Department of Computer Application

LAB MANUAL

Session 2017-2018

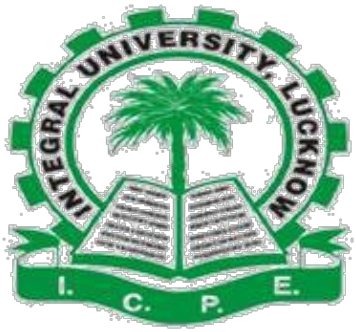
Subject Name : - Unix/Linux & Shell programming Lab

Subject Code : - CA310

Branch : - BCA

Year : - 3rd

Semester : - 5th



INTEGRAL UNIVERSITY LUCKNOW

Dasauli, Kursi Road, PO Basha-226026

Revision History

REVISION HISTORY					
REVISION #	**AUTHOR(S)**	**REVIEWER(S)**	**DATE OF RELEASE**	**OWNER**	**SUMMARY OF CHANGES**
Revision - 1	**Mr. Mohd.Mushtaq**	Internal Reviewers	08/01/2016	HOD - CA	Initial Release
Revision -2		Departmental QAC	30/01/2016	HOD - CA	Major revisions made. Incorporated review comments from departmental QAC into Revision-2
Revision -3		External Reviewers	26/02/2016	HOD - CA	
Revision -4	**Mr. Arshad Ali**	Internal Reviewers	19/082017	HOD - CA	Major revisions are update.

Week- 1	**Objective-1 1.1**	Miscellaneous commands.
	Objective-2	Cal,date,find,jobs,man,set,who,who am i, echo,wc,test,bc.
Week-2	**2.1**	File and directory related commands.
	2.2	Mkdir, cd, cd.., cd., pwd, rmdir,ls,ls-l,
	Objective-3	Cat, cp, mv,rm,comm, diff, cmp.
Week-3	**3.1**	Communication command.
	3.2	Ping, ftp,telnet,finger
	Objective-4	Du,df
Week-4	**4.1**	Storage Command.
	Objective-5	Compress,uncompress,cpio,dump, pack, tar,mt.
Week-5	**5.1**	System Status Command.
	Objective-	At, chmod, chgrp, chown, crontab, date, df, du, env, finger, ps, ruptime, shutdown, stty, who.
Week-6	**6 6.1**	Overview of shell script programming.
	6.2	Write a program for the addition, subtraction, multiplication and division of two numbers.
	6.3	Calculate the area of rectangle parameter of rectangle, area of circle and circumference of circle.
Week-7	**Objective-7.1**	Write shell script to enter the marks of five subjects and calculate the percentage of five subjects.
		Shell script to calculate the gross salary as under the following constraints.

(i) If basic pay less than 1500 then HRA=10% DA=90% of basic pay.

(ii) If basic pay equal or above than 1500 then HRA=500 DA=98% of basic pay.

Enter the employee salary through the keyboard.

7.2 Write a program to check the given year is leap year or not.

Week-8 **Objective-8.1** Write a program to find the greatest no. among three number(using multiple if)

Write a program to use command who, ls and cal through case statement.

Write a program to find the the factorial of a given number.

Week-9 **Objective-9.1** Write a program to check whether the given number is prime or not.

9.2 Write a program to print the Fibonacci series.

9.3 Shell script to find out the following.

i) Sum of digit of a given no.

ii) Reverse of a given no.

Week-10 **Objective-10** Write a program to find the power of any number.

Objective-11 Overview of system programming.

Use of fork() to create the process.

Week-9 **Objective-12** Use of fork to create the child process.

Create a parent child relationship.

Week#1

Experiment No.-1

Miscellaneous Commands:-

cal:-cal command is used to display the calendar.

Syntax:-

cal [options] [month] [year]

Tag	Description
-**1**	Display single month output. (This is the default.)
-**3**	Display prev/current/next month output.
-**s**	Display Sunday as the first day of the week. (This is the default.)
-**m**	Display Monday as the first day of the week.
-**j**	Display Julian dates (days one-based, numbered from January 1).
-**y**	Display a calendar for the current year.

Example:- $cal

date:-The date command displays the current day, date, time, and year.

