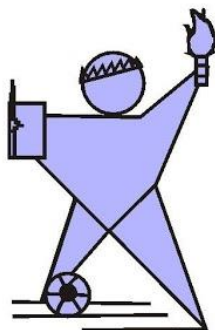


IPS ACADEMY INDORE

INSTITUTE OF ENGINEERING & SCIENCE
COMPUTER SCIENCE & ENGINEERING
DEPARTMENT



LAB MANUAL

(2020-21)

Computer Programming V Unix

Name:- PRANAV SONARE

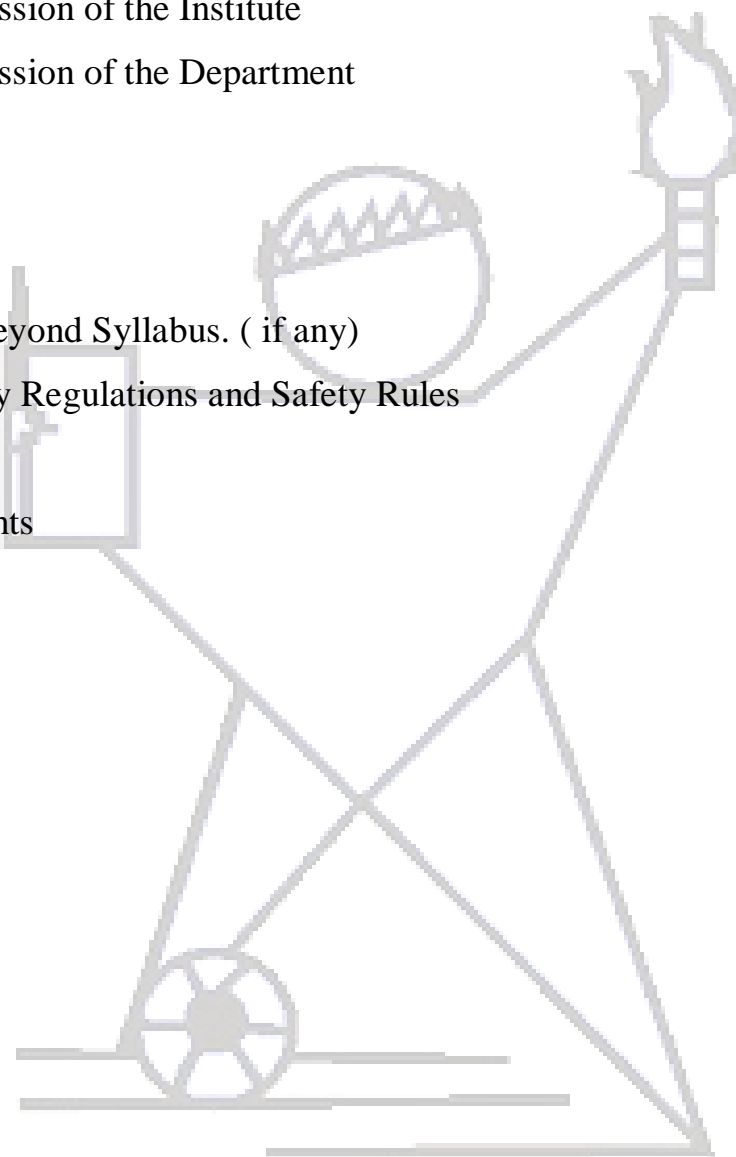
Year:- 3rd Semester:- 5th

Class Roll No.:-05

Enrollment No.:- 0808CS181127

CONTENTS

1. Vision Mission of the Institute
2. Vision Mission of the Department
3. PEOs
4. POs
5. COs
6. Content beyond Syllabus. (if any)
7. Laboratory Regulations and Safety Rules
8. Index
9. Experiments

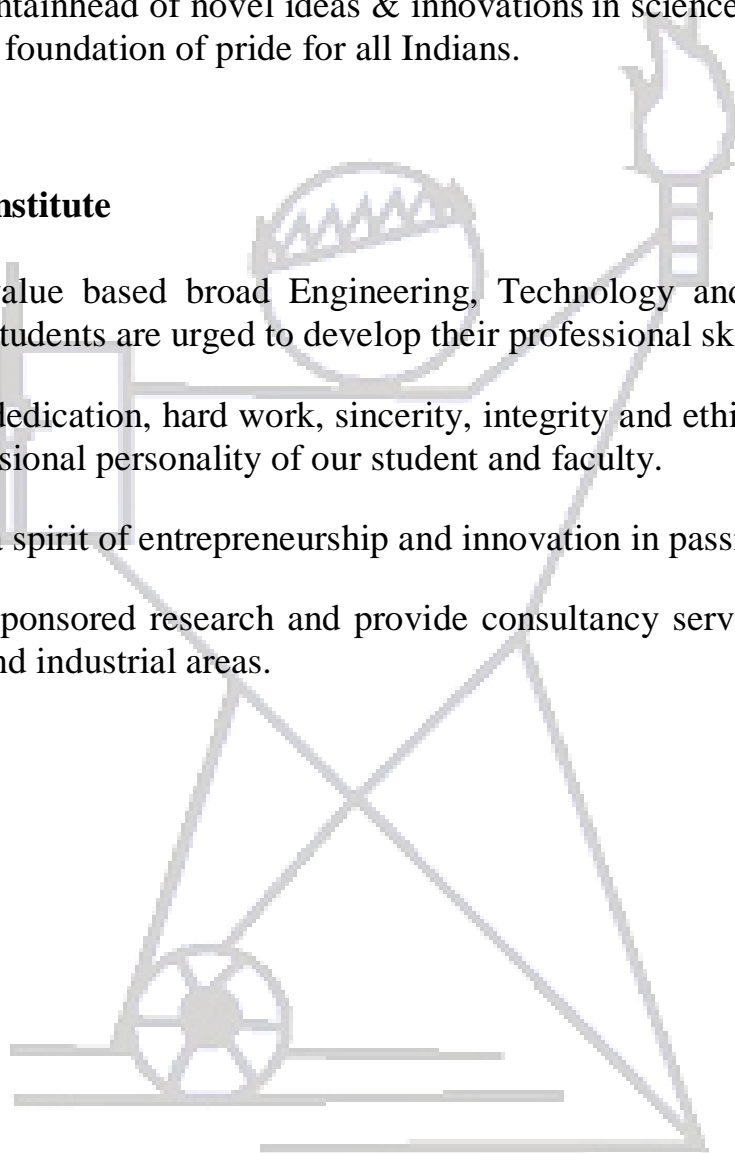


Vision of the Institute

To be the fountainhead of novel ideas & innovations in science & technology & persist to be a foundation of pride for all Indians.

Mission of the Institute

- To provide value based broad Engineering, Technology and Science where education in students are urged to develop their professional skills.
- To inculcate dedication, hard work, sincerity, integrity and ethics in building up overall professional personality of our student and faculty.
- To inculcate a spirit of entrepreneurship and innovation in passing out students.
- To instigate sponsored research and provide consultancy services in technical, educational and industrial areas.

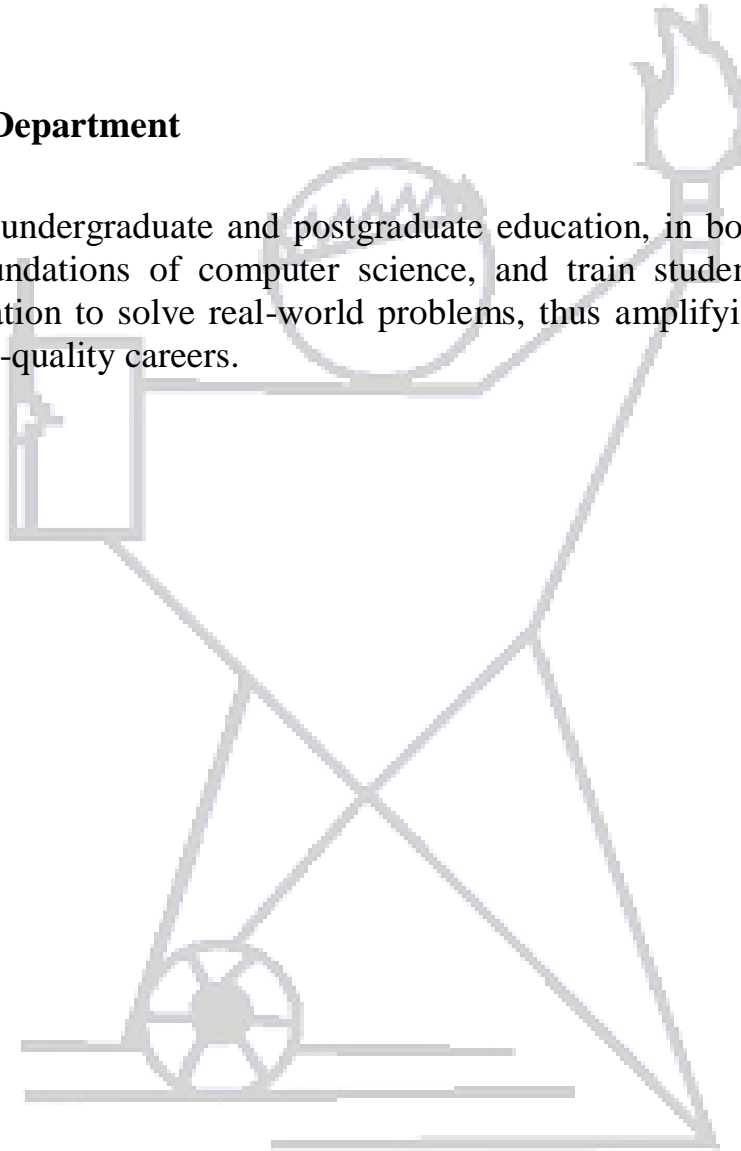


Vision of the Department

Attaining global recognition in computer science and engineering education, research and training to meet the growing needs of the industry and society.

Mission of the Department

Provide quality undergraduate and postgraduate education, in both the theoretical and applied foundations of computer science, and train students to effectively apply this education to solve real-world problems, thus amplifying their potential for lifelong high-quality careers.



Program Education Objectives (PEOs)

1. To prepare students for successful careers in software industry that meet the needs of Indian and multinational companies.
2. To develop the skills among students to analyze real world problem & implement with computer engineering solution and in multidisciplinary projects.
3. To provide students with solid foundation in mathematical, scientific and engineering fundamentals to solve engineering problems and required to pursue higher studies.
4. To develop the ability to work with the core competence of computer science & engineering i.e. software engineering, hardware structure & networking concepts so that one can find feasible solution to real world problems.
5. To inculcate in students professional and ethical attitude, effective communication skills, team work skills, multidisciplinary approach, and an ability to relate engineering issues to broader social context.
6. To motivate students perseverance for lifelong learning and to introduce them to professional ethics and codes of professional practice.

Program Outcomes (Pos)

- a. Graduates will demonstrate knowledge of mathematics, science and allied engineering in computer science & engineering.
- b. Graduates will demonstrate ability to analysis, design and implement the problems as per user requirements and specification.
- c. Graduates will possess strong fundamental concepts on database technologies, operating system, compiler design, networking, data structure, software engineering.
- d. Graduates will be able to demonstrate with excellent programming, analytical, logical and problem solving skills.
- e. Graduates will demonstrate and ability to visualize and work on laboratory and multidisciplinary tasks.
- f. Graduates will demonstrate skills to use modern engineering tools, softwares and equipment to analyze problems.
- g. Graduates will posses leadership and management skills with best professional ethical practices.
- h. Graduates will be able to communicate effectively in both verbal and written form.
- i. Graduates will have the confidence to apply computer science and engineering solution for the welfare of human being.
- j. Graduates will enhance their self confidence and capable of being a kaizen.
- k. Graduates will have ability for life long learning and acquire capability to succeed in examination like GATE, GRE, IES etc.

Course Outcomes (Cos)

- CO1 To understand Basics Unix/Linux Operating systems, how to install Unix/Linux, Run basic shell commands and explain features of Linux/Unix Operating System.
- CO2 To understand the architecture of Unix/Linux and Virtual File System..
- CO3 To understand Scheduling Priorities and Change the Priority of a time-sharing process.
- CO4 To understand File Access Commands or Students are able to know about, Access Control List (ACLs), Setting ACL Entries, Modifying ACL entries on a file, Deleting ACL entries on a file.
- CO5 To Install SAMBA, APACHE TOMCAT servers, understand how to implement DNS and implement Firewall and Proxy server

S. No.	Experiment Name	Date	Grade	Signature
1.	To Study basic & User status Unix/Linux Commands.			
2.	Study & use of commands for performing arithmetic operations with Unix/Linux.			
3.	Create a file called wlcc.txt with some lines and display how many lines, words and characters are present in that file.			
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?			
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			
6.	Create a program to find out the inode number of any desired file.			
7.	Study & use of the Command for changing file permissions			
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 users.ux			
9.	Execute shell commands through vi editor.			
10.	Installation, Configuration & Customizations of Unix/Linux.			
11.	Write a shell script that accepts any number of arguments and prints them in the reverse order.			
12	Write a shell script to find the smallest of three numbers that are read from the keyboard.			
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.			
14	Installation of SAMBA, APACHE, TOMCAT			
15	Implementation of DNS, LDAP services			
16	Study & installation of Firewall & Proxy serverhttp			
17	To Demonstrate a simple Unix shell program			

18	Shell Script Program to Print Student Mark Sheet			
19	Shell Script Program to perform all Arithmetic operations on integers			
20	Shell Script Program to perform all arithmetic operations on floating point			

Content beyond syllabus

Experiment No. 1

Aim : To Study basic & User status Unix/Linux Commands.

