.0_10
# export PATH=$JAVA_HOME/bin:$PATH
#
# which java
/usr/java/jdk1.6.0_10/bin/java
# java -version
java version "1.6.0_10"
Java(TM) SE Runtime Environment (build 1.6.0_10-b33)
Java HotSpot(TM) 64-Bit Server VM (build 11.0-b15, mixed mode)
#
```

Installing Tomcat Software

Download the latest Tomcat 6.x version from http://tomcat.apache.org/download-60.cgi. For Debian I downloaded the Binary Core Distribution file `apache-tomcat-6.0.18.tar.gz` which was the latest version at the time of this writing.

Once you downloaded the tar file make sure the *MD5 checksum matches the value posted on Tomcat's web site*, see http://www.apache.org/dist/tomcat/tomcat-6/v6.0.18/bin/apache-tomcat-6.0.18.tar.gz.md5:

```
# md5sum /tmp/apache-tomcat-6.0.18.tar.gz
8354e156f097158f8d7b699078fd39c1  /tmp/apache-tomcat-6.0.18.tar.gz
```

Installing Tomcat from a binary release (tar file) requires manual creation of the Tomcat user account. This is not necessary if you install the Tomcat RPM package on a Linux system that supports RPMs.

For security reasons I created a user account with no login shell for running the Tomcat server:

```
# groupadd tomcat
# useradd -g tomcat -s /usr/sbin/nologin -m -d /home/tomcat tomcat
```
(It should be noted that other Linux systems have `nologin` under `/sbin` not `/usr/sbin`)

Next I extracted the tar file to `/var/lib` and changed the ownership of all files and directories to `tomcat`:

```
# cd /var/lib
# tar zxvf /tmp/apache-tomcat-6.0.18.tar.gz
# chown -R tomcat.tomcat /var/lib/apache-tomcat-6.0.18
```

The get the Tomcat version of the newly installed Tomcat, run:

```
# /var/lib/apache-tomcat-6.0.18/bin/version.sh
Using CATALINA_BASE:   /var/lib/apache-tomcat-6.0.18
Using CATALINA_HOME:   /var/lib/apache-tomcat-6.0.18
```

```
Using CATALINA_TMPDIR: /var/lib/apache-tomcat-6.0.18/temp
Using JRE_HOME:        /usr
Server version: Apache Tomcat/6.0.18
Server built:   Jul 22 2008 02:00:36
Server number:  6.0.18.0
OS Name:        Linux
OS Version:     2.6.18-6-amd64
Architecture:   x86_64
JVM Version:    1.4.2
JVM Vendor:     Free Software Foundation, Inc.
#
```

Starting/Stopping Tomcat

Now try to startup the Tomcat server to see whether the default Tomcat home page is being displayed.

For security reasons I don't run the Tomcat server as user `root` but as `tomcat` which was created with no login shell. Therefore, to run Tomcat use the `su` command with the `-p` option to preserves all the environment variables when switching to `tomcat` (more on the Tomcat environment variables later). And since the `tomcat` account has no login shell, it needs to be specified with the `-s` option. (You may want to use this `su` command if you plan on writing and implementing a system startup and shutdown script for system reboots.)

```
# export JAVA_HOME=/usr/java/jdk1.6.0_10
# export PATH=$JAVA_HOME/bin:$PATH
# export CATALINA_HOME=/var/lib/apache-tomcat-6.0.18
# export CATALINA_BASE=/var/lib/apache-tomcat-6.0.18
#

# su -p -s /bin/sh tomcat $CATALINA_HOME/bin/startup.sh
Using CATALINA_BASE:   /var/lib/apache-tomcat-6.0.18
Using CATALINA_HOME:   /var/lib/apache-tomcat-6.0.18
Using CATALINA_TMPDIR: /var/lib/apache-tomcat-6.0.18/temp
Using JRE_HOME:        /usr/java/jdk1.6.0_10
#
```

Now verify that Tomcat was started successfully by opening the URL `http://localhost:8080` (Port number 8080 is the default port used by Tomcat). Note that you should also be able to use the name of your server instead of `localhost`. Once you opened the URL in your browser you should see Tomcat's Congratulation page. If you don't see the page, check the log files under `$CATALINA_HOME/logs` (`/var/lib/apache-tomcat-6.0.18/logs`).

Before you continue with the next steps, make sure to shut down Tomcat since we want to run the Tomcat server out of a separate application directory which is covered in the next chapter.

```
# su -p -s /bin/sh tomcat $CATALINA_HOME/bin/shutdown.sh
Using CATALINA_BASE:   /var/lib/apache-tomcat-6.0.18
Using CATALINA_HOME:   /var/lib/apache-tomcat-6.0.18
Using CATALINA_TMPDIR: /var/lib/apache-tomcat-6.0.18/temp
Using JRE_HOME:        /usr/java/jdk1.6.0_10
#
```

Switching to Tomcat User Account

Most of the next steps in this article assume that you switched to the `tomcat` user account. If you see a '`$`' prompt, then the steps in this article are executed as the `tomcat` user. If you see a '`#`' prompt, then the steps are executed as `root`.

Since for security reasons the `tomcat` user has no login shell, it needs to be specified with the `-s` option when switching from `root` to `tomcat`:

```
# su - -s /bin/sh tomcat
```

```
$ id
uid=1001(tomcat) gid=1001(tomcat) groups=1001(tomcat)
$
```
Note that non-root users cannot switch to the `tomcat` account.

Setting Up First Tomcat JVM Instance

It is recommended not to store the web applications's files in Tomcat's distribution directory tree. For example, having a separate directory makes Tomcat upgrades easier since it won't overwrite configuration files like `server.xml`. And since this tutorial shows how to run two Tomcat instances concurrently on a single Linux server, two separate directories are needed anyway. It should be noted here that it's also possible to run multiple web applications per Tomcat JVM instance. This HOWTO shows the creation and configuration of one web application for each Tomcat instance.

Setting up Directories and Files

In the following example I setup the first Tomcat JVM instance under the base directory `/opt/tomcat-instance/sales.example.com`. It's a good practice to name the base directory after the site name, in this example `sales.example.com`.

Creating a new base directory for a new instance requires the creation and copying of various directories and configuration files. Execute the following commands as `root`:

```
# mkdir -p /opt/tomcat-instance/sales.example.com
# cd /opt/tomcat-instance/sales.example.com
#
# cp -a /var/lib/apache-tomcat-6.0.18/conf .
# mkdir common logs temp server shared webapps work
#
# chown -R tomcat.tomcat /opt/tomcat-instance
```

Most of the remaining steps are executed as the `tomcat` user. So make sure you switch from `root` to `tomcat`:

```
# su - -s /bin/sh tomcat
$ id
uid=1001(tomcat) gid=1001(tomcat) groups=1001(tomcat)
$
```

Next I created an environment file for the new Tomcat instance. This will be useful for easily setting the environment variables when starting/stopping the new Tomcat instance:

```
$ cat > /opt/tomcat-instance/sales.env << EOF
export JAVA_HOME=/usr/java/jdk1.6.0_10
export PATH=\$JAVA_HOME/bin:\$PATH
export CATALINA_HOME=/var/lib/apache-tomcat-6.0.18
export CATALINA_BASE=/opt/tomcat-instance/sales.example.com
EOF
$
$ cat /opt/tomcat-instance/sales.env
export JAVA_HOME=/usr/java/jdk1.6.0_10
export PATH=$JAVA_HOME/bin:$PATH
export CATALINA_HOME=/var/lib/apache-tomcat-6.0.18
export CATALINA_BASE=/opt/tomcat-instance/sales.example.com
$
```
`CATALINA_HOME` is the base directory of Tomcat that contains all the libraries, scripts etc. for Tomcat. This is the parent directory of the extracted Tomcat tar file.
`CATALINA_BASE` is the base directory of the new Tomcat instance, which in this example points to `/opt/tomcat-instance/sales.example.com`.

Configuring Tomcat Network Ports

Since this is the first Tomcat instance that's being created here, the default port numbers can be left unchanged in `$CATALINA_BASE/conf/server.xml` (`/opt/tomcat-instance/sales.example.com/conf/server.xml`):

```
    <Server port="8005" shutdown="SHUTDOWN">

    <Connector port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" />

    <Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

However, these port numbers will have to be changed for the second Tomcat instance though, see Steps for Second Tomcat JVM Instance and Application.

Starting First Tomcat Instance

To start the newly created Tomcat JVM instance, ensure that the environment variables are set for the new instance and execute the startup script:

```
$ source /opt/tomcat-instance/sales.env
$ $CATALINA_HOME/bin/startup.sh
Using CATALINA_BASE:   /opt/tomcat-instance/sales.example.com
Using CATALINA_HOME:   /var/lib/apache-tomcat-6.0.18
Using CATALINA_TMPDIR: /opt/tomcat-instance/sales.example.com/temp
Using JRE_HOME:       /usr/java/jdk1.6.0_10
$
```

If everything has been configured correctly, you should now see an empty white page when opening the URL `http://localhost:8080`. Note that instead of `localhost` you should also be able to use the name of your server.
If you get an error in the browser instead of an empty page, check the log files under `$CATALINA_BASE/logs` (`/opt/tomcat-instance/sales.example.com/logs`). Note that since `CATALINA_BASE` has been changed for the new Tomcat instance, the logs are no longer written to `/var/lib/apache-tomcat-6.0.18/logs`.

Relaying HTTP Port 80 Connections to Tomcat Port 8080

By default, Tomcat listens on port 8080. To have the Tomcat server itself listen on HTTP port 80, Tomcat would have to run as `root` since only `root` can listen on ports below 1024 on Linux. But for security reasons this is not recommended. The solution I prefer is to relay port 80 TCP connections to port 8080 using the Netfilter package that comes with Linux. An alternate solution would be to use a service wrapper like `jsvc` from the Jakarta Commons Daemon project. But this solution would require the installation and maintenance of another piece of software on my system that I want to avoid.

The Netfilter package that comes already with Linux is transparent to Tomcat. The following steps show how to relay port 80 TCP connections to Tomcat's port 8080 using the `iptables` command from the Netfilter package. Note that these steps must be executed as `root`:

```
# iptables -t nat -I PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 8080
# iptables -t nat -I OUTPUT -p tcp --dport 80 -j REDIRECT --to-ports 8080
```

The first rule redirects incoming requests on port 80 generated from other computer nodes, and the second rule redirects incoming requests on port 80 generated from the local node where Tomcat is running.

To see the newly configured rules, run:

```
# iptables -t nat -L
```

```
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
REDIRECT   tcp  --  anywhere             anywhere            tcp dpt:www redir ports 8080

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
REDIRECT   tcp  --  anywhere             anywhere            tcp dpt:www redir ports 8080
#
```

To remove the NAT rules we just created, you can run the `iptables -t nat -F` command which flushes and deletes the rules. Note that this will also flush any other rules that may have been configured on your system! For more information on `iptables`, see netfilter/iptables documentation.

To make the rules permanent for reboots, you can use the following option outlined here for Debian (other Linux distributions have other methods). First save the newly created rules in a file:

```
# iptables-save > /etc/iptables.conf
```

Then edit the `/etc/network/interfaces` file and add the line highlighted in blue for the public network interface. For example:

```
iface eth0 inet static
        address 192.168.1.23
        netmask 255.255.255.0
        network 192.168.1.0
        broadcast 192.168.1.255
        gateway 192.168.1.1
        pre-up iptables-restore < /etc/iptables.conf
```

The `pre-up` configuration in this example activates the `iptables` rules on my system before the public interface `eth0` comes up. So the rules can be seen with `iptables -t nat -L` after each reboot. Note that for security reasons it's important that firewall rules are established before the network interfaces come up. Even though this is not an issue for relaying Tomcat connections, as a matter of good practice, the `iptables` rules should always be established before the network comes up.

It should be noted here that there is one Tomcat configuration parameter that you may or may not want to change, the `proxyPort` parameter in the `server.xml` file. Since Tomcat still receives requests on port 8080 as they are relayed by the Linux Netfilter system from port 80, Tomcat may display port 8080 in the URL depending on the application's content. So if you want to change it to port 80, the `proxyPort` parameter would need to be added in the `$CATALINA_BASE/conf/server.xml` (`/opt/tomcat-instance/sales.example.com/conf/server.xml`). file for port 8080:

```
    <Connector port="8080" protocol="HTTP/1.1" proxyPort="80"
               connectionTimeout="20000"
               redirectPort="8443" />
```

After that you need to restart Tomcat to make this change effective.

Connecting to First Tomcat Instance Using Default HTTP Port

If `iptables` have been configured correctly, you should now be able to open the URL `http://localhost` and see an *empty white page*. You could also use the URL `http://localhost:80` (port 80 is the default port used by browsers) or the name of your server. If you get an error in the browser instead of an empty page, check the `iptables` configuration and check the log files under `$CATALINA_BASE/logs` (`/opt/tomcat-instance/sales.example.com/logs`). Note that since `CATALINA_BASE` was changed for the new Tomcat instance, the logs are no longer written to `/var/lib/apache-tomcat-6.0.18/logs`.