Syntax:-

date [*OPTION*]... [*+FORMAT*]

Example:- $date -s

find:- find command lists all of the files within a directory and its subdirectories that match a set of conditions. This command is most commonly used to find all he files that have a certain name.

Syntax:-

find [pathnames] [conditions]

Example:- How to find for a file using

name? find -name "sum.java"

./bkp/sum.java

./sum.java

jobs:- This command reports any programs that you suspended and still have running or waiting in the background (if you had pressed Ctrl-z to suspend an editing session, for example).

Syntax:- $jobs [options] jobID

Example:-To display the status of jobs in the current shell,

enter: $jobs

Sample outputs:

[1] 7895 Running gpass &

[2] 7906 Running gnome-calculator &

[3]- 7910 Running gedit fetch-stock-prices.py &

[4]+ 7946 Stopped ping cyberciti.biz

man:-This command displays the manual page for a particular command. If you are unsure how to use a command or want to find out all its options, you might want to try using man to view the manual page.

Syntax:-

man -k [apropos options] regexp ..

Example:- $man cat

set:-

Syntax:- set *var*[*n*] =*word*

Example:- set PATH="/bin:/usr/bin:/usr/sbin:usr/local/bin"

this command sets the environment variable **PATH**, such that the shell will search for files in the**/bin**,**/usr/bin**,**/usr/sbin** and**/usr/local/bin** directories, in that order.

Who:-The **who** command prints information about all users who are currently logged in.

Syntax:- who [*OPTION*]

example:- $who

who am i :-who am i prints the effective user ID

Syntax:-

who am i [OPTION]

Example:- $who am i

echo:- echo displays a line of text.

Syntax:-$echo [*SHORT-OPTION*]... [*STRING*]...

Example:-

echo Hello, World!

wc:- the wc command is used to print the no. of lines, words and characters in file.

Syntax:-$wc file name

Example:-

$wc abc

test:-Checks file types and compares

values. **Syntax**:- $test expression

Example:-$if test 5 -

gt 4 then

echo

“hello” else

echo

“hi” fi

bc:- **bc** is an arbitrary-precision language for performing math calculations.

Syntax:-bc [-hlwsqv] [*long-options*] [*file* ...]

Example:- pi=$(echo "scale=10; 4*a(1)" | bc -l)

Week#2

File and Directory related command in Unix:-

Commands can be run by themselves, or you can pass in additional arguments to make them do different things. Typical command syntax can look something like this:

command [-argument] [-argument] [--argument] [file]

Experiment No-2

ls:- list directory contents

Syntax:- **ls**[*OPTION*]... [*FILE*]...

Examples:- ls -l

Lists the total files in the directory and sub directories, the names of the files in the current directory, their permissions, the number of sub directories in directories listed, the size of the file, and the date of last modification.

Cd:-The **cd** command, which stands for "change directory", changes the shell's current working directory.

Syntax:- cd [-L|-P] *directory*

Example:-$cd hope

The above example would change the working directory to the hope sub directory if it exists.

Cp:- Copies files and directories.

Syntax:-cp [*OPTION*]... [-T] source

dest **Example:-**cp file1.txt file2.txt

mv:-mv renames file source to dest, or moves the source file (or files) to

directory **Syntax:-**mv [*OPTION*]... [**-T**] source dest

Example:-$mv file1.txt file2.txt

rm:-The **rm** command removes (deletes) file or

directories. **Syntax:-** rm [*OPTION*]... file...

Example:- $rm myfile.txt

Remove the file **myfile.txt**. If the file is write-protected, you will be prompted to confirm that you really want to delete it.

Pwd:-Print the name of the working directory.

Syntax:-pwd [*OPTION*]...

Example:-$pwd

Print the name of the current working directory.

Mkdir:-Short for "make directory", mkdir is used to create directories on a file system.

Syntax:-mkdir [option ...] directory

… **Example:-** $mkdir mydir

Creates a new directory called mydir whose parent is the current directory

rmdir:-To delete directories.

Syntax:- rmdir directory name.

Example:-$rmdir directory1

cat:- cat stands for "catenate." It reads data from files, and outputs their contents. It is the simplest way to display the contents of a file at the command line.