Here are some basic and user status unix/linux commands alphabetically.

A

at : execute commands at a specified time/date.

awk : a scripting language, especially useful for manipulating text and automation.

B

bash : invokes the Bourne Again Shell (standard on most boxes).

batch : execute commands when load permits.

bc : interactive C-like calculator (integers only).

C

cal : displays a calendar, also lets you choose month/year using parameters.

calendar : invoke a reminder service.

cancel : cancel request to calendar.

cat : concatenate files (displays a file without scrolling ability. Simply dumps it to the standard output. Can be useful when chaining multiple applications to do complicated jobs, so one application can use another's output as input).

cd : change the current working directory.

chgrp : change group ownership of a file.

chmod : change access patterns (permissions) to files.

chown : change user ownership of files.

clear : clear the screen.

cmp : compare two files.

cp : copy files.

cpio : archive and extract files.

cron : clock daemon (executes "batch" and "at" commands).

crontab : schedules commands at regular intervals.

crypt : encrypt , decrypt files using altered DES, standard to Unix passwords (restricted distribution).

csh : invoke the C shell.

csplit : split file into several other files.
cu : call up another unix terminal.
cut : cut selected fields from each line of file.

D

date : displays the time and date (can also change it if you're root).
dd : convert and copy a file.
df : reports space (free, total etc') on all mounted file systems.
diff : compare two files.
diff3 : compare 3 or more files.
dircmp : compare two directories.
du : report disk usage.

E

echo : echo argument to standard output.
ed : line oriented editor.
egrep : extended version of grep (searches for extended regular expressions).
expr : evaluate boolean and arithmetic expression.

F

fgrep : same as grep, only it interprets patterns as a list of fixed strings.
false : return nonzero (false) exit status.
file : report type of file.
find : find matching files and run specified programs on them (optional).
finger : report user information (operates remotely only if a finger server is running on the remote host).
ftp : (file transfer protocol) a client for FTP servers.

G

grep : search files for regular expression matches.

H

haltsys : gracefully shutdown system (can only be run by root. halt in Linux).
head : display first 10 lines of a file.

J

join : display the combination (lines with common field) of two fields.

K

kill : send a signal to terminate a process.

ksh : invoke the korn shell.

L

line : read a specific line out of a file (shell script usage).

ln : create a link to a file/directory.

logname : gets your login name.

lpr : sends a request to printer.

lprint : prints on local printer.

lpstat : reports printer status.

lpq : same as above.

ls : lists the contents of directory.

M

mail : send and receive mail.

man : displays manual pages.

mesg : grant or deny permissions to receive messages from other users using the write command.

mkdir : create a new directory .

mknod : build a special file.

more : display file one page at a time.

mount : mount a storage device.

mv : move or rename a file.

N

news : display news item from NNTP servers.

nice : change priorities of processes.

nohup : run a command after logout (ignores hangup signals).

nroff : format files for printing.

nslookup : retrieve information from DNS servers.

O

od : displays a file in 8-based octals.

P

passwd : create or change login password.

paste : merge lines of files.

pr : format and print file.
ps : reports status of active processes.
pstat : report system status.
pwcheck : check /etc/passwd (default) file.
pwd : display current working directory.

R

rm : remove (erase) files or directories (unrecoverable).
rmdir : remove an empty directory.
rsh : invoke Restricted Bourne Shell.

S

sed : the stream editor.
set : assign value to variable.
setenv : assign value to environment variable.
sh : invoke Bourne shell.
sleep : suspend execution of a command for a given period.
sort : sort and merge files.
spell : find spelling errors.
split : split file to smaller files.
stty : set options for a terminal.
su : spawns a subshell with a different username, requires other user's password, unless you're root.
sum : compute checksums and number of blocks for files.

T

tabs : set tabs on a terminal.
tail : display last 10 lines of file.
tar : a simple compression tool that merges multiple files into a single one, originally made to make backing up materials on backup tapes easier.
tee : create a tee in a pipe.
telnet : access remote systems using the telnet protocol.
test : test various expressions and files.
time : display elapsed time (execution, process, and system times) for a command.
touch : change time/date stamps of files.
tr : substitutes sets of characters.
translate : translates files to different format.

troff : format files to phototyper.

true : return zero (true) exit status.

tset : set terminal mode.

tty : report a name of a terminal.

U

umask : set file-creation mode (permissions) mask.

umount : unmount a device.

uname : display the name of the current system.

uniq : report any duplicate line in a file.

units : convert numbers from one unit to another.

unzip : extract files from zip archive.

uptime : report system activity.

uucp : copy files between two unix systems (oldie but still beautiful).

uulog : report uucp status.

uuname : list uucp sites known to this site.

uudecode : decode to binary after "uuencode" transmission.

uencode : encode binary file for email transmission.

uustat : report status of uucp or cancel a job.

uupick : receive public files sent by uuto.

uuto : send files to another public Unix system.

uux : execute command to remote Unix system.

V

vi : a screen oriented (visual) editor (cool ,but Vim is better).

W

wall : sends message to all users (root only).

wait : await completion of background process.

wc : count lines, words, bytes etc' in one or more files.

who : report active users.

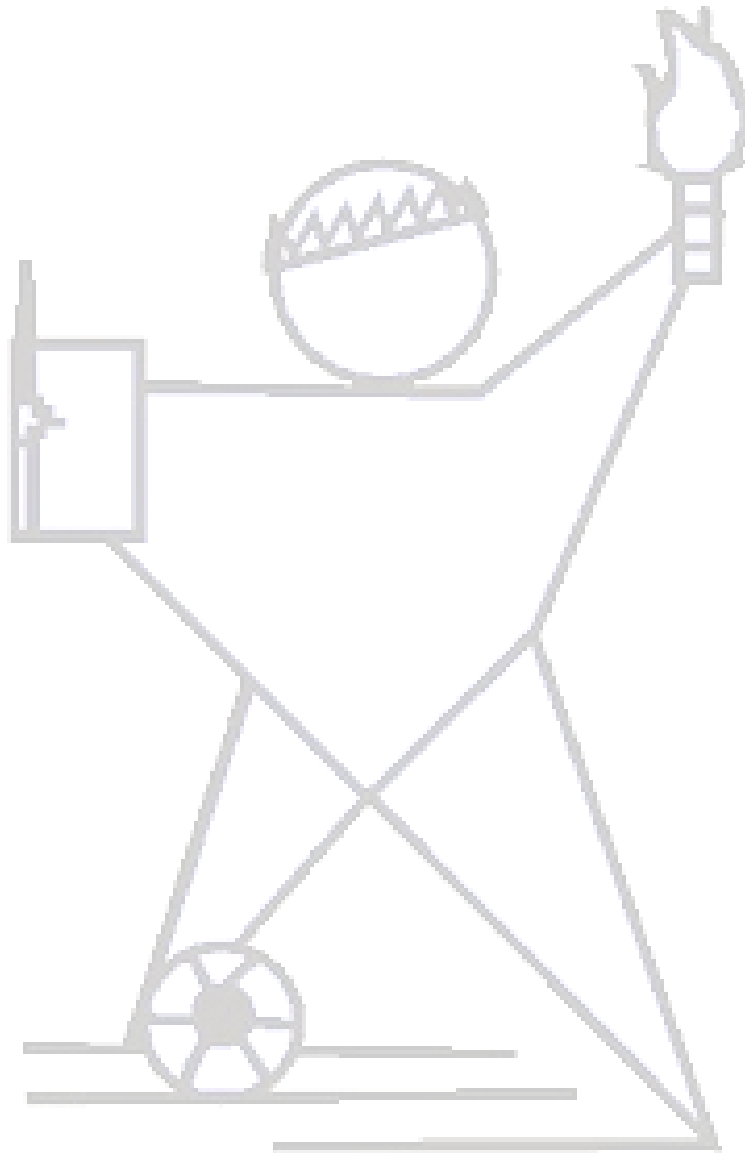
whois : search for user information.

write : send a message for another user (see mesg).

whoami : which user you are logged in as at the moment. If you, for example, switch to a different user, logname will show the original username you logged in as, and whoami will show the current user.

Z

zip : archive file or files in zip format.



Experiment No.2

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

You can perform math operations on [Bash shell variables](#). The `bash` shell has built-in arithmetic option. You can also use external command such as [expr](#) and [bc calculator](#). Arithmetic expansion and evaluation is done by placing an integer expression using the following format:

```
$((expression))  
$(( n1+n2 ))  
$(( n1/n2 ))  
$(( n1-n2 ))
```

There are following points to note down –

- There must be spaces between operators and expressions for example `2+2` is not correct, where as it should be written as `2 + 2`.
- Complete expression should be enclosed between ```, called inverted commas.
- You should use `\` on the `*` symbol for multiplication.
- **if...then...fi** statement is a decision making statement.

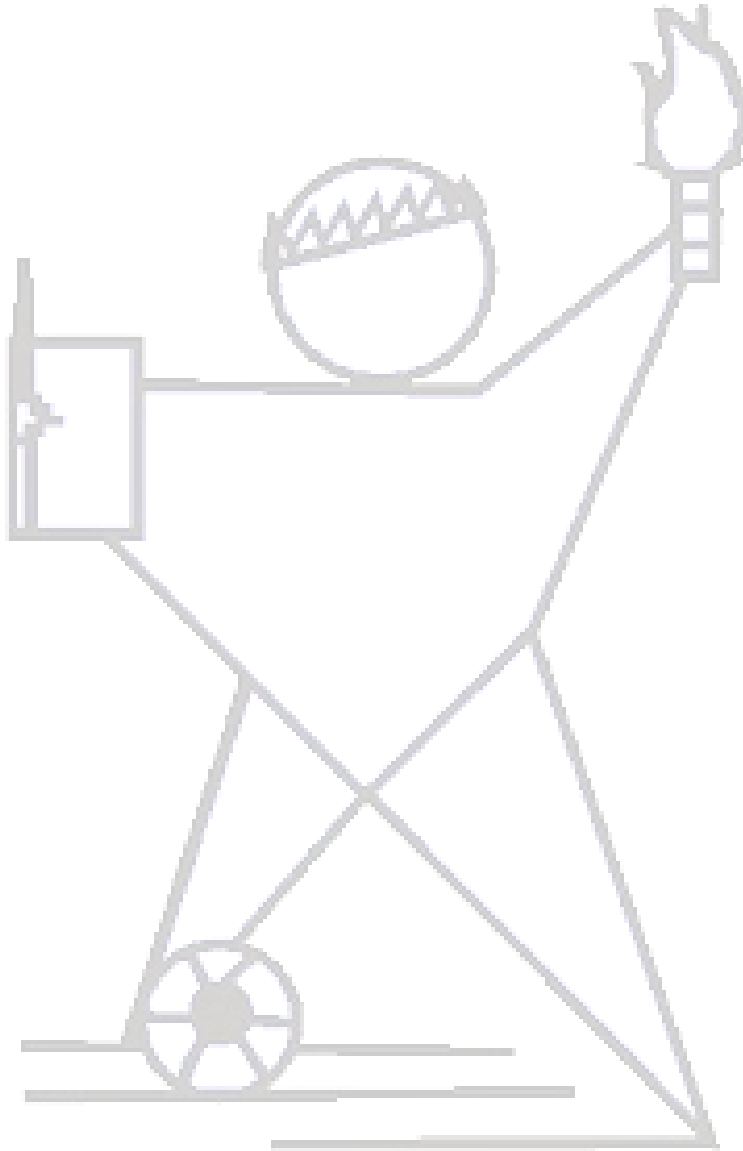
- `#!/bin/sh`
-
- `a=10`
- `b=20`
- `val=`expr $a + $b``

- `echo "a + b : $val"`
-
- `val=`expr $a - $b``
- `echo "a - b : $val"`
-
- `val=`expr $a * $b``
- `echo "a * b : $val"`
-
- `val=`expr $b / $a``
- `echo "b / a : $val"`
-
- `val=`expr $b % $a``
- `echo "b % a : $val"`
-
- `if [$a == $b]`
- `then`
- `echo "a is equal to b"`
- `fi`
-
- `if [$a != $b]`
- `then`
- `echo "a is not equal to b"`
- `fi`

- This would produce following result –

- `a + b : 30`
- `a - b : -10`
- `a * b : 200`
- `b / a : 2`

- $b \% a : 0$
- a is not equal to b



Experiment No.3

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

The **wc** (word count) command in Unix/Linux operating systems is used to find out number of newline count, word count, byte and characters count in a files specified by the file arguments. The syntax of **wc** command as shown below.

```
# wc [options] filenames
```

The following are the options and usage provided by the command.

```
wc -l : Prints the number of lines in a file.  
wc -w : prints the number of words in a file.  
wc -c : Displays the count of bytes in a file.  
wc -m : prints the count of characters from a file.  
wc -L : prints only the length of the longest line in a file.
```

Algorithm:

- 1.Input the name of the file.
- 2.Count the words of the given file.
- 3.Count the characters of the given file.
- 4.Count the lines of the given file.
- 5.Print the result.