Experiment No. 15

Study of installation of DNS,LDAP services.

DNS Servers

DNS is the mother of the Internet (Domain Name server). The DNS server is responsible for translating IP addresses into actual names. For example when you type in your web browser: (www.domainname.com)

Before a web browser can request a web page sitting on the web server at that domain, first the browser contacts the nearest DNS server to query an IP address that matches that name.

But how does it actually work?
To answer this question, it is much better setting it up than explaining how it works.

Setting up the DNS Server

Setting up the DNS server is quite simple. Most, if not all, Linux distributions come with Bind (Berkley Internet Name Daemon) version 8 or 9.

Most likely, it was installed during your Linux installation, but if not refer to software installation in this book to install it. I really recommend it to be installed during the initial Linux installation simply because **named-bootconf.pl** generates a serial number for it (known as secret).

To configure the DNS server the **named.conf** file is used

- /etc/named.conf
- Directories: /var/named

The **named.conf** file will point to /var/named in order to query each zone. Every time you create a zone, a file will be created in the /var/named directory.

Before you make any changes make a backup copy of this file:

```
# cp named.conf named.conf.original
# vi named.conf
```

```
// generated by named-bootconf.pl
// secret must be the same as in /etc/rndc.conf
key "key" {
        algorithm       hmac-md5;
        secret
"c3Ryb25nIGVub3VnaCBmb3IgYSBtYW4gYnV0IG1hZGUgZm9yIGEgd29tYW4K";
};

controls {
    inet 127.0.0.1 allow { any; } keys { "key"; };
};


options {
        pid-file "/var/run/named/named.pid";
        directory "/var/named";
        /*
         * If there is a firewall between you and name servers you want
         * to talk to, you might need to uncomment the query-source
         * directive below.  Previous versions of BIND always asked
```

```
         * questions using port 53, but BIND 8.1 uses an unprivileged
         * port by default.
         */
        // query-source address * port 53;
};

//
// a caching only nameserver config
//
zone "." {
        type hint;
        file "named.ca";
};

zone "0.0.127.in-addr.arpa" {
        type master;
        file "named.local";
};
```

Observe that by default there are only two zones. Do not modify these two zones. By the way you should do an ls to /var/named, see that currently it holds two files **named.ca and named.local**.

When creating zones, you can name it whatever you want; but be consistent so your work will be professional (normally you will be using your domain name).

Editing the **named.conf** file can be done in several ways: manually, pre-configured bash script or using utilities

Installing the LDAP Server

Five steps are necessary to install the server: Install the pre-required packages (if not already installed), Download the server, Unpack the software, Configure the Makefiles and Build the server.

2.1 Pre-Requirements

To be fully LDAPv3 compliant, OpenLDAP clients and servers require installation of some aditional packages. In my particular case I also installed OpenLdap v2.0.11 on a out-of-box RedHat 2.2.15 distribution. My intention was to figure out if the build scripts would complain about the pre-required packages. They didn't ! Anyway, this is not the rule, you might still need to obtain and install these aditinal packages to successfully build OpenLDAP v2.xx:

OpenSSL TLS libraries

The OpenSSL TLS libraries are normally part of the base system or compose an optional software component. The official OpenSSL url is http://www.openssl.org

Kerberos Authentication Services

OpenLDAP clients and servers support Kerberos-based authentication services. In particular, OpenLDAP supports SASL/GSSAPI authentication mechanism using either Heimdal or MIT Kerberos V packages. If you desire to use Kerberos-based SASL/GSSAPI authentication, you should install either Heimdal or MIT Kerberos V. Heimdal Kerberos is available from http://www.pdc.kth.se/heimdal. MIT Kerberos is available from http://web.mit.edu/kerberos/www.

The use of strong authentication services, such as those provided by Kerberos, is highly recommended.

Cyrus's Simple Authentication and Security Layer Libraries

Cyrus's SASL libraries are normally part of the base system or compose an optional software component. Cyrus SASL is available from http://asg.web.cmu.edu/sasl/sasl-library.html. Cyrus SASL will make use of OpenSSL and Kerberos/GSSAPI libraries if preinstalled.

Database Software

OpenLDAP's slapd primary database backend, LDBM, requires a compatible database package for entry storage. LDBM is compatible with Sleepycat Software's BerkeleyDB (recommended) or with the Free Software Foundation's GNU Database Manager (GDBM). If neither of these packages are available at configure time, you will not be able build slapd with primary database backend support.

If your operating system doesn't provide one of these two packages, it's necessary to obtain one of them and install it.

BerkeleyDB is available from Sleepycat Software's download page http://www.sleepycat.com/download.html. There are several versions available. At the time of this writing, the latest release, version 3.1, is recommended.

GDBM is available from FSF's download site ftp://ftp.gnu.org/pub/gnu/gdbm. At the time of this writing, version 1.8 is the latest release.

Threads

OpenLDAP is designed to take advantage of threads. OpenLDAP supports POSIX pthreads, Mach CThreads, and a number of other varieties. configure script will complain if it cannot find a suitable thread subsystem. If this occurs, please consult the Software - Installation - Platform Hints section of the OpenLDAP FAQ http://www.openldap.org/faq.

TCP Wrappers

slapd supports TCP wrappers (IP level access control filters) if preinstalled. Use of TCP wrappers or other IP-level access filters (such as those provided by an IP-level firewall) is recommended for servers containing non-public information.

2.2 Downloading the package

There are two free distributed LDAP servers: University of Michigan LDAP server and OpenLDAP server. There's also the Netscape Directory Server, which is free only under some conditions (educational institutions get it free, for example). The OpenLDAP server is based on the latest version of the University of Michigan Server and there are mailing lists and additional documentation available for it. This document assumes that you are using the OpenLDAP server.

It's latest tar gzipped version is avaiable on the following address:

http://www.openldap.org

If you want to get the latest version of University of Michigan Server, go to this address:

ftp://terminator.rs.itd.umich.edu/ldap

To write this document, I used the 2.0.4 version of the OpenLDAP package. My operating system is a Slackware Linux with kernel 2.2.13.

On the OpenLDAP site you can always find the latest development and stable versions of the OpenLDAP server. By the time this document was updated, the latest stable version was openldap-stable-20000704.tgz. The latest development version was openldap-2.0.4.tgz.

2.3 Unpacking the server

Now that you have the tar gzipped package on your local machine, you can unpack it.