Syntax:-cat [option]... [file]...

Example:- cat file.txt

Read the contents of file.txt and display them on the screen.

Less:- less is a simple, feature-rich command line file viewer.

Syntax:- less [option] file name

Example:-less file.txt

Views the file file.txt.

cd.:- It refer the current directory.

Syntax:- cd .

Example:- $cd .

Cd ..:- cd .. is refer to the parent directory of the current working directory.

Syntax:- cd ..

Example:- $cd ..

cmp:- Compare two files byte by byte **Syntax**:-

cmp [option]... file1 [file2 [skip1 [skip2]]]

Example:-$cmp file1.txt file2.txt

diff:-diff analyzes two files and prints the lines that are different. Essentially, it outputs a set of instructions for how to change one file in order to make it identical to the second file.

Syntax:- diff [*OPTION*]... files

Example:-diff -y file1.txt file2.txt

comm:- Compare two sorted files line-by-line.

Syntax:- comm [OPTION]... FILE1 FILE2

Example:-comm -12 myfile1.txt myfile2.txt

Print only the lines that are present in myfile1.txt and not myfile2.txt, and vice versa.

Chmod:-To set/modify a file's permissions you need to use the chmod program. Of course, only the owner of a file may use chmod to alter a file's permissions.

Syntax:- chmod [options] mode file(s)

Options:-

u he owner user

g the owner group

o others (neither u, nor

g) **a** all users

Example:-Following example removes read and write permission for the user

$ chmod u-rx filename

Week#3

Experiment No-3

Overview of Communication in Unix: While working on a Linux operating system you may need to communicate with other devices. For this, there are some basic utilities that you can make use of.

These utilities can help you communicate with:

- networks,
- other Linux systems
- and remote users

Some basic communication related commands as following.

ping

The ping command sends an echo request to a host available on the network. Using this command you can check if your remote host is responding well or not.

The ping command is useful for the following:

- Tracking and isolating hardware and software problems.
- Determining the status of the network and various foreign hosts.
- Testing, measuring, and managing networks.

Syntax:

Following is the simple syntax to use **ping** command:

```
$ping hostname or ip-address
```

Above command would start printing a response after every second. To come out of the command you can terminate it by pressing CNTRL + C keys.

Example:

Following is the example to check the availability of a host available on the network:

```
$ping google.com
PING google.com (74.125.67.100) 56(84) bytes of data.
64 bytes from 74.125.67.100: icmp_seq=1 ttl=54 time=39.4 ms
64 bytes from 74.125.67.100: icmp_seq=2 ttl=54 time=39.9 ms
64 bytes from 74.125.67.100: icmp_seq=3 ttl=54 time=39.3 ms
64 bytes from 74.125.67.100: icmp_seq=4 ttl=54 time=39.1 ms
64 bytes from 74.125.67.100: icmp_seq=5 ttl=54 time=38.8 ms
--- google.com ping statistics ---
22 packets transmitted, 22 received, 0% packet loss, time
21017ms rtt min/avg/max/mdev = 38.867/39.334/39.900/0.396 ms
$
```

If a host does not exist then it would behave something like this:

```
$ping giiiiiigle.com
ping: unknown host giiiiigle.com
```

ftp:-

Here ftp stands for **F**ile **T**ransfer **P**rotocol. This utility helps you to upload and download your file from one computer to another computer.

The ftp utility has its own set of UNIX like commands which allow you to perform tasks such as:

- Connect and login to a remote host.
- Navigate directories.
- List directory contents
- Put and get files
- Transfer files as ascii, ebcdic or binary

Syntax:-

Following is the simple syntax to use ping

command: $ftp hostname or ip-address

Above command would prompt you for login ID and password. Once you are authenticated, you would have access on the home directory of the login account and you would be able to perform various commands.

telnet:-

Many times you would be in need to connect to a remote Unix machine and work on that machine remotely. Telnet is a utility that allows a computer user at one site to make a connection, login and then conduct work on a computer at another site.

Once you are login using telnet, you can perform all the activities on your remotely connect machine. Here is example telnet session:

C:>telnet

amrood.com Trying...

