Experiment-2

- **Aim**: To explore basic commands in Linux.
- **Objective**:- To practice various basic linux commands
- **Basic Commands**:

ls - if we want to see list of files on linux/Unix, use 'ls' command.

pwd - To display path of present working directory.

cd - To go to home ~~dolir~~ directory.

cd.. - To go to parent directory.

mkdir - To create new directory.

rmdir - To create ~~a files~~ the directory.

rm - To delete a file in directory.

touch - To create a file in directory.

man & help - man → To show manual pages of a command.

help → it shows which command can be used.

cp - copy files through command line it takes 2 arguments first the file is copied, second where to copy.

uname - to show info about given system which is running.

diff filename, filename - compare files and show where they are different.

wc filename - tells you how many lines, words and characters are there in a file.

more ~~filename~~ - show first part of file, Just press spacebar to see more or Z to quit.

File Compression

gzip filename - compress files to smaller size possible end with '.gz'

gunzip filename - uncompress files compressed by gzip.

gzcat filename - Directly print using gzcat filename.

apt-get - Work with packages in linux command line.

chmod - make file executable and to change permissions granted it to linux.

host name - To know your host (or) network. Display hostname and IP.

ping - To check connection to a server.

mv - move files from one place to another it is also used to rename files.

locate - used when we don't know where file is saved.

echo - move some data usually text to file

nano - one of the text editor that denotes keywords with color and can recognize most languages.

vi - Can create a new file or modify a file using this editor.

jed - It is also a text editor.

sudo - stands for "SuperUser Do". If you want any command to be done with administrative or root priviledges, you can use sudo

df - To see available disk space in each of the partitions in your system.

- du - To know disk usage of file in your system.
- tar - Used to work with tarballs. Compress and uncompress types of tar like tar, tar.gz, tar.bz2.
- Cal - Display calender.

Syntax - cal [[month] year]

- Date :- display date & time.

  Syntax - date + [format]
- Banner - Prints a large banner on the standard output.

  Syntax - banner message.
- w : → who logged in and what they are doing.
- who - who logged in and where they are coming from.
- finger username - gives information of the user.
- last-1 username - tells you ~~the~~ when the last user last logged on and off and from where.
- talk username - Lets you have a (typed) conversation with another user.
- write username - lets you exchange one-line messages with another user.
- whoami - Displays the user id of currently logged in user.
- history - shows history of all basic commands that we used in bash.
- clear. - clears the clutters on the terminal.

Experiment - 3

- Aim: To write shell programs for factorial, multiplication, summation, adding array and comparing 2 strings.

1. Factorial of numbers:

```
echo "Enter a number"
read num
fact = 1
while [$num -gt 1]
do
    fact = $(fact * num)
    num = $(num -1)
done
    echo $fact
```

Input = 5
Output = 120

2. Multiply 2 numbers

```
echo "Enter num 1"
read num 1
echo "Enter num 2"
read num 2
multiply = "Expt $num 1 - $num2"
echo "multiplication is : $multiply"
```

Input = 2 3
Output = 6

3. Summation till n

```
echo "Enter size (N)"
read N
i = 1
Sum = 0
echo " "Enter number"
```

```
while [$i -le $N]
do
  read num
  sum = $(sum + num)
   i = $ (i+1)
done
   echo $ sum
```

Input = 5 1 2 4 3 5

Output = 15

4. Adding array elements

```
arr =(10, 20, 30, 40, 50)
sum= 0
for i in ${[arr[a]}
do
  sum = " exp $ sum + $ i "
done
echo & sum
```

Input - 10 20 30 40 50

Output - 150

5. Compare 2 Strings

```
read p " Enter 2 strings " str1 str2
if [$str1 == $ str2]
then
echo "Equal"
else
echo "not equal"
fi
```

Input - shell bash
shell shell

Output - Unequal
equal

6. Area and circumference of a circle

```
echo "Enter radius"
read r.
area = $ (echo "3.14 * $r * $r" / bc)
cir = $ (echo "3.14 * 2 * $r" / br)
echo " area of circle is" $ area
echo " circumference of circle is" $ cir
```

Output -

Enter the radius
3.5
Area of circle is 38.56
Circumference of circle is 21.98.

7. Check for palindrome

```
echo "Enter a string"
read input
reverse = " "
len = ${# input}
for (( i=$ len-1 ; i>=0 ; i--))
do reverse = "$ reverse ${input : $ i=1}
done
  if [$ input == reverse]
then
   echo "$ input is palindrome"
else
   echo " $ input is not palindrome"
fi
```

Output

Enter a string
Malayalam
Malayalam is palindrome.