First copy the package to a desirable directory, for example /usr/local.

Then use the following command:

tar xvzf openldap-stable.tgz

You can use this command too, as well:

gunzip openldap-stable.tgz | tar xvf -

2.4 Configuring the software

There are several options that you would like to customize so you can build the best software for your site.

To configure the software you just need 2 steps:

- Edit the file ldapconfig.h.edit, located on the subdirectory include beneath the directory where you unpacked the software.
- Run the configure script (if you are a tough guy, you can also edit the Make-common file instead of running the configure script :^)

In the file include/ldapconfig.h.edit you can set options like the location of the slapd and slurpd daemons. The file itself is well commented and it's default settings also reflect the most common administrator choices so, if you are in a hurry you can skip this step:

vi include/ldapconfig.h.edit

The OpenLDAP server sources are distributed with a configuration script for setting options like instalation directories, compiler and linker flags. Type the following command on the directory where you unpacked the software:

./configure --help
This will print all options that you can customize with the configure script before you build the software. Some usefull options are --prefix=pref , --exec-prefix=eprefix and --bindir=dir, for setting instalation directories. Normally if you run configure without options, it will auto-detect the appropriate settings and prepar to build things on the default common location. So just type:

./configure
And watch the output to see if all went well

2.5 Building the server

After configuring the software you can start building it. First build the dependencies, using the command:

make depend
After build the server, using the command:

make
If all goes well, the server will build as configured. If not, return to the previous step to review the configuration settings. You should check the platform specific hints, they are located in the path doc/install/hints under the directory you unpacked the software.

Now install the binaries and man pages. You may need to be superuser to do this (depending on where you are installing things):

su
make install
That's all, now you have the binary of the server and the binaries of several other utilities. Go to the next section to see how to configure the operation of your LDAP server.

The binary of the OpenLdap 2.0 server is called slapd. OpenLdap 2.0 was officially released on August, 30th and it comprises Ldap protocol v3, as defined on the RFC 2251.

The main features of OpenLDAP 2.0 are:

- LDAPv2 and LDAPv3 Support (RFC2251-2256,2829-2831)
- Maintenance of interoperability with existing clients
- IPv4 and IPv6 support
- Strong Authentication (SASL) (RFC2829)
- Start TLS (RFC2830)
- Language Tags (RFC2596)
- DNS-based service location (RFC2247+"locate" I-D)
- Enhanced Standalone Server
- Named References/ManageDsaIT ("nameref" I-D)
- Enhanced Access Control subsystem
- Thread pooling
- Preemptive threading support
- Multiple listener support
- LDIFv1 (RFC2849)
- Improved platform/subsystem detection

Note: There will be a document on the Linux Documentation Project (LDP) called LDAP Implementation HOWTO. This document will be a great resource for those who want to explore the new features of OpenLDAP 2.0. The date for it's release is around December 2000.

On the latest versions of the OpenLDAP package, it's also possible to test the recently built binaries. The package comes with a test script, which you can run using the command:

make test

If anything goes wrong with the script you can just abort it hitting Ctrl-C.

Experiment No. 16

Study of installation of Firawall & Proxy Server

Understanding Firewalls

A firewall is a structure intended to keep a fire from spreading. Building have firewalls made of brick walls completely dividing sections of the building. In a car a firewall is the metal wall separating the engine and passenger compartments.

Internet firewalls are intended to keep the flames of Internet hell out of your private LAN. Or, to keep the members of your LAN pure and chaste by denying them access the all the evil Internet temptations. ;-)

The first computer firewall was a non-routing Unix host with connections to two different networks. One network card connected to the Internet and the other to the private LAN. To reach the Internet from the private network, you had to logon to the firewall (Unix) server. You then used the resources of the system to access the Internet. For example, you could use X-windows to run Netscape's browser on the firewall system and have the display on your work station. With the browser running on the firewall it has access to both networks.

This sort of dual homed system (a system with two network connections) is great if you can TRUST ALL of your users. You can simple setup a Linux system and give an account accounts on it to everyone needing Internet access. With this setup, the only computer on your private network that knows anything about the outside world is the firewall. No one can download to their personal workstations. They must first download a file to the firewall and then download the file from the firewall to their workstation.

BIG NOTE: 99% of all break-ins start with gaining account level access on the system being attacked. Because of this I don't recommend this type of firewall. It is also very limiting.

2.1 Firewall Politics

You shouldn't believe a firewall machine is all you need. *Set policies first*.

Firewalls are used for two purposes.

1. to keep people (worms / crackers) out.
2. to keep people (employees / children) in.

When I started working on firewalls I was surprised to learn the company I worked for were more interested in "spying" on their employees then keeping crackers out of their networks.

At least in my state (Oklahoma) employers have the right to monitor phone calls and Internet activity as long as they inform the employees they are doing it.

Big Brother is not government. Big Brother = Big Business.

Don't get me wrong. People should work, not play at work. And I feel the work ethic has been eroding. However, I have also observed that management types are the biggest abusers of the rules they set. I have seen hourly workers reprimanded for using the Internet to looking for bus routesto get to work while the same manager used hours of work time looking for fine restaurants and nightclubs to take prospective customers.

My fix for this type of abuse is to publish the firewall logs on a Web page for everyone to see.

The security business can be scary. If you are the firewall manager, watch your back.

How it create a security policy

I have seen some realy high folutin documentation on how to create a security policy. After many years of experence I know now say, don't believe a word of them. Create a security policy is simple.

1. describe what you need to service
2. describe the group of people you need to service
3. describe which service each group needs access to
4. for each service group describe how the service should be keep secure
5. write a statment making all other forms of access a vialation

Your policy will become more complicated with time but don't try to cover to much ground now. Make it simple and clear.

2.2 Types of Firewalls

There are two types of firewalls.

1. Filtering Firewalls - that block selected network packets.
2. Proxy Servers (sometimes called firewalls) - that make network connections for you.

Packet Filtering Firewalls

Packet Filtering is the type of firewall built into the Linux kernel.

A filtering firewall works at the network level. Data is only allowed to leave the system if the firewall rules allow it. As packets arrive they are filtered by their type, source address, destination address, and port information contained in each packet.

Many network routers have the ability to perform some firewall services. Filtering firewalls can be thought of as a type of router. Because of this you need a deep understanding of IP packet structure to work with one.

Because very little data is analyzed and logged, filtering firewalls take less CPU and create less latency in your network.

Filtering firewalls do not provide for password controls. User can not identify themselves. The only identity a user has is the IP number assigned to their workstation. This can be a problem if you are going to use DHCP (Dynamic IP assignments). This is because rules are based on IP numbers you will have to adjust the rules as new IP numbers are assigned. I don't know how to automate this process.