Connected to amrood.com.

Escape character is '^]'.

finger:-The finger command displays information about users on a given host. The host can be either local or remote.

Finger may be disabled on other systems for security reasons.

Syntax:- finger

Check all the logged in users on local machine as follows:

$ finger

Login	Name	Tty	Idle	Login Time	Office
amrood		pts/0		Jun 25 08:03	(62.61.164.115)

Week#4

Experiment No-4

Storage Command:-

compress:-Compress command compresses a file and returns the original file with .z extension, to uncompress this filename.Z file use uncompress file name command.

Syntax:-compress options files

Options

- -bn limit the number of bits in coding to n.
- -c write to standard output (do not change files).
- -f compress conditionally, do not prompt before overwriting files.
- -v Print the resulting percentage of reduction for files.

Example:- $compress -f file1

Uncompress:-Uncompress file uncompresses a file and return it to its original form.

Syntax:-uncompress file name.*z*

this uncompresses the compressed file to its

original name. **Example:-** uncompress file1.z

Cpio:-cpio command is useful to backup the file systems. It copy file archives in from or out to tape or disk, or to another location on the local machine.

Syntax:-cpio flags [options]

It has three flags, -i, -o, -p

cpio -i [options] [patterns]

- cpio -i copy in files who names match selected patterns.
- If no pattern is used all files are copied in.
- It is used to write to a

  tape. cpio -o
- Copy out a list of files whose name are given on standard

  output. cpio -p
- copy files to another directory on the same

  system. Options
- -a reset access times of input files.
- -A append files to an archive (must use with -o).
- -b swap bytes and half-words. Words are 4 bytes.
- -B block input or output using 5120 bytes per record.
- -c Read or write header information as Ascii character.
- -d create directories as needed.
- -l link files instead of copying.
- -o file direct output to a file.
- -r rename files interactively.
- -R ID reassign file ownership and group information to the user's login ID.
- -V print a dot for each file read or written.
- -s swap bytes.
- -S swap half bytes.
- -v print a list of filenames.

Example:-pio -icdv "save"" < /dev/rst8

will restore all files whose name contain "save"

Dump:-It is useful to backup the file systems.
dump command copies all the files in filesystem that have been changed after a certain date. It is good for incremental backups. This information about date is derived from /var/adm/dumpdates and /etc/fstab .

Syntax:-dump [option [argument ...] filesystem]

Options:-

- 0-9 This number is dump level. 0 option causes entire filesystem to be dumped.
- B blocking factor taken into argument.
- D density of tape default value is 1600.
- f place the dump on next argument file instead of tape.

Example:- /usr/sbin/dump 0df 6250 /dev/rmt/c0t0d0BEST /mnt

Pack:-pack command compacts each file and combine them together into a filename.z file. The original file is replaced. Pcat and unpack will restore packed files to their original form.

Syntax:-Pack options files

Options:-

- - Print number of times each byte is used, relative frequency and byte code.
- -f Force the pack even when disk space isn't saved.
- To display Packed files in a file use pcat
  command pcat filename.z
- To unpack a packed file use unpack command as unpack

filename.z . **Example**:-$pack -f file1

Tar:-tar command creates an archive of files into a single file.Tar copies and restore files to a tape or any storage media.

Syntax:-tar [options] [file]

Options:

- b n use blocking factor of n.
- l print error messages about links not found.
- L follow symbolic links.
- v print function letter (x for extraction or a for archive) and name of

files. **Example**:-tar cvf - 'find . -print' > backup.tar

The above example will creates an archive of current directory and store it in file backup.tar.

Mt:-Mt command is used for tape and other device functions like rewinding, ejecting, etc. It give commands to tape device rather than tape itself. Mt command is BSD command and is seldom found in system V unix versions.

Syntax:-mt [-t tapename] command [count]

Example:-mt -t /dev/rmt/0mnb rew will rewind the tape in this device.

Week#5

Experiment No.-5

System Status Command:-

at:-command.
at command along with crontab command is used to

schedule jobs. **Syntax:-**at options time [date] [+increment]

Example:- $at -m 01:35 < my-at-jobs.txt

Chmod:-To set/modify a file's permissions you need to use the chmod program. Of course, only the owner of a file may use chmod to alter a file's permissions.