Coding:

```
echo Enter the filename  
read wlcc.txt
```

```
w= 'cat $wlcc.txt | wc -w'
```

```
c= 'cat $wlcc.txt | wc -c'
```

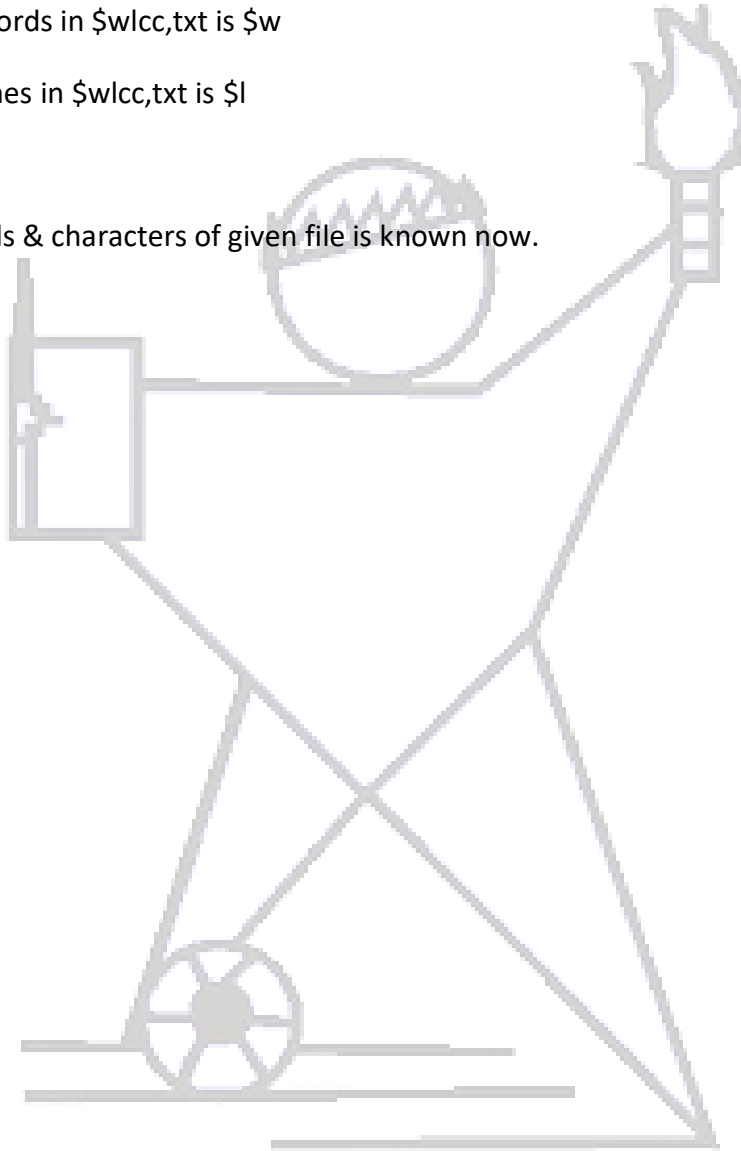
```
l= 'grep -c "." $wlcc.txt'
```

echo Number of characters in \$wlcc.txt is \$c

echo Number of words in \$wlcc.txt is \$w

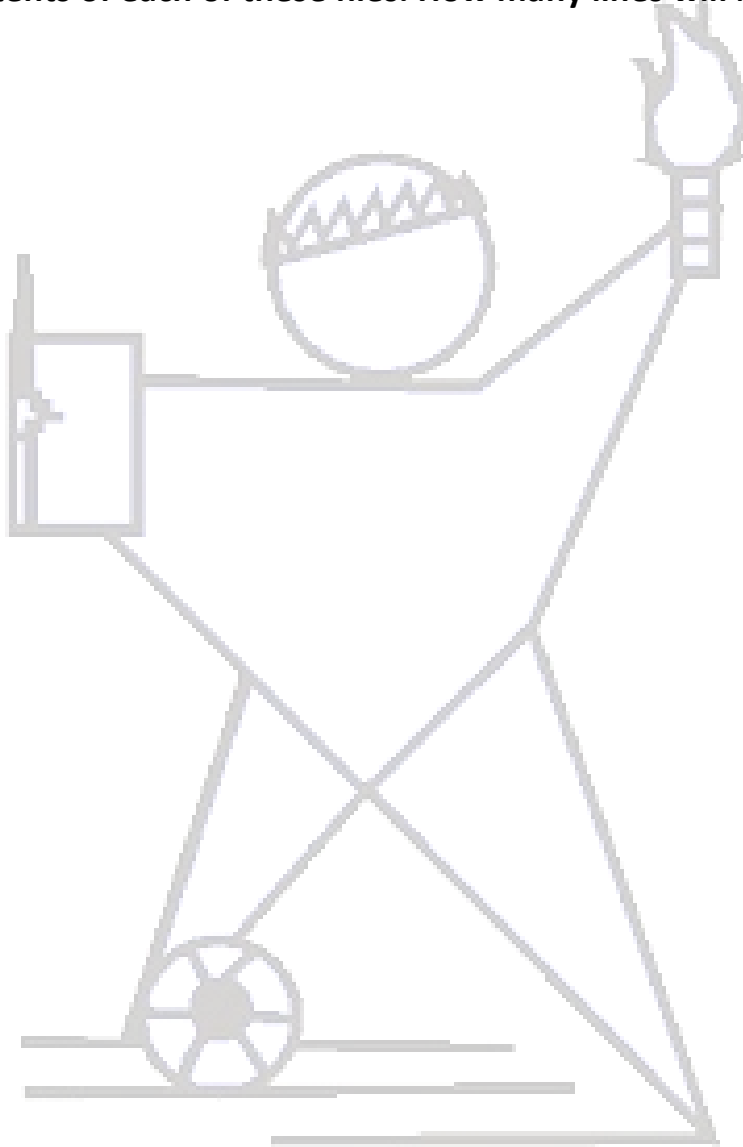
echo Number of lines in \$wlcc.txt is \$l

Result: Lines, words & characters of given file is known now.



Experiment No. 4

Aim : 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?



Experiment No.5

Aim : 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

Joins the lines of two files which share a common field of data.

[join syntax](#)

```
join [OPTION]... FILE1 FILE2
```

For each pair of input lines with identical join fields, write a line to standard output. The default join field is the first, delimited by [whitespace](#). When *FILE1* or *FILE2* (not both) is -, read [standard input](#).

[Options](#)

-a FILENUM	also print unpairable lines from file <i>FILENUM</i> , where <i>FILENUM</i> is 1 or 2 , corresponding to <i>FILE1</i> or <i>FILE2</i> .
-e EMPTY	replace missing input fields with <i>EMPTY</i>
-i, --ignore-case	ignore differences in case when comparing fields
-j FIELD	equivalent to "-1 <i>FIELD</i> -2 <i>FIELD</i> ".
-o FORMAT	obey <i>FORMAT</i> while constructing output line.
-t CHAR	use <i>CHAR</i> as input and output field separator.
-v FILENUM	like -a FILENUM , but suppress joined output lines.
-1 FIELD	join on this <i>FIELD</i> of file 1 .
-2 FIELD	join on this <i>FIELD</i> of file 2 .

--check-order	check that the input is correctly sorted, even if all input lines are pairable.
--nocheck-order	do not check that the input is correctly <u>sorted</u> .
--header	treat the first line in each file as field headers, print them without trying to pair them.
--help	display a help message and exit.
--version	display version information and exit.

Unless **-t CHAR** is given, fields are separated by leading blank spaces; otherwise, fields are separated by *CHAR*. Any *FIELD* is a field number counted from 1.

If *FORMAT* is the keyword **auto**, then the first line of each file determines the number of fields output for each line.

FILE1 and *FILE2* must be sorted on the join fields. The sort command can accomplish this. If the input is not sorted and some lines cannot be joined, a warning message will be given.

[Join examples](#)

If we have a file, **myfile1.txt**, whose contents are:

1 Amit

2 Rakesh

3 Neeraj

4 Lokesh

5 Ram

...and another file, **myfile2.txt**, whose contents are:

1 Amit

2 Namit

3 Rakesh

4 Suresh

5 Tony

The common fields are the fields, We can join the contents using the following command:

```
join myfile1.txt myfile2.txt
```

...which outputs the following to standard output:

1.Amit

2.Rakesh

If we wanted to create a new file with the joined contents, we could use the following command:

```
join myfile1.txt myfile2.txt > myjoinedfile.txt
```

...which directs the output into a new file called **myjoinedfile.txt**, containing the same output as the example above

Experiment No.6

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

Every file has an unique inode number, using that we can identify that file. Create two files with similar name. i.e one file with a space at the end.

An inode identifies the file and its attributes such as file size, owner, and so on. A unique [inode number](#) within the file system identifies each inode. But, why to delete file by an inode number? Sure, you can use [rm command](#) to delete file. Sometime accidentally you creates filename with control characters or characters which are unable to be input on a keyboard or special character such as ?, * ^ etc.

PROGRAM:

```
read fname    #file name to be searched.
p=`pwd`
cd /
l=`find -name $fname`
l=`echo $l | ls -li`
set $l
echo "inode is $1"
cd $p
```

OUTPUT:

```
inode is 11483864
```

Experiment No.7

Aim : Study & use of the Command for changing file permissions.

A file has three types of permissions (read, write and execute) and three sets of users (user (owner), group and other (world)) with specific permissions. Only file's owner or the superuser can change a file's permissions.

Chmod

chmod

Change file access permissions.

SYNOPSIS

chmod [-R] mode FILE...

-R, --recursive

change files and directories recursively

Two ways of expressing mode:

1. As an assignment, addition or subtraction of privileges for a specified set of users. First specify the set of users: **u** = user; **g** = group; **o** = other; **a** = all. Next specify the operator: +, -, or =. Finally list which permissions are being changed or set.
2. {u | g | o | a } {+ | - | =} {rwx}*
3.
4. `chmod -R g+w code # recursively add group write permissions the code directroy`
5. `chmod o-rwx * # remove read, write, execute for other on all files`
6. Using a three digit octal number assignment. The three digits correspond to user, group and other. The value of each digit is as if the rwx permissions were a three digit binary number. (read = 4, write = 2, execute = 1)

Permission Pattern	Octal Number
rwxr-x—	750
rw-r-r—	644

Permission Pattern	Octal Number
rw-rw-r--	664

The first approach, with either the + or - operator, is usually preferred when operating recursively on a directory tree. This is because some file or directories may have special permissions, thus it is better to add or remove permissions rather explicitly setting them. Also directories require the execute bit set to use the directory. Non-executable files should not have the execute bit set.

Directory Permissions

- Read directory permission grants the ability to view a file.
- Write directory permission grants the ability to add, change or remove files from the directory, assuming the file permissions do not conflict.
- Execute directory permission grants the ability to list (**ls**) the directory content of search (**find**) for files in the directory.
- Desirable permission settings include: 755, 750 or 700.

Note

A file is as secure as its directory. The execute permission is not as intuitive as the other two. If this permission is removed, you can't:

1. **cd** to the directory.
2. use that directory in a pathname.
3. create or remove files in the directory.