Filtering firewalls are more transparent to the user. The user does not have to setup rules in their applications to use the Internet. With most proxy servers this is not true.

Proxy Servers

Proxies are mostly used to control, or monitor, outbound traffic. Some application proxies cache the requested data. This lowers bandwidth requirements and decreases the access the same data for the next user. It also gives unquestionable evidence of what was transferred.

There are two types of proxy servers.

1. Application Proxies - that do the work for you.
2. SOCKS Proxies - that cross wire ports.

Application Proxy

The best example is a person telneting to another computer and then telneting from there to the outside world. With a application proxy server the process is automated. As you telnet to the outside world the client send you to the proxy first. The proxy then connects to the server you requested (the outside world) and returns the data to you.

Because proxy servers are handling all the communications, they can log everything they (you) do. For HTTP (web) proxies this includes very URL they you see. For FTP proxies this includes every file you download. They can even filter out "inappropriate" words from the sites you visit or scan for viruses.

Application proxy servers can authenticate users. Before a connection to the outside is made, the server can ask the user to login first. To a web user this would make every site look like it required a login.

SOCKS Proxy

A SOCKS server is a lot like an old switch board. It simply cross wires your connection through the system to another outside connection.

Most SOCKS server only work with TCP type connections. And like filtering firewalls they don't provide for user authentication. They can however record where each user connected to.

The SOCKS Proxy Server

Setting up the Proxy Server

The SOCKS proxy server available from **http://www.socks.nec.com/**.

Uncompressed and untar the files into a directory on your system, and follow the instructions on how to make it. I had a couple problems when I made it. Make sure that your Makefiles are correct.

One important thing to note is that the proxy server needs to be added to /etc/inetd.conf. You must add a line:

```
socks  stream  tcp  nowait  nobody  /usr/local/etc/sockd  sockd
```

to tell the server to run when requested.

Configuring the Proxy Server

The SOCKS program needs two separate configuration files. One to tell the access allowed, and one to route the requests to the appropriate proxy server. The access file should be housed on the server. The routing file should be housed on every UNIX machine. The DOS and, presumably, Macintosh computers will do their own routing.

The Access File

With socks4.2 Beta, the access file is called "sockd.conf".It should contain 2 lines, a permit and a deny line. Each line will have three entries:

- The Identifier (permit/deny)
- The IP address
- The address modifier

The identifier is either permit or deny. You should have both a permit and a deny line.

The IP address holds a four byte address in typical IP dot notation. I.E. 192.168.1.0.

The address modifier is also a typical IP address four byte number. It works like a netmask. Envision this number to be 32 bits (1s or 0s). If the bit is a 1, the corresponding bit of the address that it is checking must match the corresponding bit in the IP address field. For instance, if the line is:

```
permit 192.168.1.23 255.255.255.255
```

it will permit only the IP address that matches every bit in 192.168.1.23, eg, only 192.168.1.3. The line:

```
permit 192.168.1.0 255.255.255.0
```

will permit every number within group 192.168.1.0 through 192.168.1.255, the whole C Class domain. One should not have the line:

```
permit 192.168.1.0 0.0.0.0
```

as this will permit every address, regardless.

So, first permit every address you want to permit, and then deny the rest. To allow everyone in the domain 192.168.1.xxx, the lines:

```
permit 192.168.1.0 255.255.255.0
deny 0.0.0.0 0.0.0.0
```

will work nicely. Notice the first "0.0.0.0" in the deny line. With a modifier of 0.0.0.0, the IP address field does not matter. All 0's is the norm because it is easy to type.

More than one entry of each is allowed.

Specific users can also be granted or denied access. This is done via ident authentication. Not all systems support ident, including Trumpet Winsock, so I will not go into it here. The documentation with socks is quite adequate on this subject.

The Routing File

The routing file in SOCKS is poorly named "socks.conf". I say "poorly named" because it is so close to the name of the access file that it is easy to get the two confused.

The routing file is there to tell the SOCKS clients when to use socks and when not to. For instance, in our network, 192.168.1.3 will not need to use socks to talk with 192.168.1.1, firewall. It has a direct connection in via Ethernet. It defines 127.0.0.1, the loopback, automatically. Of course you do not need SOCKS to talk to yourself. There are three entries:

- deny
- direct
- sockd

Deny tells SOCKS when to reject a request. This entry has the same three fields as in sockd.conf, identifier, address and modifier. Generally, since this is also handled by sockd.conf, the access file, the modifier field is set to 0.0.0.0. If you want to preclude yourself from calling any place, you can do it here.

The direct entry tells which addresses to not use socks for. These are all the addresses that can be reached without the proxy server. Again we have the three fields, identifier, address and modifier. Our example would have

```
direct 192.168.1.0 255.255.255.0
```

Thus going direct for any on our protected network.

The sockd entry tells the computer which host has the socks server daemon on it. The syntax is:

```
sockd @=<serverlist> <IP address> <modifier>
```

Notice the @= entry. This allows you to set the IP addresses of a list of proxy servers. In our example, we only use one proxy server. But, you can have many to allow a greater load and for redundancy in case of failure.

The IP address and modifier fields work just like in the other examples. You specify which addresses go where through these. 6.2.3. DNS from behind a Firewall

Setting up Domain Name service from behind a firewall is a relatively simple task. You need merely to set up the DNS on the firewalling machine. Then, set each machine behind the firewall to use this DNS.

Working With a Proxy Server

Unix

To have your applications work with the proxy server, they need to be "sockified". You will need two different telnets, one for direct communication, one for communication via the proxy server. SOCKS comes with instructions on how to SOCKify a program, as well as a couple pre-SOCKified programs. If you use the SOCKified version to go somewhere direct, SOCKS will automatically switch over to the direct version for you. Because of this, we want to rename all the programs on our protected network and replace them with the SOCKified programs. "Finger" becomes "finger.orig", "telnet" becomes "telnet.orig", etc. You must tell SOCKS about each of these via the include/socks.h file.

Certain programs will handle routing and sockifying itself. Netscape is one of these. You can use a proxy server under Netscape by entering the server's address (192.168.1.1 in our case) in the SOCKs field under Proxies. Each application will need at least a little messing with, regardless of how it handles a proxy server.

MS Windows with Trumpet Winsock

Trumpet Winsock comes with built in proxy server capabilities. In the "setup" menu, enter the IP address of the server, and the addresses of all the computers reachable directly. Trumpet will then handle all outgoing packets.