Syntax:- chmod [options] mode file(s)

Options:-

u he owner user g-

the owner group

o others (neither u, nor

g) a all users

Example:-Following example removes read and write permission for the user

$ chmod u-rx filename

chgrp:-Changes group ownership of a file or

files. Syntax:-chgrp [option] group file...

Example:-$chgrp hope file.txt

chown:-The chown command changes the owner and owning group of files.

Syntax:-chown [option]... [owner][:[group]] file...

Example:-$chown chope file.txt

crontab:- The crontab is a list of commands that you want to run on a regular schedule, and also the name of the command used to manage that list.

Syntax:-crontab [-u user] file

Example:-crontab -e

date:-Date displays today's date, to use it type date at prompt.

Syntax:- date

du:- du estimates and displays the disk space used by files.

Syntax:-du [option]... [file]...

Example:-$du -s *.txt

Reports the size of each file in the current directory with the extension **.txt**. Below is an example of the output:

df:- Report the amount of available disk space on file systems.

Syntax:-df [option]... [file]...

Example:-$df

env:-It is a shell command for Linux, Unix operating systems. It can be used to print a list of the current environment variables, or to run another program in a custom environment without modifying the current one.

Syntax:-env [option]... [-] [name=value]... [command [arg]...]

Example:-$env

Executing env with no options displays the current environment variables and their values.

Ps:- ps command is probably the most useful command for systems administrators. It reports information on active processes.

Syntax:-ps options

options:

- -a Lists all processes in system except processes not attached to terminals.
- -e Lists all processes in system.
- -f Lists a full listing.
- -j print process group ID and session ID.

Ruptime:- Ruptime command tells the status of local networked machines. **Syntax:-**ruptime options

options.

- -a include user even if they've been idle for more than one hour.
- -l sort by load average.
- -r reverse the sort order.
- -t sort by up time.
- -i sort by number of users.

Shutdown:-Shutdown command can only be executed by root. To gracefully bring down a system, shutdown command is used.

Syntax:-$shutdown -i1

options:

- -gn use a grace-period of n seconds (default is 60).
- -ik tell the init command to place system in a state k
- s single-user state (default)
- 0 shutdown for power-off.

- 1 like s, but mount multi- user file systems.
- 5 stop system, go to firmware mode.
- 6 stop system then reboot.
- -y suppress the default prompt for confirmation.

Stty:-stty command sets terminal input output options for the current terminal. without options stty reports terminal settings.

Syntax:-stty options modes < device

Example:-$stty sane

Reset all terminal settings to "sane" values; this has the effect of "fixing" the terminal when another program alters the terminal settings to an unusable condition.

Week#6

Shell script:-

A shell script is a text file that contains a sequence of commands for a unix based operating system. It's called a shell script because it combines into a "script" in a single file a sequence of commands that would otherwise have to be presented to the system from a keyboard one at a time. The shell is the operating system's command interpreter and the set of commands you use to communicate with the system. A shell script is usually created for command sequences for which a user has a repeated need. You initiate the sequence of commands in the shell script by simply entering the name of the shell script on a command line.

Experiment No-6

Objective: A shell script to accept two no. and perform all arithmetic operations on it.

Echo “Enter the first

no.” read a

echo “enter the second

no.” read b

c=`expr

a+b` echo

$a d=`expr

a-b` echo

$d e=`expr

a*b` echo

$e f=`expr

a/b` echo $f

Experiment No-7

Objective: Calculate the are of rectangle, parameter of rectangle, area of circle and circumference of circle.

```
echo " enter the length and breadth of

rectangle " read a b

ar=`expr $a \* $b`

pr=`expe 2 \*( $a +$

b )`

echo "enter the radius of

circle" read r

ac=`expr 3.14 \* r \* r`

cc=`expr 2 \* 3.14 \* r \* r`

echo "Area of rectangle=$ar"

echo "perimeter of rectangle is=$pr"

echo "Area of circle is=$ac"

echo "Circumference of circle is=$cc"
```