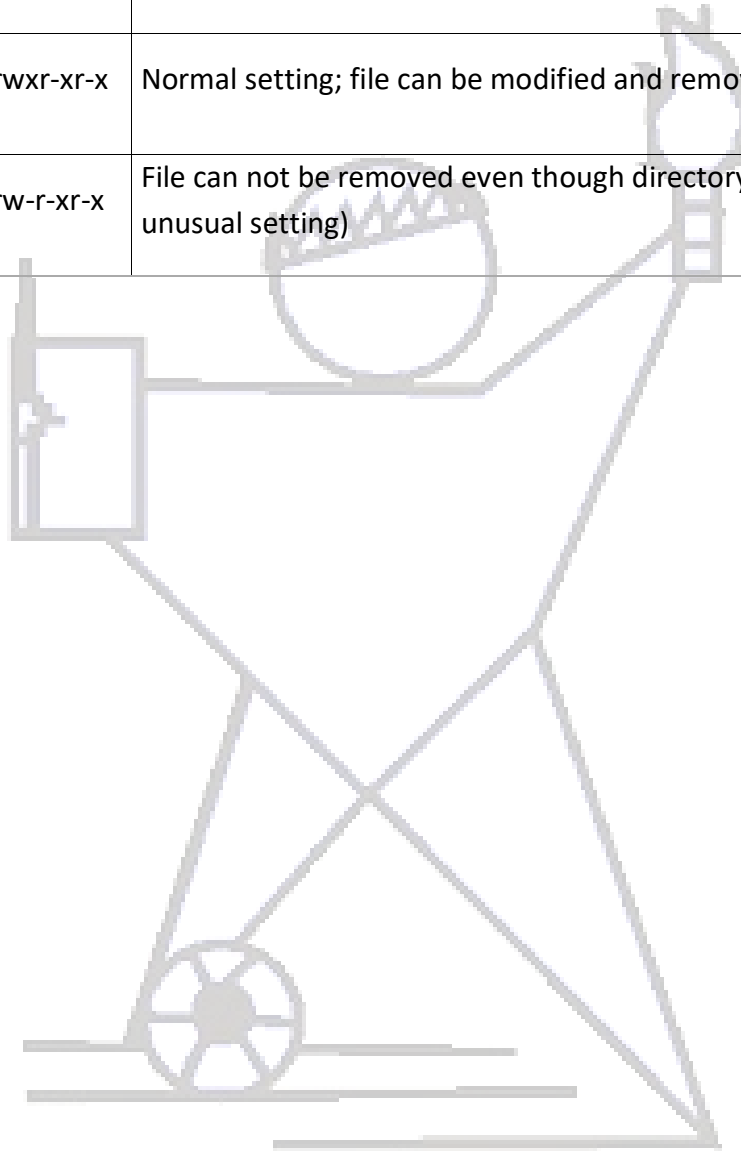
This means that to be able to create or remove files, the directory must have both write and execute permission. Mere write permission is not enough.

How a Directory Influences File Permissions

Examining only the user category

File	Directory	Significance
r--r--	rwxr-xr-x	A write-protected file; can't be modified but can be removed.

File	Directory	Significance
rw-r-- r--	r-xr-xr-x	write-protected directory; file can't be removed but can be modified.
r--r-- r--	r-xr-xr-x	A write-protected file and directory; file can't be modified or removed.
rw-r-- r--	rw-xr-xr-x	Normal setting; file can be modified and removed.
rw-r-- r--	rw-r-xr-x	File can not be removed even though directory is writable. (An unusual setting)



Experiment No.8

Aim : 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

Using the given code which displays the output of a program on the monitor and copies the information about the number of users using the system at present on a file called usere.ux into a file:

Use the **tee** command.

```
who | tee -a usere.ux
```

tee -a : display the o/p on monitor and save (append) in the file also

DESCRIPTION

The tee command reads standard input, then writes the output of a program to standard output and simultaneously copies it into the specified file or files.

SYNOPSIS

```
tee [ -a ] [ File ... ]
```

OPTIONS

-a

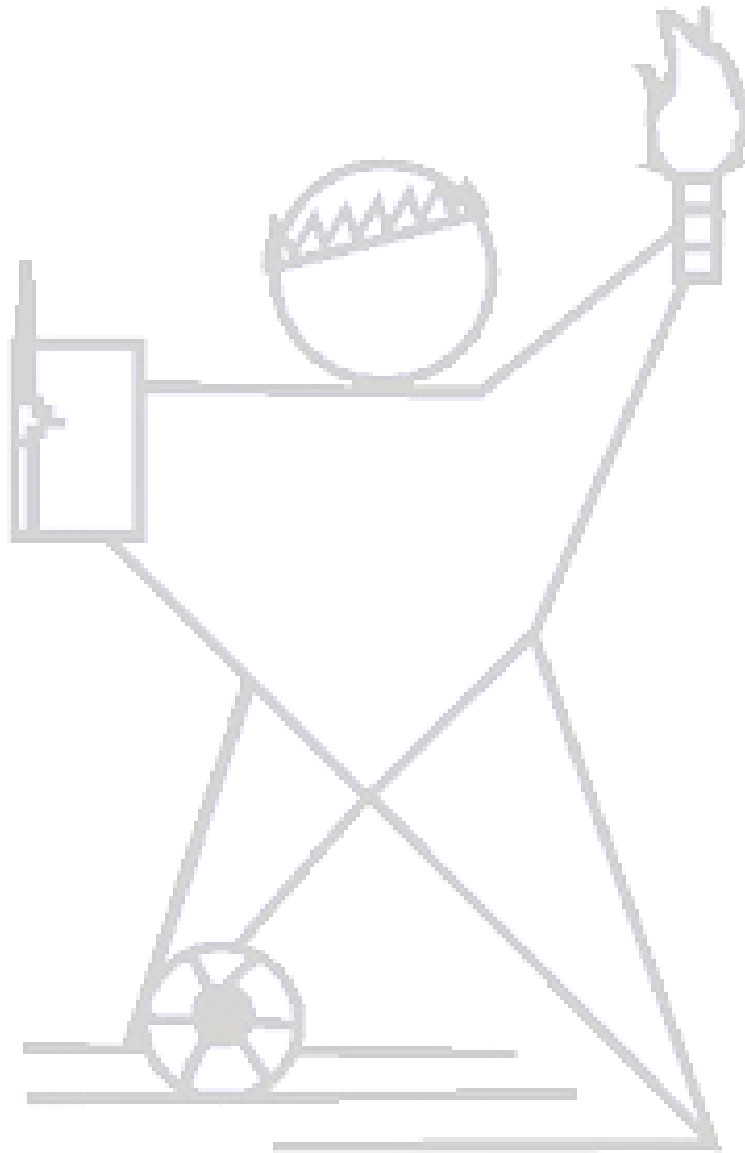
Adds the output to the end of File instead of writing over it.

EXAMPLES

1. _____ Write output to stdout, and also to a file

- The following command displays output only on the screen (stdout).
- **\$ ls**
- The following command writes the output only to the file and not to the screen.
- **\$ ls > file**
- The following command (with the help of tee command) writes the output both to the screen (stdout) and to the file.

- \$ ls | tee file1.txt



Experiment No.9

Aim : Execute shell commands through vi editor.

Running UNIX shell commands from vi

You can run UNIX commands and see their output without leaving vi. You can also insert the output of a UNIX command into the file you that are editing.

To run a single UNIX command use the command:

`:!UNIX_command`

You can start a shell from within vi and use it as you would your usual UNIX environment, then exit the shell and return to vi

To start up a shell enter the command:

`:sh`

The type of shell that is started is determined by the \$SHELL variable. You can specify that some other shell is to be started by setting the vi [shell option](#)

Return to using vi by entering the command **exit** or Ctrl-D

To insert the output from a UNIX command into a file, immediately after the cursor:

`:r!command`

For example, this facility would be very useful if you were using vi to document a UNIX command and you wanted to include examples of the output from this command.

Experiment No.10

Aim : Installation, Configuration & Customizations of Unix/Linux.

Installation:

Linux makes more effective use of PC hardware than MS-DOS, Windows or NT, and is accordingly less tolerant of misconfigured hardware. There are a few things you can do before you start that will lessen your chances of being stopped by this kind of problem.

First, collect any manuals you have on your hardware -- motherboard, video card, monitor, modem, etc. -- and put them within easy reach.

Second, gather detailed information on your hardware configuration. One easy way to do this, if you're running MS-DOS 5.0, or up, is to print a report from the Microsoft diagnostic utility msd.exe (you can leave out the TSR, driver, memory-map, environment-strings and OS-version parts). Among other things, this will guarantee you full and correct information on your video card and mouse type, which will be helpful in configuring X later on.

Third, check your machine for configuration problems with supported hardware that could cause an un-recoverable lockup during Linux installation.

- It is possible for a DOS/Windows system using IDE hard drive(s) and CD ROM to be functional even with the master/slave jumpers on the drives incorrectly set. Linux won't fly this way. If in doubt, check your master-slave jumpers!
- Is any of your peripheral hardware designed with neither configuration jumpers nor non-volatile configuration memory? If so, it may require boot-time initialization via an MS-DOS utility to start up, and may not be easily accessible from Linux. CD-ROMs, sound cards, Ethernet cards and low-end tape drives can have this problem. If so, you may be able to work around this with an argument to the boot prompt; see the [Linux Boot Prompt HOWTO](#) for details).
- Some other operating systems will allow a bus mouse to share an IRQ with other devices. Linux doesn't support this; in fact, trying it may lock up your machine. If you are using a bus mouse, see the [Linux Bus Mouse HOWTO](#), for details.

If possible, get the telephone number of an experienced Linux user you can call in case of emergency. Nine times out of ten you won't need it, but it's comforting to have.

Budget time for installation. That will be about one hour on a bare system or one being converted to all-Linux operation. Or up to three hours for a dual-boot system (they have a much higher incidence of false starts and hangups).

CONFIGURATION:

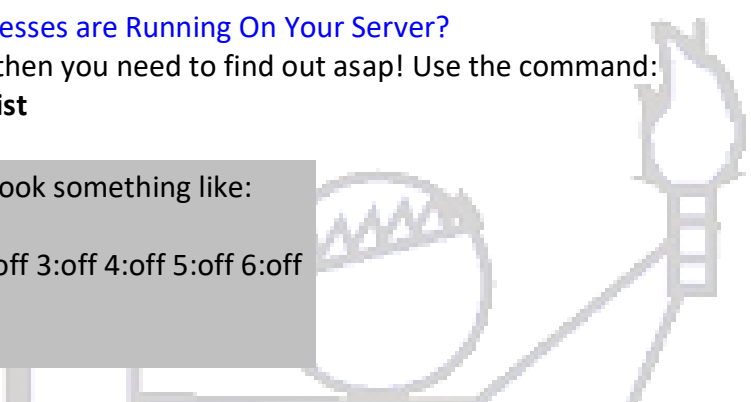
First off, I should mention that this guide is best used when in front of your Linux computer, with an open xterm session. The exact commands come from RedHat 7.1+ related server, but apply to most Linux servers.

Step 1. Which Processes are Running On Your Server?

If you don't know, then you need to find out asap! Use the command:
/sbin/chkconfig --list

The output would look something like:

```
...  
httpd 0:off 1:off 2:off 3:off 4:off 5:off 6:off  
telnet: off  
...
```



The above command will give you a long list of processes with info beside them like "off". Any process with the word "off" next to it can be assumed disabled by default during startup. You should look for your processes that are usually needed for running a webserver like httpd, telnet, wu-ftp, mysqld. All of these should be "on" by default.

Step 2. Get Processes Started

Starting up your webserver (httpd), mysql (mysqld), sendmail, etc. is easy so long as you follow the directions from the steps below.

For your webserver and mysql, you can enable these things right away for use during this session.

Change to the initialization (aka init) directory:

cd /etc/rc.d/init.d/

This directory (when listed) shows all processes you can start like httpd and mysqld. For now let's start our web server with the command:

./httpd start

You should then see:

Starting httpd: [OK]

Now enable your webserver (httpd) for **ALL future STARTUPS!**

1. Edit the config files as applied to the "rc" directory of your choice. Remember that all resource files activated at different run times are in different rc.d directories. For instance, when your server is loaded at runtime level 5 (usual) then all the resources under rc5.d are activated. Change directory to:

/etc/rc.d/rc5.d

Remember that the *rc5.d* is a resource directory (under /etc) for run level 5... etc.

You **edit** files in these directories to control what occurs at different run levels. Files with a prefix of **K** are NOT installed to run at startup. Files with **S** are ready to run at startup. Example names: K74ypserv or S14nfslock.

You can always use something like the command:

`/sbin/chkconfig --add httpd`

to add the web server to the future startups. However, I prefer doing my change manually.

2. You can manually force this by simply using a command like:

mv K15httpd S15httpd

Summary for those needing one... You now should have your webserver started and ready as default for all future starts with:

- `./httpd start`
- `mv k15httpd s15httpd`

Step 3. What About telnet and ftp?

Ok, you're smart enough to have noticed that following the steps above you can **not** get telnet or ftp started. That's because **they are not part of the initd process**, but rather the **xinetd** process. The xinetd process handles the startup of all of your network related protocols etc.

1st Start telnet first by changing directories to xinetd:

cd /etc/xinetd.d/

Next type **ls** to list all of the processes that can be configured. You'll notice for instance the file telnet.

2nd Edit the telnet file and change two lines:

default: on

...

disabled = no

These lines are not adjacent, but usually the first and last lines of the configuration file (in our

case *telnet*). You need to **edit all configuration files** that apply to things you're trying to start. Many processes come by default turned off and disabled = yes. You can edit files like *telnet*, *wu-ftp*, etc.

3rd Once you have edited and saved the files with the default on and disabled = no, you can force an automatic restart of the *xinetd* to load without rebooting:

/etc/rc.d/init.d/xinetd restart

Finally, you should see:

Stopping *xinetd*: [OK] Starting *xinetd*: [OK]

Believe it or not, following all of this you should now have running:

- *httpd* (webserver)
- *telnet*

Now check to see what processes you have running again by using:

/sbin/chkconfig --list

or use the long "process" *ps* command like: *ps -e | grep http*.

You can use these same steps above to get *mysql* and *ftp* running. Replace *httpd* with *mysql*, and *telnet* with *wu-ftp*. Always remember there is a difference between configuration and startup files under *initd* and *xinetd*.