8. Find an Armstrong number ;

```
echo "Enter the number"
read n
function arms
{
    t = $n
    s = 0
    b = 0
    c = 10
    while [$n -gt $b]
    do
    r = $ ((n%c))
    i = $ ((n*r*r))
    S = $ ((1+i))
    n = $ ((n/c))
    done
    echo $s
    echo "Armstrong number"
    else
    echo "Not an armstrong number"
    fi
}
results = "arms $n'
echo = " $ results "
```

Output

```
Enter a number
153
Armstrong number.
```

10. Count No. of vowels in a line of test

```
clear
echo "Enter a string to find the number
      of vowels"
read st
len = 'expr $st | wc -c'
len = 'expr $ len -1'
count = 0
while [ $ len - gt 0]
do
ch = 'expr $st / nt - c $ len'
case $ ch in
([aeiou, AEIOU]) count = 'expr $ count+1' ;;
esac
echo " Number of vowels in given line
      is $ count"
```

Output

```
I am a bad guy.
No. of vowels : 5
```

11. Sum of Digits

```
echo "Enter a number"
read num
sum =0
while [ $ num - gt 0]
do
    mod = $ ((num % 10))
    sum = $ ((sum + mod))
    num = $ ((num / 10))
done
echo $ sum.
```

~~Output~~ Output :

```
Enter a number
100
1
```

12. Prime Number

```
number = 53
i=2
flag = 0
while test $ i -u ' expr $ number 12'
do
if test 'expr $ number %. $ i '. -eq 0
then
flag = 1
fi
i = 'expr $ i+1 '
done if test '$ flag -eq'
then
echo " The number is prime"
else
echo " The number is not prime"
fi
```

Output

```
Numbers
53
The number is prime.
```

13. Given number is odd or even

```
echo " Enter number" a
read x
if ['expr $ x %. 2' = 0]
then
echo " Even no. "
else
echo " Odd no. "
fi
```

Output

```
83
Odd number.
```

14. Second Largest Numbers -

```
echo "Enter no. of elements ="
read n
a=0
b=0
for ((i=1 ; i<=n ; i++)) do
 echo "Enter num"
read no
if [$no -gt $a]
then
b=$aa=$no
elif [$no -gt $b]
then
b=$no
fi
echo "The second largest no. is : $b"
```

Output

```
Enter number : 23, 46, 91, 41
The second largest no. is 46,
```

15. Sum of two numbers using function,

```
function func_name()
{ }
function add()
{
  sum=$((  $1 + $2 ))
   echo "sum = $sum"
}
a=10
b=20
add $a $b
```

Output

```
Sum=30,
```

Experiment - 7

// File name : basic fork.c

1.
```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main(){
    fork();
    printf("Called fork() system call \n");
    return 0;
}
```

2.
```
#include <stdio.h>
#include <sys/types.h>
int main(){
    fork();
    fork();
    fork();
    printf("hello\n");
    return 0;
}
```

3.
```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
void forkexample() {
    // child process because return value zero
    if (fork() == 0)
        printf("Hello from Child!\n");
    //parent process because return value non-zero
    else
        printf("Hell from parent!\n");
}
int main() {
    forkexample();
    return 0;
}
```

4. Program to demonstrate creating processes using fork() -

```
#include <unistd.h>
#include <stdio.h>
int main() {
    // creating first child
    int n1 = fork();
    // creating second child. First child also
    //executes this line and creates grandchild.
    int n2 = fork();
    if (n1>0 && n2>0) {
        printf("parent\n");
        printf("%d %d\n", n1, n2);
        printf("my id is %d\n", getpid());
    }
```

```
else if (n1==0 && n2>0){
        printf("First child \n");
        printf("%d %d \n ",n1,n2);
        printf("my id is %d \n", getpid());
}
else if (n1>0 && n2==0){
        printf("Second child \n");
        printf("%d %d \n", n1,n2);
        printf("my id is %d \n", getpid());
}
else{
        printf("Third child \n");
        printf("% %d \n", n1,n2);
        printf("my id is %d \n", getpid());
}
return 0;
}
```

5.
```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main(){
    pid_t pid, mypid, myppid;
    pid = getpid();
    printf("Before fork: Process id is %d \n", pid
    pid = fork();
    if (pid < 0){
        perror("fork() failure\n");
        return 1;
    }
    // Child process
    if (pid == 0) {
        printf("This is child process \n");
        mypid = getpid();
        myppid = getppid();
        printf("Process id is %d and
                PPID is %d \n", mypid, myppid);
    }
    else { // Parent process
        sleep(2);
        printf("This is parent process \n");
        mypid = getpid();
        myppid = getppid();
        printf("Process id is %d and PPID
                is %d \n", mypid, myppid);
        printf("Newly created process id or
                child pid is %d \n", pid);
    }
    return 0;
}
```

6. Example with Fork, Wait, Exit System Call.

```
#include <stdio.h> // printf()
#include <stdlib.h> // exit()
#include <sys/types.h> // pid_t
#include <sys/wait.h> // wait()
#include <unistd.h> // fork

int main(){ int argc, char **argv) {
    pid_t pid;
    pid = fork();
    if (pid == 0) {
        printf("%t is the child process and
            pid is %d\n", getpid());
        int i=0;
        for (i=0; i<8; i++) {
            printf("%d\n", i);
            exit(0);
        }
        exit(0);
    }
    else if (pid>0) {
        printf("%t is the parent process and
            pid is %d\n", getpid());
        int status;
        wait(&status);
        printf("Child is reaped\n");
    }
    else {
        printf("Error in forking...\n");
        exit(EXIT_FAILURE);
    }
    return 0;
}
```

7. #include <stdio.h>

```
int main (void){
    char * argv [3] = {"Command-line ", ".", NULL};
    int pid = fork();
    if (pid == 0) {
        execvp ("find", argv);
    }
    wait(2);
    printf ("Finished executing the parent
        process \n" ); " - the child won't get
        here -- you will only see this once\n");
    return 0;
}
```