Experiment No.-8

Objective:- write a program to enter the marks of five subjects and calculre the percentage of five subjects.

```
Echo “enter the marks of five subjects of

any student.” read a b c d e

t=`expr $a +$ b + $c +

$d + $e`

p=`expr t / 5`

echo “the percentage of the student is=$p”
```

Week#7

Experiment No-9

Objective: Shell script to calculate the grass salary as under the following constraints. If basic pay less than 1500 then HRA=10% DA=90% of basic pay. If basic pay equal or above than 1500 then HRA=500 DA=98% of basic pay. Enter the employee salary through the keyboard.

```
Echo “enter the basic salary of
employee” read bs
if [ $bs -gt 1500
] then
hra=`expr (bs\*10)/100`
da=`expr  (bs\*90)/100`
else
hra=1500
da=`expr
($bs\*98)/100` fi
ts=` expr $bs +
$da+$hra`
echo $ts
```

Experiment No-10

Objective: Write a program to check the given year is leap year or not

```
echo "Enter
Year:" read y
year=$y
y=$(( $y % 4
)) if [ $y -eq 0
] then
    echo "$year is Leap Year!"
else
    echo "$year is not a Leap Year!"
fi
```

Week#8

Experiment No-11

objective: write a program to find the greatest no. among three number(using multiple if)

```
echo “enter any three

no.” read a b c

l=$a

if [ $b -gt $l

] then

l=$b

fi

if [ $c -gt $l

] then

l=$c

fi

echo Largest of $a $b $c is $l
```

Experiment No.-12

Objective:-write a program using command who, ls and cal through case statement. Echo “1-for who 2 -for ls 3 -for cal”

```
echo “enter your
choice” read a
case &a in
    1) $who
    2) $ls
    3) $cal
   *) echo “ wrong choice”
esac
```

Experiment No.-13

Objective:- Write a program to find the Factorial of a given no. echo "Total no of factorial wants"

```
read fact

ans=1

counter=0

while [ $fact -ne

$counter ] do

    counter=`expr $counter + 1`

    ans=`expr $ans \* $counter`

done

echo "Total of factorial is $ans"
```

Week#9

Experiment No.-14

Objective:- Write a program to check whether the given number is prime or not.

```
echo -n "Enter a
number: " read num
i=2

while [ $i -lt $num
] do
if [ `expr $num % $i` -eq
0 ] then
echo "$num is not a prime
number" echo "Since it is
divisible by $i" exit
fi
i=`expr $i +
1` done

echo "$num is a prime number "
```

Experiment No.-15

Objective:- Write a program to print the Fibonacci series.

```
echo "Enter How many
numbers:" read num

num1=0
num2=1

echo "Fibonacci series:"

echo $num1
echo $num2

count=2

while [ $count -le
$num ] do
num3=`expr $num1 +
$num2` echo $num3
num1=$num2
num2=$num3
count=`expr $count +
1` done
```

Experiment No-16

Objective:- Shell script to find the followings.

1) Sum of digits.
2) Reverse of Digits.

Sum of Digits:-

```
echo “enter any
no.” read a
s=0
while[$a -gt
0] do
s=`expr $s+$a%10`
a=`expr $a/10`
done
echo $s
```

Reverse of Digits:-

```
echo “enter any
no.” read a
r=0
while[$a -gt
0] do
r=`expr $r\*10 +
$a%10` a=`expr $a/10`
done
echo $r
```

Week#10

Experiment No.-17

Objective:- Write a program to find the power of any number.

```
echo "Input
number" read no
echo "Input power"
read power

counter=0
ans=1

while [ $power -ne
$counter ] do
    ans=`expr $ans \* $no`
    counter=`expr $counter + 1`
done

echo "$no power of $power is $ans"
```

System Programming :-

fork():- fork() creates a child process that differs from the parent process only in its PID and PPID, and in the fact that resource utilizations are set to 0. File locks and pending signals are not inherited.

Under Linux,fork() is implemented using copy-on-write pages, so the only penalty that it incurs is the time and memory required to duplicate the parent's page tables, and to create a unique task structure for the child.