Customizations:

FEATURES	FUNCTION	sh	csh	ksh
Job control	Allows processes to be run in the background	No	Yes	Yes
History substitution	Allows previous commands to be saved, edited, and reused	No	Yes	Yes
File name completion	Allows automatic completion of partially typed file name	No	Yes	Yes
Command line	Allows the use of an editor to	No	No	Yes

editing	modify the command line text			
Command aliasing	Allows the user to rename commands	No	Yes	Yes

Choosing your shell

It is possible to invoke any available shell from within another shell. To start a new shell, you can simply type the name of the shell you want to run, ksh, csh, or sh.

It is also possible to set the default startup shell for all your future sessions. The default shell for your account is stored in the system database `/etc/passwd`, along with the other information about your account. To change your default shell, use the `chsh` command. The `chsh` command requires one argument, the name of the shell you want as your default. To change your default shell to the C shell, you could enter the command

```
chsh /bin/csh
```

On ISU's HP-UX system, the available shells are `/bin/sh`, `/bin/posix/sh`, and `/bin/csh`. The default shell for accounts is `/bin/posix/sh`, which is, for all practical purposes, the same as ksh.

Default file access permissions

Whenever you create a file or directory in a Unix filesystem, the newly created file or directory is stamped with a default set of permissions. That default set of permissions is stored in a variable called the *umask*. You can change the value of *umask* to suit your preferences. To see the current value of the *umask* variable, enter the shell command:

```
umask
```

The *umask* is stored as an octal (base 8) number, that defines which permissions to deny. As you recall, three kinds of file permissions (read, write, and execute) are given for each of three classes of users (owner, group, and others). Each of the nine permissions is specified as a zero (allow access), or a one (deny access).

	User	Group	Other	
	rwX	rwX	rwX	
BIT PATTERN:	000	010	010	Denies write access to group, others
OCTAL:	0	2	2	
BIT PATTERN:	000	111	111	Denies all access to group, others
OCTAL:	0	7	7	

To set your umask to deny write permission to group and others, use the command

```
umask 022
```

To deny all access to group and others, use the command

```
umask 077
```

Some versions of Unix provide a more user-friendly way of specifying your umask. In HP-UX sh-posix (or ksh), you are allowed to specify the access permissions in manner of the chmod command. The command

```
umask u=rwx,g=r,o=r
```

would set the umask to deny write and execute permissions to the group, and to others. That kind of command syntax will not work in HP-UX's C shell or Bourne shell. The HP-UX posix shell also allows the use of the command

```
umask -S
```

to print your umask setting in a more readable fashion.

Experiment No.11

Aim : Write a shell script that accepts any number of arguments and prints them in the reverse order.

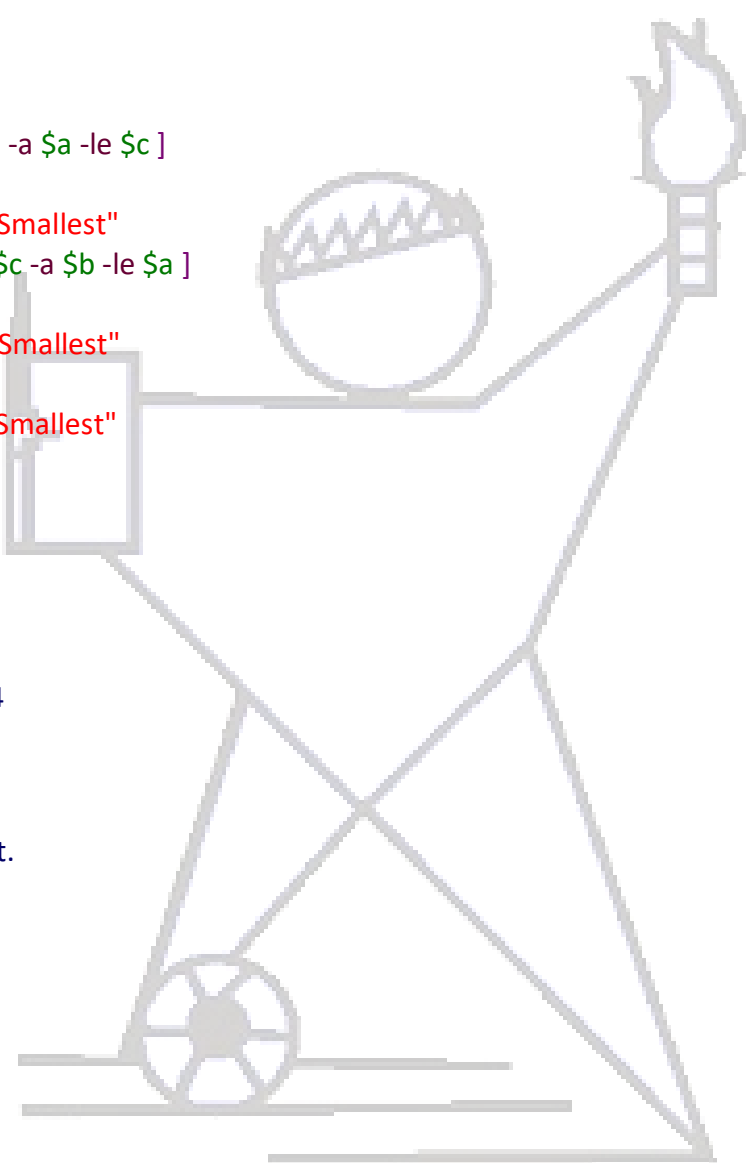
```
a=$#  
echo "Number of arguments are" $a  
x=$*  
c=$a  
res=""  
while [ 1 -le $c ]  
do  
c=`expr $c - 1`  
shift $c  
res=$res' '$1  
set $x  
done  
echo Arguments in reverse order $res
```

Output

```
sh 1prg.sh a b c  
No of arguments arguments are 3  
Arguments in reverse order c b a
```

Experiment No.12

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



```
1. read a b c
2. if [ $a -le $b -a $a -le $c ]
3. then
4. echo "$a is Smallest"
5. elif [ $b -le $c -a $b -le $a ]
6. then
7. echo "$b is Smallest"
8. else
9. echo "$c is Smallest"
10. fi
```

stdin:

a=7 b=5 c=4

Output:

C is smallest.

Experiment No.13

Aim : 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.

/* In this program the maximum waiting time is 1 minute after that it will terminate */

```
echo " enter the login name of the user "  
read name  
period=0  
until who| grep -w"$name 2> /dev/null  /* search for the user error are send to special file */  
do  
    sleep 60  
    period=`expr $period + 1`  
    if [ $period -gt 1 ]  
    then  
        echo " $name has not login since 1 minute "  
        exit  
    fi  
done  
echo " $name has now logged in "
```

Output:

1) \$sh 10b.sh

Enter the login name of the user

mca5

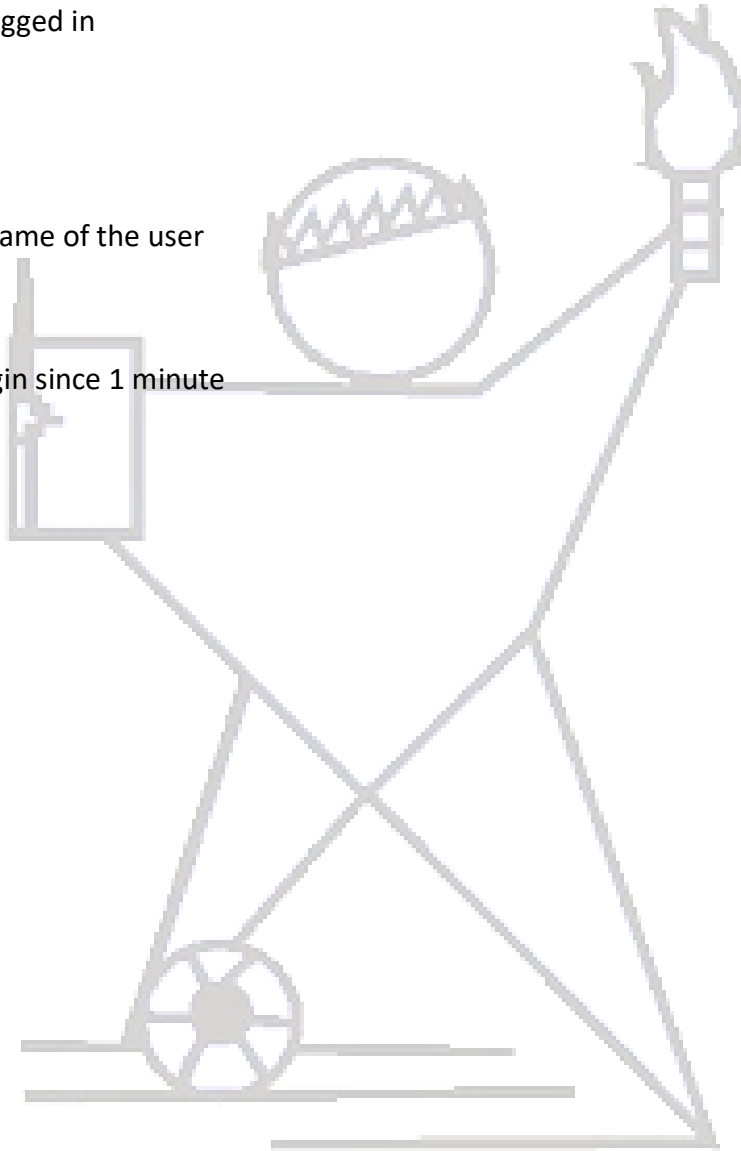
mca5 has now logged in

2) \$sh 10b.sh

Enter the login name of the user

mca6

mca6 has not login since 1 minute



Experiment No.14

Aim : Installation of SAMBA, APACHE, TOMCAT.

Installation of SAMBA:

Install samba using Yum.

yum install samba if ubuntu, # apt-get install samba

after Installation you have to edit config file,

vi /etc/samba/smb.conf

Shift+G , Go to Last line

[share]

path = /share

hosts allow = 192.168.1.

browsable = yes

read list = fileserv

write list = +fileserv

sharing folder name

User name - fileserv

+ refers to user, fileserv groups

Save this file.

smbpasswd -a fileserv

```
# chcon -Rt samba_share_t /share
```

```
# /etc/init.d/smbd restart
```

Then go to windows sysem, using samba server ip you can access your file server with authentication.

If you want to multiple folders to share, again add these lines each folder.

```
[share]
```

```
path = /share2
```

```
hosts allow = 192.168.1.
```

```
browsable = yes
```

```
read list = fileservr
```

```
write list = +fileservr
```

sharing folder name

User name - fileservr

+ refers to user, fileservr groups

Installation of APACHE:

Manual installation offers several benefits:

- backing up, reinstalling, or moving the web server can be achieved in seconds (see [8 Tips for Surviving PC Failure](#))
- you have more control over how and when Apache starts
- you can install Apache anywhere, such as a portable USB drive (useful for client demonstrations).

Step 1: configure IIS, Skype and other software (optional)

If you have a Professional or Server version of Windows, you may already have IIS installed. If you would prefer Apache, either [remove IIS as a Windows component or disable its services](#).

Apache listens for requests on TCP/IP port 80. The default installation of Skype also listens on this port and will cause conflicts. To switch it off, start Skype and choose Tools > Options > Advanced > Connection. Ensure you untick "Use port 80 and 443 as alternatives for incoming connections".

Step 2: download the files

We are going to use the unofficial Windows binary from [Apache Lounge](http://www.apacheounge.com/download/). This version has performance and stability improvements over the official Apache distribution, although I am yet to notice a significant difference. However, it is provided as a manually installable ZIP file from www.apacheounge.com/download/

You should also [download and install the Windows C++ runtime from Microsoft.com](http://www.microsoft.com/windows/win7/compat/Windows7/Windows7Cplusplus.aspx). You may have this installed already, but there is no harm installing it again.

As always, remember to virus scan all downloads.

Step 2: extract the files

We will install Apache in C:\Apache2, so extract the ZIP file to the root of the C: drive.

Apache can be installed anywhere on your system, but you will need to change the configuration file paths accordingly...

Step 3: configure Apache

*Apache is configured with the text file **conf\httpd.conf** contained in the Apache folder. Open it with your favourite text editor.*

Note that all file path settings use a '/' forward-slash rather than the Windows backslash. If you installed Apache anywhere other than C:\Apache2, now is a good time to search and replace all references to "c:/Apache2".

Installation of TOMCAT:

In order to use Tomcat for developing web applications, you must first install it (and the software it depends on). The required steps are outlined in the following subsections.

JDK

Tomcat 7.0 was designed to run on Java SE 6.

Compatible JDKs for many platforms (or links to where they can be found) are available at <http://www.oracle.com/technetwork/java/javase/downloads/index.html>.

Tomcat

Binary downloads of the **Tomcat** server are available from <http://tomcat.apache.org/>. This manual assumes you are using the most recent release of Tomcat 7. Detailed instructions for downloading and installing Tomcat are available [here](#).