Getting the Proxy Server to work with UDP Packets

The SOCKS package works only with TCP packets, not UDP. This makes it quite a bit less useful. Many useful programs, such as talk and Archie, use UDP. There is a package designed to be used as a proxy server for UDP packets called UDPrelay, by Tom Fitzgerald <fitz@wang.com>. Unfortunately, at the time of this writing, it is not compatible with Linux.

Drawbacks with Proxy Servers

The proxy server is, above all, a `security device`. Using it to increase internet access with limited IP addresses will have many drawbacks. A proxy server will allow greater access from inside the protected network to the outside, but will keep the inside completely inaccessible from the outside. This means no servers, talk or archive connections, or direct mailing to the inside computers. These drawbacks might seem slight, but think of it this way:

- You have left a report you are doing on your computer inside a firewall protected network. You are at home, and decide that you would like to go over it. You can not. You can not reach your computer because it is behind the firewall. You try to log into `firewall` first, but since everyone has proxy server access, no one has set up an account for you on it.
- Your daughter goes to college. You want to email her. You have some private things to talk about, and would rather have your mail sent directly to your machine. You trust your systems administrator completely, but still, this is private mail.
- The inability to use UDP packets represents a big drawback with the proxy servers. I imagine UDP capabilities will be coming shortly.

FTP causes another problem with a proxy server. When getting or doing an `ls`, the FTP server opens a socket on the client machine and sends the information through it. A proxy server will not allow this, so FTP doesn't particularly work.

And, proxy servers run slow. Because of the greater overhead, almost any other means of getting this access will be faster.

Basically, if you have the IP addresses, and you are not worried about security, do not use a firewall and/or proxy servers. If you do not have the IP addresses, but you are also not worried about security, you might also want to look into using an IP emulator, like Term, Slirp or TIA. Term is available from **`ftp://sunsite.unc.edu`**, Slirp is available from **`ftp://blitzen.canberra.edu.au/pub/slirp`**, and TIA is available from marketplace.com. These packages will run faster, allow better connections, and provide a greater level of access to the inside network from the internet. Proxy servers are good for those networks which have a lot of hosts that will want to connect to the internet on the fly, with one setup and little work after that.

Experiment.17

Study and use of commands for Process Management

Starting a Process:

When you start a process (run a command), there are two ways you can run it:

- Foreground Processes
- Background Processes

Foreground Processes:

By default, every process that you start runs in the foreground. It gets its input from the keyboard and sends its output to the screen.
You can see this happen with the ls command. If I want to list all the files in my current directory, I can use the following command:
$ls ch*.doc
This would display all the files whose name start with ch and ends with .doc:
ch01-1.doc ch010.doc ch02.doc ch03-2.doc
ch04-1.doc ch040.doc ch05.doc ch06-2.doc
ch01-2.doc ch02-1.doc

The process runs in the foreground, the output is directed to my screen, and if the ls command wants any input (which it does not), it waits for it from the keyboard.
While a program is running in foreground and taking much time, we cannot run any other commands (start any other processes) because prompt would not be available until program finishes its processing and comes out.

Background Processes:

A background process runs without being connected to your keyboard. If the background process requires any keyboard input, it waits.
The advantage of running a process in the background is that you can run other commands; you do not have to wait until it completes to start another!
The simplest way to start a background process is to add an ampersand (&) at the end of the command.

$ls ch*.doc &

This would also display all the files whose name start with ch and ends with .doc:
ch01-1.doc ch010.doc ch02.doc ch03-2.doc
ch04-1.doc ch040.doc ch05.doc ch06-2.doc
ch01-2.doc ch02-1.doc
Here if the **ls** command wants any input (which it does not), it goes into a stop state until I move it into the foreground and give it the data from the keyboard.
That first line contains information about the background process - the job number and process ID. You need to know the job number to manipulate it between background and foreground.
If you press the Enter key now, you see the following:

[1] + Done ls ch*.doc & $

The first line tells you that the **ls** command background process finishes successfully. The second is a prompt for another command.

Listing Running Processes:

It is easy to see your own processes by running the **ps** (process status) command as follows:

$ps

PID TTY TIME CMD
18358 ttyp3 00:00:00 sh
18361 ttyp3 00:01:31 abiword
18789 ttyp3 00:00:00 ps
One of the most commonly used flags for ps is the **-f** (f for full) option, which provides more information as shown in the following example:

$ps -f

UID PID PPID C STIME TTY TIME CMD
amrood 6738 3662 0 10:23:03 pts/6 0:00 first_one
amrood 6739 3662 0 10:22:54 pts/6 0:00 second_one
amrood 3662 3657 0 08:10:53 pts/6 0:00 -ksh

UID	User ID that this process belongs to (the person running it).
PID	Process ID.
PPID	Parent process ID (the ID of the process that started it).
C	CPU utilization of process.
STIME	Process start time.
TTY	Terminal type associated with the process
TIME	CPU time taken by the process.
CMD	The command that started this process.

Experiment.18

Configure, customize and deploy DHCP in Linux.

The DHCP clients request an IP address and other network settings from the **DHCP server** on the network. The **DHCP server** in turn leases the client an IP address within a given range or leases the client an IP address based on the MAC address of the client's network interface card (NIC). The information includes its IP address, along with the network's name server, gateway, and proxy addresses,including the netmask.

Nothing has to be configured manually on the local system, except to specify the **DHCP server** it should get its network configuration from. If an IP address is assigned according to the MAC address of the client's NIC, the same IP address can be leased to the client every time the client requests one. DHCP makes network administration easier and less prone to error.

Exam Question Configure the **DHCP server** by matching the following conditions:

- Subnet and netmask should be 192.168.0.0 255.255.255.0
- Gateway Should be 192.168.0.254
- DNS Sever Should be 192.168.0.254
- Domain Name should be example.com
- Range from 192.168.0.10-50

Exam Question You have **DHCP server**, which assigns the IP, gateway and DNS server ip to Clients. There is one DNS servers having MAC address **(00:50:FC:98:8D:00** in your LAN, But it always required fixed IP address **(192.168.0.10)**. Configure the **DHCP server** to assign the fixed IP address to DNS server.

Configure dhcp server

In this example we will configure a **dhcp server** and will lease ip address to clients.