Experiment No-18

Process Creation:-

A example of fork() follows. Here, the parent process prints Hello to stdout, and the new child process prints World. Note that the order of printing is not guaranteed. Without some method of synchronizing the processes execution, Hello may or may not be printed before World.

```
#include

#include

char string1[] = "\n Hello";

char string2[] = "

CS560.\n"; int main(void)

{

pid_t PID;

PID = fork();

if (PID == 0)
```

```
printf("%s",
string2); else
printf("%s",
string1); exit(0);
}
```

Week#11

Experiment No.-19

Objective:-A program to create the child process.

```
#include <stdio.h>
#define EVER ;;

int main()
{
  int process;
  char line[81];

  for (EVER)
    {
    fprintf(stderr, "cmd: ");
      if ( gets (line) == (char *) NULL) /* blank line input
        */ exit (0);
```

```
  /* create a new process */

    process = fork ();

    if (process > 0)        /* parent */
      wait ((int *) 0);    /* null pointer - return value not saved */
    else if (process == 0)    /* child */
      {                    /* execute program */
      execlp (line, line, (char *) NULL);
                           /* some problem if exec returns
      */ fprintf (stderr, "Can't execute %s\n", line);
      exit (1);
      }
    else if ( process == -1)   /* can't create a new process */
      {
      fprintf (stderr, "Can't
      fork!\n"); exit (2);
      }
    }
}
```

Experiment No.-20

Objective:-A program to create parent child relationship.

```
#include <stdio.h>
 /* for fork() */
#include <sys/types.h>
#include <unistd.h> /*
for wait*() */ #include
<sys/wait.h>

int main() {
   pid_t mypid,
   childpid; int status;

    /* what's our pid?
   */ mypid = getpid();
   printf("Hi. I'm the parent process. My pid is %d.\n", mypid);

    /* create the child
   */ childpid = fork();
   if ( childpid == -1 ) {
      perror("Cannot proceed. fork() error");
```

```
    return 1;
  }


  if (childpid == 0) {
     /* then we're the child process "Child 1" */
    printf("Child 1: I inherited my parent's pid as %d.\n", mypid);

     /* get our pid: notice that this doesn't touch the value of parent's
    "mypid" value */ mypid = getpid();
    printf("Child 1: getppid() tells my parent is %d. My own pid instead
is %d.\n", getppid(), mypid);


     /* forks another child
    */ childpid = fork();
    if ( childpid == -1 ) {
       perror("Cannot proceed. fork()
       error"); return 1;
    }

    if (childpid == 0) {
        /* this is the child of the first child, thus "Child 2" */
       printf("Child 2: I hinerited my parent's PID as %d.\n", mypid);
```

```
        mypid = getpid();
        printf("Child 2: getppid() tells my parent is %d. My own pid instead
is %d.\n", getppid(), mypid);

        childpid = fork();
        if ( childpid == -1 ) {
           perror("Cannot proceed. fork()
           error"); return 1;
        }

        if (childpid == 0) {
            /* "Child 3" sleeps 30 seconds then terminates 12, hopefully
before its parent "Child 2" */
           printf("Child 3: I hinerited my parent's PID as %d.\n", mypid);

           mypid = getpid();
           printf("Child 3: getppid() tells my parent is %d. My own pid instead
is %d.\n", getppid(), mypid);

           sleep(30);
           return 12;
        } else /* the parent "Child 2" suddendly returns 15 */
      return 15; } else {
        /* this is still "Child 1", which waits for its child to exit */
```

```
        while ( waitpid(childpid, &status, WNOHANG) == 0 ) sleep(1);


        if ( WIFEXITED(status) ) printf("Child1: Child 2 exited with exit
status %d.\n", WEXITSTATUS(status));
        else printf("Child 1: child has not terminated correctly.\n");
      }
   } else {
      /* then we're the parent process, "Parent" */
      printf("Parent: fork() went ok. My child's PID is %d\n", childpid);


      /* wait for the child to terminate and report about
      that */ wait(&status);


      if ( WIFEXITED(status) )
printf("Parent: child has exited with status %d.\n", WEXITSTATUS(status));
      else printf("Parent: child has not terminated normally.\n");
   }


   return 0;
}
```