In the remainder of this manual, example shell scripts assume that you have set an environment variable `CATALINA_HOME` that contains the pathname to the directory in which Tomcat has been installed. Optionally, if Tomcat has been configured for multiple instances, each instance will have its own `CATALINA_BASE` configured.

Ant

Binary downloads of the **Ant** build tool are available from <http://ant.apache.org/>. This manual assumes you are using Ant 1.8 or later. The instructions may also be compatible with other versions, but this has not been tested.

Download and install Ant. Then, add the `bin` directory of the Ant distribution to your `PATH` environment variable, following the standard practices for your operating system platform. Once you have done this, you will be able to execute the `ant` shell command directly.

CVS

Besides the required tools described above, you are strongly encouraged to download and install a *source code control* system, such as the **Concurrent Version System** (CVS), to maintain historical versions of the source files that make up your web application. Besides the server, you will also need appropriate client tools to check out source code files, and check in modified versions.

Detailed instructions for installing and using source code control applications is beyond the scope of this manual. However, CVS server and client tools for many platforms (along with documentation) can be downloaded from <http://www.cvshome.org/>.



Experiment No.15

Aim : Implementation of DNS, LDAP services,

There are two ways of dns that can be configured via ldap, client side and server side. The first, client side, is using the name server switch to access the dns entries in the ldap database. This means that only clients that modify their `/etc/nsswitch.conf` file will see the dns entries from ldap. The second way to do it is to use ldap as a backend for bind or tinydns. There are some projects going on about this subject and i will describe them below.

Using nss

When using nss to access (additional) host entries, please take note that only "friendly" machines (e.g. machines that you know of and whose configuration you can control) can use this service. It might be useful for intranet host lookups that change often, but it cannot be used to distribute your webserver's virtual hostnames to the world. Note that also the **nslookup** command bypasses both `/etc/hosts` and ldap, so it cannot be used to check if your setup is working. Use something like **host** or **ping** instead, which does a lookup with the `inet_ghostbyname()` function.

Configuration

To have the name server switch use ldap for dns lookups it must be configured with `nss_ldap`. How to set up `nss_ldap`, you can find in [Section 2](#). Here i will assume you have a working `nss_ldap` configuration. The dns lookups of nss are controlled with the `hosts` line in `/etc/nsswitch.conf`. It is very unlikely that you do not already have a `hosts` line. Most probably it will contain the `files` and `dnssentries`. You should add `ldap` to it like this:

```
hosts:          files, dns, ldap
```

Think well about the order in which you specify these! It is advised always to put `files` as the first entry. Then, if you want ldap to override your local dns server, you have to make sure that the ip of the ldap server can be found in the `/etc/hosts` file. If not, you will have a nice recursive lookup going. -- You want to look up a host, it's not in files, so we try to contact the ldap server, whose ip we don't know, so we try to look it up in files, where we cannot find it, so we try to contact the ldap server -- get the point? You could bypass this problem entirely by referring to your ldap server with an ip number instead of a hostname (in `/etc/ldap.conf`, that is.)

Schema

The schema used for this, and similar services, can be found in [RFC 2307](#). Entries used for mapping names to ipnumbers are in an objectclass *ipHost*. The name part of the mapping is given in the attribute *cn*, while the ip part lives in *ipHostNumber*. A typical Idif entry would therefore look like this:

```
dn: cn=somehostname.mydomain.com,ou=Network,o=YourOrg,c=NL
objectclass: top
objectclass: ipHost
cn: somehostname.internal.example.com
ipHostNumber: 10.1.5.13
```

Of course, the usual restrictions and possibilities that come with dns apply.

Using bind

There are a few possibilities with bind or tinydns nowadays, but imho none of them is a "real" solution (yet). I must say, however, that i have no experience with any of them. They are listed below.

Bind patch

David Storey is working on a patch for Bind, which makes it get its data directly from ldap. This means that every time a request is performed on the bind daemon, it does a lookup in ldap. At this time, his future plans were: (Taken from the source) to have at least two modes of operation: cached and dynamic. Cached mode operates just like an rbtldb by loading the entire zone into memory and reloading whenever the server is HUP'ed. Dynamic mode is much like it is now: every request means an LDAP lookup. For up to date information you should check out the [sources](#).

ldap2dns

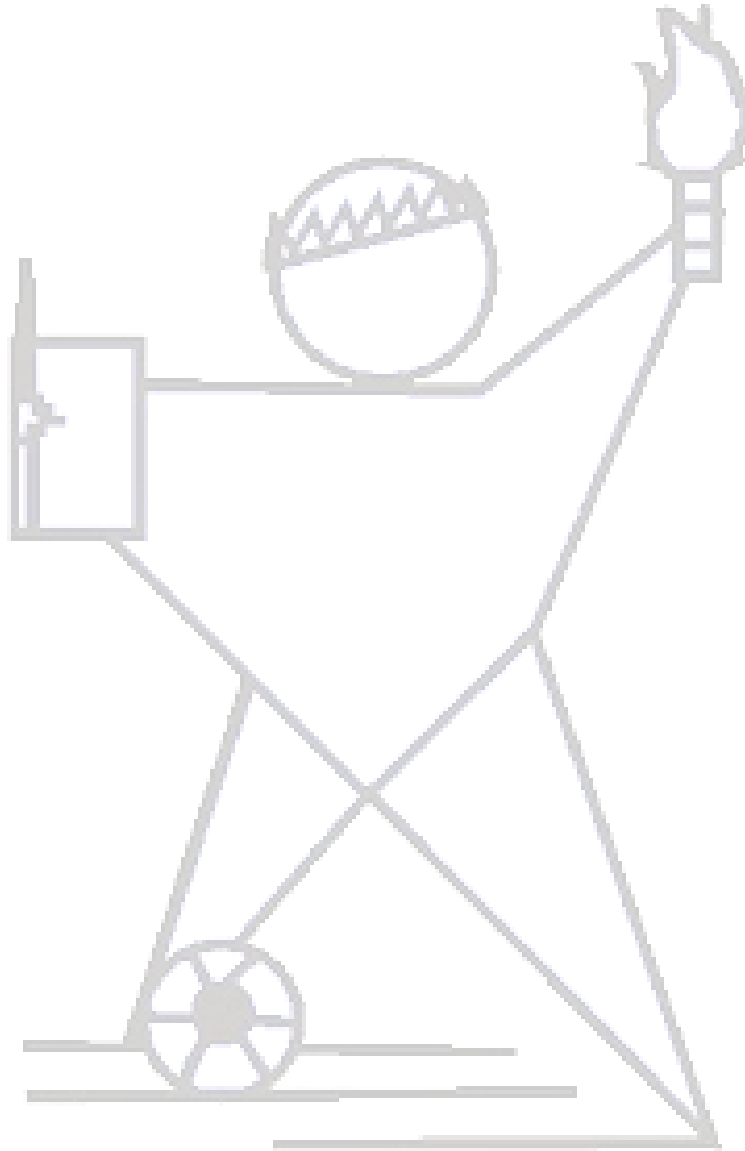
Taken entirely from their website:

ldap2dns is a program to create DNS records directly from a LDAP directory. It can and should be used to replace the secondary name-server by a second primary one. ldap2dns helps to reduce all kind of administration overhead. No more flat file editing, no more zone file editing. After having installed ldap2dns, the administrator only has to access the LDAP directory. If he desires he can add access control for each zone, create a webbased GUI and add all other kind of zone and resource record information without interfering with the DNS server. ldap2dns is designed to write binary data.cdb files used by tinydns, but also may be used to write .db-files used by named.

The projects homepage is [here](#).

ispman

ispman is a perl-based isp management package. It uses an ldap database backend for it's configuration. It can do lot's of things, so you might check out what you need exactly. It's at ispman.org.



Experiment No.16

Aim : Study & installation of Firewall & Proxy server[http:](http://)

Proxy server is used to share controlled internet in the network. Proxy server works as intermediate computer between end users and internet. Through proxy server we can decide who is going to access what from internet in our network. We can also put a time cap on internet uses.

For demonstration purpose we will use CCProxy server. CCProxy is a lightweight proxy server. At the time of preparing this tutorial CCProxy is available in two versions; free and paid. There are no differences between free and paid version beside the number of PCs it supports. Free version is limited with three computers. You can download free CCProxy sever from its developer site.

CCProxy server works in server/clients mode.

Server: - Server is the computer that is directly connected with the internet.

Clients: - Clients are the computers those access internet through the server.

In [computing](#), a **firewall** is a [network security](#) system that monitors and controls the incoming and outgoing network traffic based on predetermined security rules.^[1] A firewall typically establishes a barrier between a trusted, secure internal network and another outside network, such as the Internet, that is assumed not to be secure or trusted.^[2] Firewalls are often categorized as either *network firewalls* or *host-based firewalls*. Network firewalls filter traffic between two or more networks; they are either [software appliances](#) running on general purpose hardware, or hardware-based [firewall computer appliances](#). Host-based firewalls provide a layer of software on one host that controls network traffic in and out of that single machine.^{[3][4]} Firewall appliances may also offer other functionality to the internal network they protect, such as acting as a [DHCP](#)^{[5][6]} or [VPN](#)^{[7][8][9][10]} server for that network.^{[11][12]}

Installation of firewall:

Step 1: Downloading

Config Server Firewall is not currently available in Debian or Ubuntu repositories, and has to be downloaded from the ConfigServer's website.

```
wget http://download.configserver.com/csf.tgz
```

This will download CSF to your current working directory.

Step 2: Uncompressing

The downloaded file is a compressed from of tar package, and has to be uncompressed and extracted before it can be used.

```
tar -xzf csf.tgz
```

Step 3: Installing

If you are using another firewall configuration scripts, such as UFW, you should disable it before proceeding. Iptables rules are automatically removed.

UFW can be disabled by running the following command:

```
ufw disable
```

Now it is time to execute the CSF's installer script.

```
cd csf
```

```
sh install.sh
```

The firewall is now installed, but you should check if the required iptables modules are available.

```
perl /usr/local/csf/bin/csftest.pl
```

The firewall will work if no fatal errors are reported.

Note: Your IP address was added to the whitelist if possible. In addition, the SSH port has been opened automatically, even if it uses custom port. The firewall was also configured to have testing mode enabled, which means that the iptables rules will be automatically removed five minutes after starting CSF. This should be disabled once you know that your configuration works, and you will not be locked out.

Basic Configuration

CSF can be configured by editing its configuration file csf.conf in /etc/csf:

```
nano /etc/csf/csf.conf
```

The changes can be applied with command:

```
csf -r
```

Step 1: Configuring ports

The less access there is to your VPS, the more secure your server is. However, not all ports can be closed as the clients must be able to use your services.

The ports opened by default are the following:

```
TCP_IN = "20,21,22,25,53,80,110,143,443,465,587,993,995"
```

```
TCP_OUT = "20,21,22,25,53,80,110,113,443"
```

```
UDP_IN = "20,21,53"
```

```
UDP_OUT = "20,21,53,113,123"
```

Installation of Proxy server[http:](http://)

1.Installing the proxy

To install Squid type the following command in a terminal:

```
sudo aptitude install squid
```

2. Configuring the proxy

Configuration of Squid is done by editing the following file: **/etc/squid/squid.conf**

To edit this file, type **Alt+F2** and enter the following command:

```
gksu gedit /etc/squid/squid.conf
```

2.1. Naming the proxy

It's important that Squid knows the name of the machine. To do this, locate the line **visible_hostname**.

For example, if the machine is called **ubuntu** insert:

```
visible_hostname ubuntu
```

2.2 Choosing the Port

By default, the proxy server will use port 3128. To choose another port, locate the line:

```
http_port 3128
```

and change the port number, for example:

```
http_port 3177
```

2.3.Choosing the interface

By default the proxy server will listen on all interfaces. For security reasons, its better to put it on your local network only. For example, if the network card connected to your LAN has IP 10.0.0.1, change the line:

```
http_port 10.0.0.1:3177
```

2.4. Setting access rights and priorities

By default, nobody else is allowed to connect to the proxy server. A list of permissions must be created.

For example, we will define a group encompassing the local network.

Find the line beginning with **acl localhost...**

At the end of the section, add:

```
acl lanhome src 10.0.0.0/255.255.255.0
```

(lanhome is a random name chosen).

```

#
#Recommended minimum configuration:
acl all src 0.0.0.0/0.0.0.0
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl to_localhost dst 127.0.0.0/8
acl SSL_ports port 443          # https
acl SSL_ports port 563          # snews
acl SSL_ports port 873          # rsync
acl Safe_ports port 80          # http
acl Safe_ports port 21          # ftp
acl Safe_ports port 443          # https
acl Safe_ports port 70          # gopher
acl Safe_ports port 210         # wais
acl Safe_ports port 1025-65535  # unregistered ports
acl Safe_ports port 280         # http-mgmt
acl Safe_ports port 488         # gss-http
acl Safe_ports port 591         # filemaker
acl Safe_ports port 777         # multiling http
acl Safe_ports port 631         # cups
acl Safe_ports port 873         # rsync
acl Safe_ports port 901         # SWAT
acl purge method PURGE
acl CONNECT method CONNECT
acl lanhome src 10.0.0.0/255.255.255.0

```

2.5. Authorizing access to group

Now that the group is defined, we will authorise it to use the proxy.