For this example we are using three systems one linux server one linux clients and one window clients.

dhcp rpm is required to configure dhcp server. check it if not found then install

```
[root@Server ~]# rpm -qa dhcp
dhcp-3.0.5-7.el5
[root@Server ~]# _
```

Now check **dhcpd** service in system service it should be on

```
#setup
Select  System service
from list [*]dhcpd
```

To assign IP to dhcp server

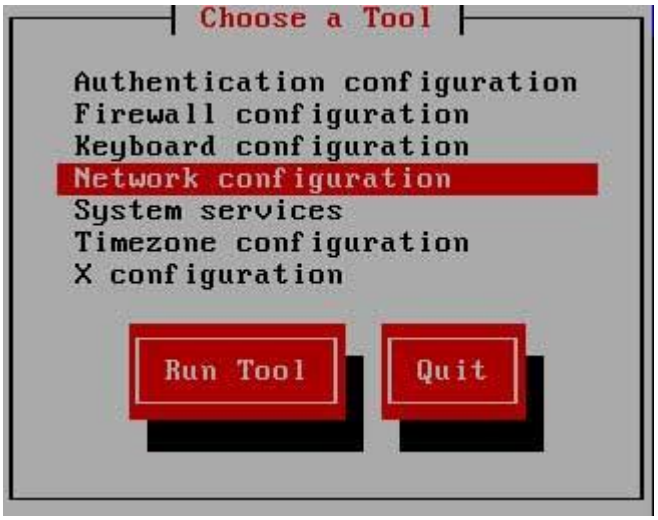
DHCP server have a static a ip address. First configure the ip address **192.168.0.254** with netmask of **255.255.255.0** on server.

Run **setup** command form root user

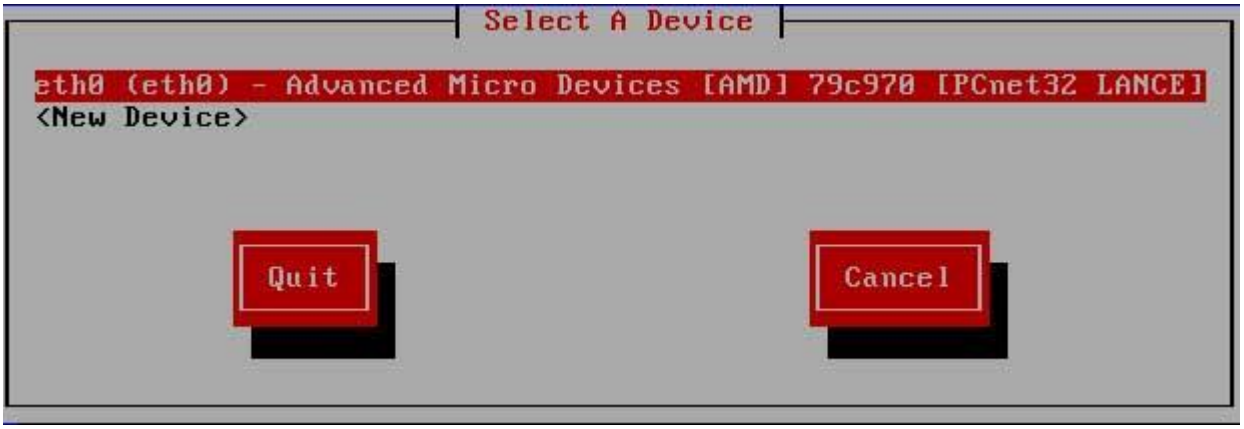
```
#setup
```

```
[root@localhost Server]# setup_
```

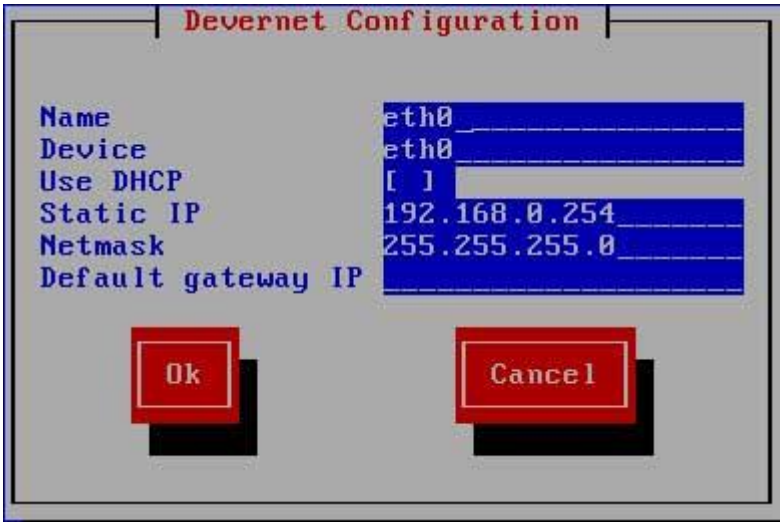
this will launch a new window select **network configuration**



now a new window will show you all available LAN card select your LAN card (if you don't see any LAN card here mean you don't have install driver)



assign IP in this box and click ok



click on ok, quit and again quit to come back on root prompt.

restart the **network service** so new ip address can take place on LAN card

```
#service network restart
```

main configuration file of dhcp server is **dhcpd.conf.** This file located on **/etc** directory. If this file is not present there or you have corrupted this file, then copy new file first, if ask for overwrite press **y**

```
[root@Server ~]# cp /usr/share/doc/dhcp-3.0.5/dhcpd.conf.sample /etc/dhcpd.conf
cp: overwrite `/etc/dhcpd.conf'? y
[root@Server ~]# _
```

now open **/etc/dhcpd.conf**

```
[root@Server ~]# vi /etc/dhcpd.conf _
```

default entry in this file look like this

```
ddns-update-style interim;
ignore client-updates;

subnet 192.168.0.0 netmask 255.255.255.0 {

# --- default gateway
        option routers                  192.168.0.1;
        option subnet-mask              255.255.255.0;

        option nis-domain               "domain.org";
        option domain-name              "domain.org";
        option domain-name-servers      192.168.1.1;

        option time-offset              -18000; # Eastern
#       option ntp-servers              192.168.1.1;
#       option netbios-name-servers     192.168.1.1;
# --- Selects point-to-point node (default is hybrid). Do
# -- you understand Netbios very well
#       option netbios-node-type 2;

        range dynamic-bootp 192.168.0.128 192.168.0.254;
        default-lease-time 21600;
        max-lease-time 43200;
```