Locate the line **http_access allow ...** and add below (before the line http_access deny all):

```
http_access allow lanhome
```

```

# Example rule allowing access from your local networks. Adapt
# to list your (internal) IP networks from where browsing should
# be allowed
#acl our_networks src 192.168.1.0/24 192.168.2.0/24
#http_access allow our_networks
http_access allow localhost
http_access allow lanhome

# And finally deny all other access to this proxy
http_access deny all

```

2.6. Allow the use non-standard ports

By default, Squid allows HTTP traffic only on specific ports (e.g. 80). This can cause problems on websites using other ports.

- For example, <http://toto.com:81/images/titi.png> will be blocked by Squid

To avoid this deadlock, find the line **http_access deny! Safe_ports** and the edit it to: **# http_access deny! Safe_ports**

3.Starting the Proxy

Restart the proxy to apply the modifications you made. Type:

sudo /etc/init.d/squid restart

Miscellaneous

Server logs

The proxy logs are located in: **/var/log/squid/access.log**

Experiment No.17

Aim : To Demonstrate a simple Unix shell program

In UNIX, commands are submitted to the Operating System via a *shell*. A shell is an environment which allows commands to be issued, and also includes facilities to control input and output, and programming facilities to allow complex sets of actions to be performed. Whenever you type commands at the prompt in Unix, you are actually communicating interactively with a shell.

In addition to typing commands directly to the shell, you can place them in a file (which must be given execute permission), and then run the file. A file containing UNIX shell commands is known as a *script* (or shell script). Executing the script is equivalent to typing the commands within it.

The Unix shells are actually quite sophisticated programming languages in their own right: there are entire books devoted to programming in them. Together with the large number of special utility programs which are provided as standard, scripts make Unix an extremely powerful operating system.

Scripts are *interpreted* rather than *compiled*, which means that the computer must translate them each time it runs them: they are thus less efficient than programs written in (for example) C. However, they are extremely good where you want to use Operating System facilities; for example, when processing files in some fashion.

There are actually no less than three different types of scripts supported in Unix: Bourne shell, C shell, and Korn shell. Bourne is the most common, Korn the most powerful, and C the most C-like (handy for C programmers). This tutorial will concentrate on the simplest of the three: the Bourne shell.

A simple Bourne-shell script

If you simply type Unix commands into a file, and make it executable, then run it, Unix will assume that the commands in it are written in whatever shell language you happen to be using at the time (in your case, this is probably the C shell). To make sure that the correct shell is run, the first line of the script should always indicate which one is required. Below is illustrated a simple Bourne-shell script: it just outputs the message "hello world":

```
#!/bin/sh  
echo "hello world"
```

Use the text editor to create a file with the lines above in it. Save the file, calling it *hello*. Then make the *hello* file executable by typing:

```
chmod 755 hello
```

You can then run the script by simply typing *hello*. When you type *hello*, UNIX realises that this is a script, and runs the UNIX commands inside it (in this case, just the *echo* command).

Naming Shell Script Files

You can give your shell scripts almost any name you like. Be careful, however! If the name you use happens to already be an existing UNIX command, when you try to run your script you will end up running the system-defined command instead. Some fairly obvious names can cause you this problem; for example, *test*. To make sure that your intended name is O.K., you should check whether it exists before you start editing the new command. You can do this by using the *which* command, which will locate a command, if it exists, and tell you that it can't locate it if it doesn't. For example, to find out whether there are commands called *dc* and *dac*, type:

```
which dc
which dac
```

Most UNIX commands are stored in the */bin* directory, so you can get a list of most of them by typing:

```
ls /bin
```

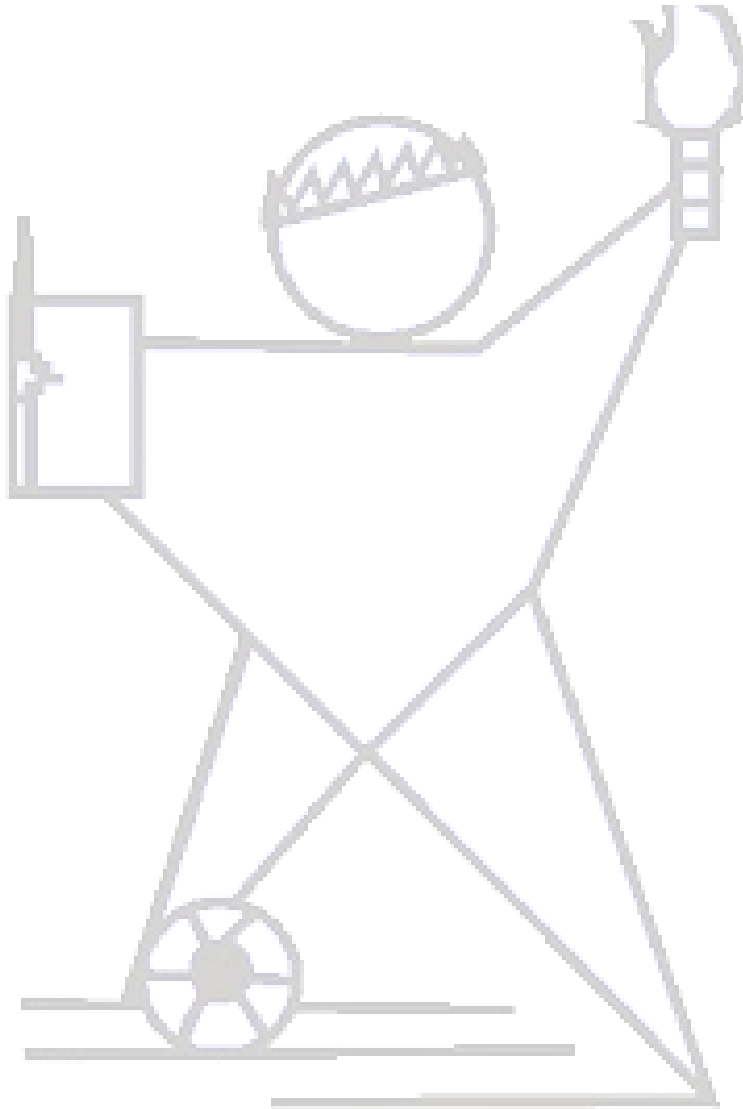
Changing the size of the cache

- The Squid cache is enabled by default, which helps accelerate the loading of some pages.
- The default allocated size is 100 MB (found in ***/var/spool/squid***)
- To change its size, edit the ***/etc/squid/squid.conf*** file.
- Find the line: ***# cache_dir ufs /var/spool/squid 100 16 256***
- Edit it. You can change the value 100 to whatever you want (e.g. 200 for 200 MB): ***cache_dir ufs /var/spool/squid 200 16 256***

Functions and additional modules

Squid is full of options and modules:

- Prefetch (to preload the pages and speed up navigation).
- Antivirus filters, AntiPopUp, etc.
- Access control via proxy login and password.
- Time-based access control.



Experiment No.18

Aim : Shell Script Program to Print Student Mark Sheet

```
echo "Enter the five subject marks for the student"
```

```
read m1 m2 m3 m4 m5
```

```
sum1=`expr $m1 + $m2 + $m3 + $m4 + $m5`
```

```
echo "Sum of 5 subjects are: " $sum1
```

```
per=`expr $sum1 / 5`
```

```
echo " Percentage: " $per
```

```
if [ $per -ge 60 ]
```

```
then
```

```
echo "You get Distinction"
```

```
elif [ $per -ge 50 ]
```

```
then
```

```
echo "You get First class"
```

```
elif [ $per -ge 40 ]
```

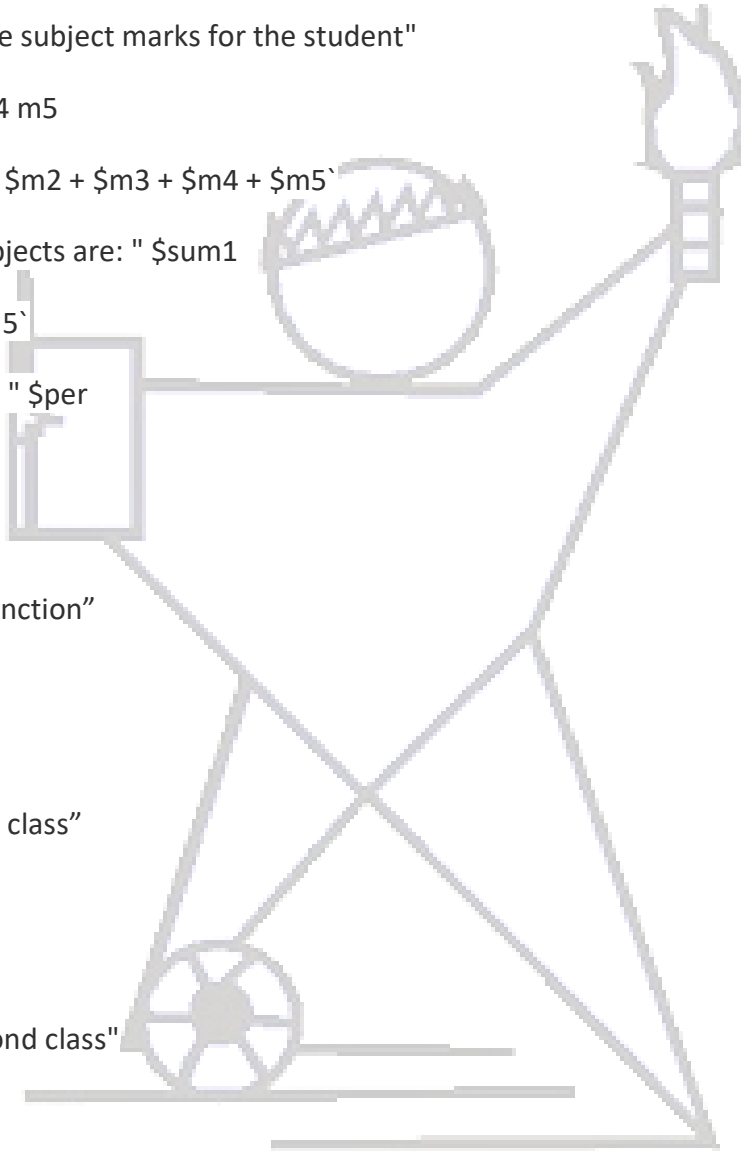
```
then
```

```
echo "You get Second class"
```

```
else
```

```
    echo "You get Fail"
```

```
fi
```



OUTPUT

```
[04mca58@LINTEL 04mca58]$ sh filemenu.sh
```

Enter the five subject marks for the student

45

35

30

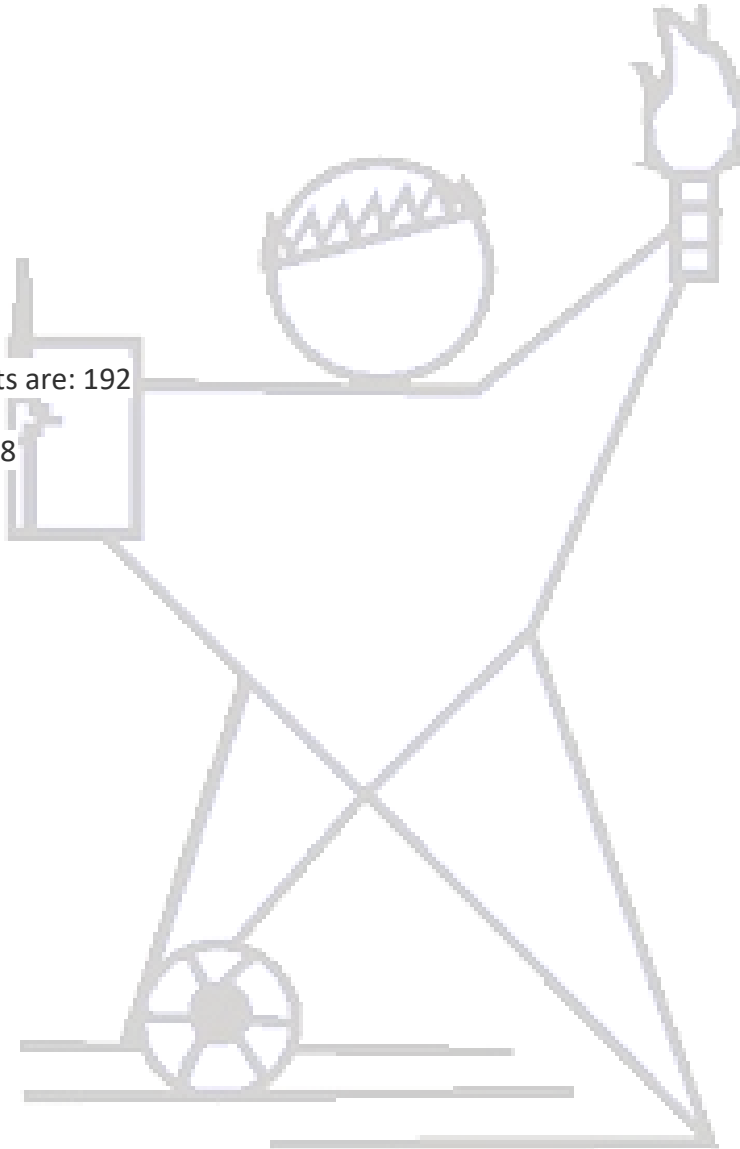
40

42

Sum of 5 subjects are: 192

Percentage : 76.8

You get Distinction



Experiment No. 19

Aim :Shell Script Program to perform all Arithmetic operations on integers

```
a=10
```

```
b=20
```

```
val=`expr $a + $b`
```

```
echo "a + b : $val"
```

```
val=`expr $a - $b`
```

```
echo "a - b : $val"
```

```
val=`expr $a \* $b`
```

```
echo "a * b : $val"
```

```
val=`expr $b / $a`
```

```
echo "b / a : $val"
```