make **these change** in this file to configure **dhcp server**

```
remove this line # - - - default gateway
set option routers to 192.168.0.254
set option subnet-mask to 255.255.255.0
option nis domain to example.com
option domain-name to example.com
option domain-name-servers to 192.168.0.254
range dynamic-bootp to 192.168.0.10 192.168.0.50;
```

```
ddns-update-style interim;
ignore client-updates;

subnet 192.168.0.0 netmask 255.255.255.0 {

        option routers                  192.168.0.254;
        option subnet-mask              255.255.255.0;

        option nis-domain               "example.com";
        option domain-name              "example.com";
        option domain-name-servers      192.168.0.254;

        option time-offset              -18000; # Eastern
#       option ntp-servers              192.168.1.1;
#       option netbios-name-servers     192.168.1.1;
# --- Selects point-to-point node (default is hybrid). Do
# -- you understand Netbios very well
#       option netbios-node-type 2;

        range dynamic-bootp 192.168.0.10 192.168.0.50;
        default-lease-time 21600;
        max-lease-time 43200;
```

how to assign fix ip address to any host

locate this paragraph and change **hardware Ethernet** to client's **mac address** and **fixed -address** to **ip address** which you want to provide that host

```
# we want the nameserver to appear at a fixed address
host ns {
        next-server marvin.redhat.com;
        hardware ethernet 12:34:56:78:AB:CD;
        fixed-address 207.175.42.254;
}
```

after making necessary change **save** file and exit

now create a **blank file** use to store the allocated **ip address information**

```
[root@Server ~]# touch /var/lib/dhcpd/dhcpd.leases
[root@Server ~]# _
```

Now **restart dhcpd service** and on it with **chkconfig** commands

```
[root@Server ~]# service dhcpd restart
Shutting down dhcpd:                                       [FAILED]
Starting dhcpd:                                            [  OK  ]
[root@Server ~]# chkconfig dhcpd on
[root@Server ~]# _
```

Linux Client configuration

Client configuration is very easy and straightforward. All you need to do is set ip address to dynamic in the properties of lan card. In linux

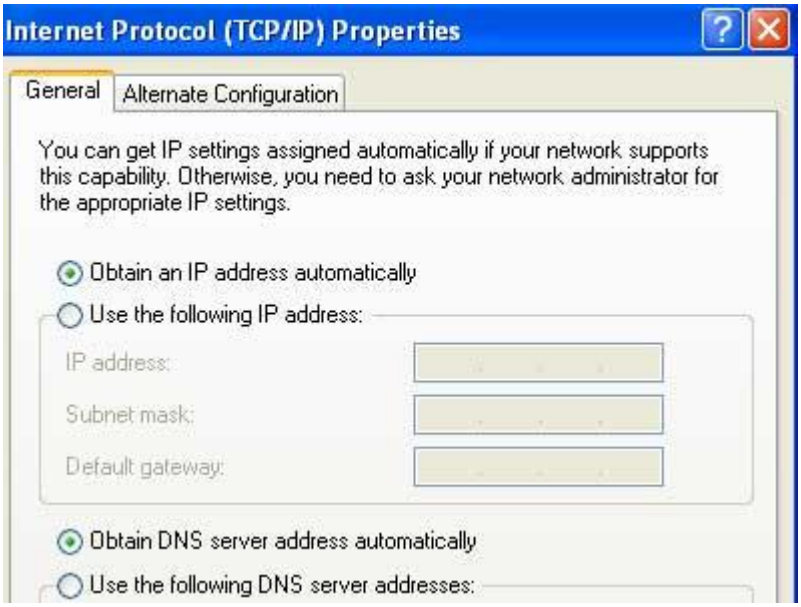
#setup
select **network configuration** from menu list
Select **lan card** and enter on ok
Select **USE DHCP** and enter on ok
Now click on **quit**
and **quit** to come back on root prompt

Now restart the **network service** to obtain ip from **dhcp server**

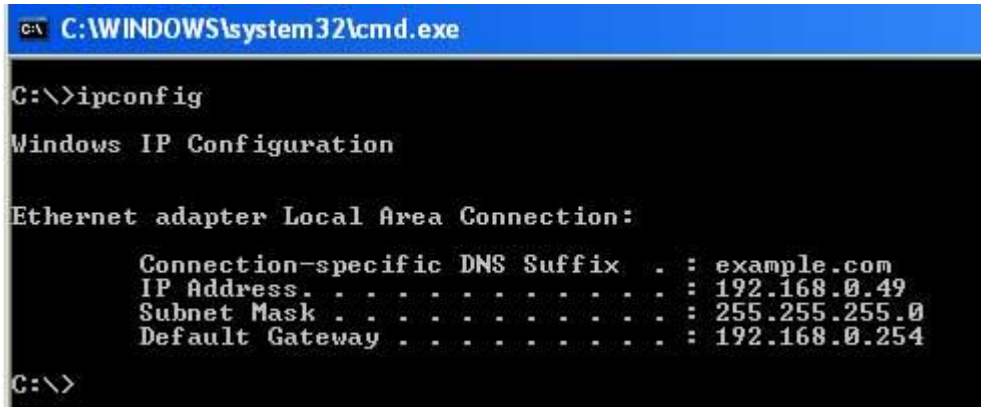
```
[root@Client1 temp]# service network restart
Shutting down interface eth0:                              [  OK  ]
Shutting down loopback interface:                          [  OK  ]
Bringing up loopback interface:                            [  OK  ]
Bringing up interface eth0:
Determining IP information for eth0... done.
                                                           [  OK  ]
[root@Client1 temp]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0C:29:62:28:1A
          inet addr:192.168.0.50  Bcast:192.168.0.255  Mask:255.255.
          inet6 addr: fe80::20c:29ff:fe62:281a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:42 errors:0 dropped:0 overruns:0 frame:0
          TX packets:134 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5962 (5.8 KiB)  TX bytes:23484 (22.9 KiB)
          Interrupt:67 Base address:0x2000

[root@Client1 temp]# _
```

Window Client configuration

To configure windows system as **dhcp clients** open lan card **properties** and select **tcp/ip** and click on properties and set **obtain ip address automatically**



Go on **command prompt** and check new **ip address**



Check lease on DHCP server

you can check **allocated address** on server.

```
[root@Server ~]# cat /var/lib/dhcpd/dhcpd.leases_
```

```
lease 192.168.0.50 {
  starts 3 2010/02/17 12:13:27;
  ends 3 2010/02/17 18:13:27;
  binding state active;
  next binding state free;
  hardware ethernet 00:0c:29:62:28:1a;
}
lease 192.168.0.49 {
  starts 3 2010/02/17 12:14:38;
  ends 3 2010/02/17 18:14:38;
  binding state active;
  next binding state free;
  hardware ethernet 00:0c:29:69:d8:2f;
  uid "\001\000\014)i\330/";
  client-hostname "nikki-82617912b";
}
[root@Server ~]# _
```