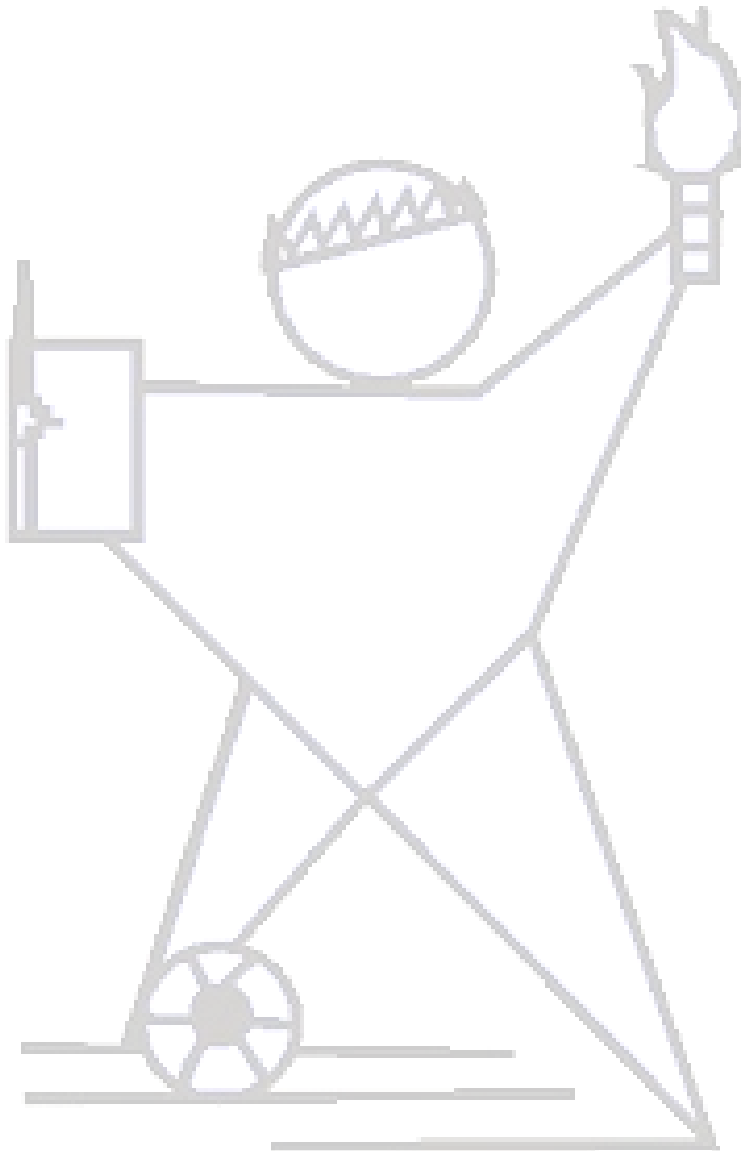
```
val=`expr $b % $a`
```

```
echo "b % a : $val"
```

```
if [ $a == $b ]
```

```
then
```

```
    echo "a is equal to b"
```



fi

if [\$a != \$b]

then

echo "a is not equal to b"

fi

This would produce following result –

a + b : 30

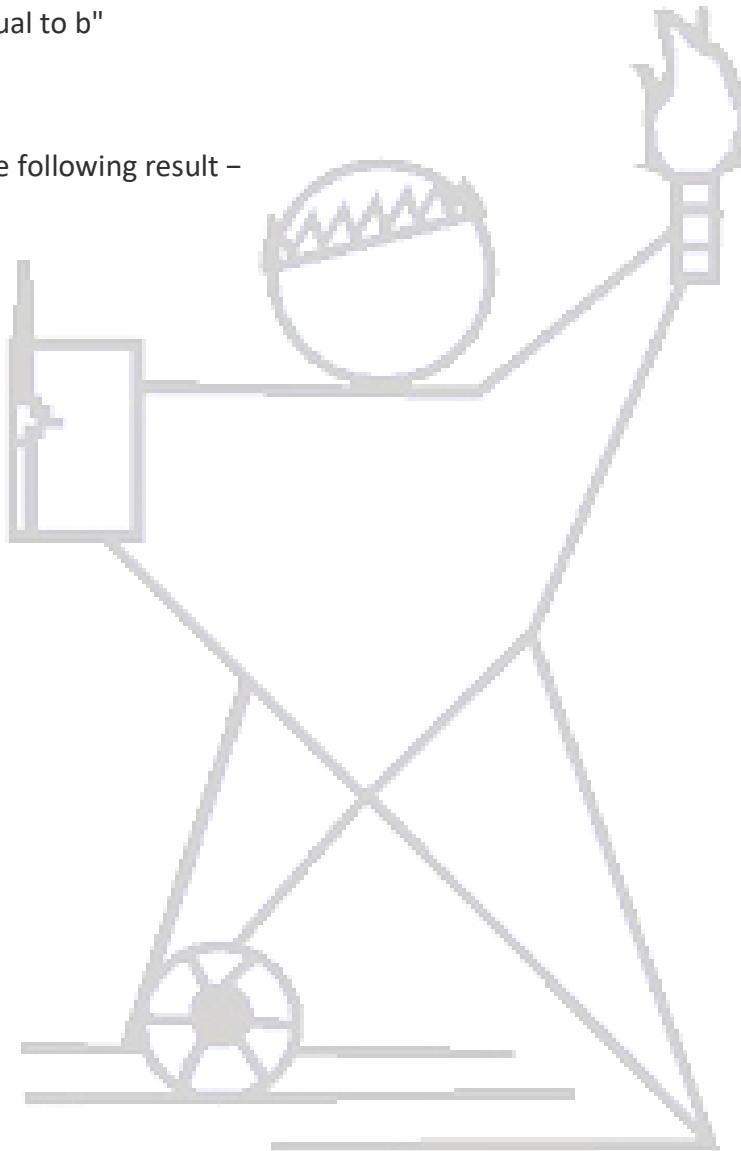
a - b : -10

a * b : 200

b / a : 2

b % a : 0

a is not equal to b



Experiment No.20

Aim : Shell Script Program to perform all arithmetic operations on floating point

1. Factorial:

```
echo "Enter an integer: "
```

```
read n
```

```
# Below we define the factorial function in bc syntax
```

```
fact="define f (x) {
```

```
i=x
```

```
fact=1
```

```
while (i > 1) {
```

```
fact=fact*i
```

```
i=i-1
```

```
}
```

```
return fact
```

```
}"
```

```
# Below we pass the function defined above, and call it with n as a parameter and pipe it to bc
```

```
factorial=`echo "$fact;f($n)" | bc -l`
```

```
echo "$n! = $factorial"
```


Output:

Enter an integer:

n=6! = 720

2.Log:

```
1. #!/bin/bash
2. echo "Enter value: "
3. read x
4.
5. echo "Natural Log: ln($x) : "
6. echo "l($x)" | bc -l
7.
8. echo "Ten Base Log: log($x) : "
9. echo "l($x)/l(10)" | bc -l
```

Output:

Enter value: n=2

Natural Log: ln(x=2) :.69314718055994530941

Ten Base Log: log(x=2) :.30102999566398119521

3.Other Operations:

Floating point number functions.

float_scale=2

.

```
function float_eval()
{
    local stat=0
    local result=0.0
    if [[ $# -gt 0 ]]; then
        result=$(echo "scale=$float_scale; $*" | bc -q 2>/dev/null)
        stat=$?
        if [[ $stat -eq 0 && -z "$result" ]]; then stat=1; fi
    fi
    echo $result
}
```

```

    return $stat
}

#####
# Evaluate a floating point number conditional expression.

function float_cond()
{
    local cond=0
    if [[ $# -gt 0 ]]; then
        cond=$(echo "$*" | bc -q 2>/dev/null)
        if [[ -z "$cond" ]]; then cond=0; fi
        if [[ "$cond" != 0 && "$cond" != 1 ]]; then cond=0; fi
    fi
    local stat=$((cond == 0))
    return $stat
}

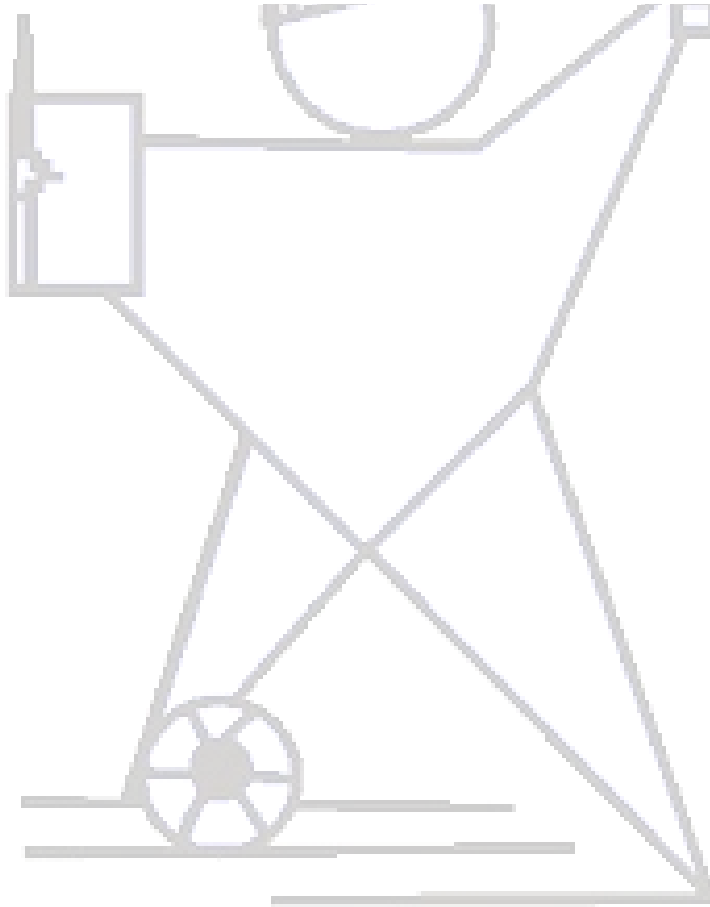
# Test code if invoked directly.
if [[ $(basename $0 .sh) == 'float' ]]; then
    # Use command line arguments if there are any.
    if [[ $# -gt 0 ]]; then
        echo $(float_eval $*)
    else
        # Turn off pathname expansion so * doesn't get expanded
        set -f
        e="12.5 / 3.2"
        echo $e is $(float_eval "$e")
        e="100.4 / 4.2 + 3.2 * 6.5"
        echo $e is $(float_eval "$e")
        if float_cond '10.0 > 9.3'; then
            echo "10.0 is greater than 9.3"
        fi
        if float_cond '10.0 < 9.3'; then
            echo "Oops"
        else
            echo "10.0 is not less than 9.3"
        fi
        a=12.0
        b=3.0
        c=$(float_eval "$a / $b")
        echo "$a / $b" is $c
    fi
fi

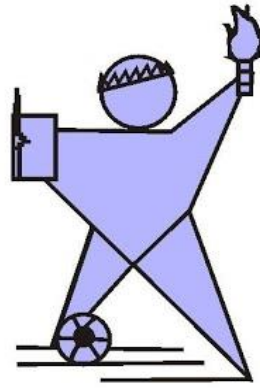
```

```
set +f
fi
fi
result=$(echo "scale=$float_scale; $*" | bc -q 2>/dev/null)
```

Output:

```
$ sh float.sh
12.5 / 3.2 is 3.90
100.4 / 4.2 + 3.2 * 6.5 is 44.70
10.0 is greater than 9.3
10.0 is not less than 9.3
12.0 / 3.0 is 4.00
```





Knowledge, Skills, Values

IPS ACADEMY

16 Collages, 71 Courses, 51 Acre Campus
ISO 9001: 2008 Certified

Knowledge Village
Rajendra Nagar
A.B.Road Indore
452012(M.P.) India

Ph: 0731-4014601-604 Mo: 07746000161

E Mail: office.ies@ipsacademy.org

Website: ies.ipsacademy.org & www.ipsacademy.org