

MNNIT ALLAHABAD

OPERATING SYSTEM

SUBMITTED BY :

NAME : SHISHU

ROLL/REG ID : 2020CA089

SUBMIT DATE : 17/11/2021

SUBMITTED TO :

TEACHER _____

DEPPT : COMPUTER SCIENCE

DEADLINE: 17/11/2021

Assignment 1.

1. Study of Unix/Linux general purpose utility command list : man, who, cat, cd, cp, ps, ls, mv, rm, mkdir, rmdir, echo, more, date, time, kill, history, chmod, chown, finger, pwd, cal, logout, shutdown commands.

touch :- The touch is a command-line utility that is basically used to create a new empty files and update the timestamps of existing files and directories.

```
[root@localhost ~]# touch shishu
[root@localhost ~]# ls
bench.py  hello.c  shishu
[root@localhost ~]#
```

ls :- ls command is use for list all file

ls -a :- In linux hidden files start with .(dot) symbol and they are not visible in regular directory. The (ls -a) command will enlist the whole list of the current directory including the hidden files.

```
[root@localhost ~]# ls -a
.      .bash_logout  .bashrc      .cshrc      hello.c  .tcshrc
.     .bash_profile bench.py     .fldev_cfg  shishu
[root@localhost ~]#
```

ls -R :- Display directory Recursive.

```
[root@localhost /]# ls -R
var  tmp  srv  run  proc  mnt  lost+found  lib  etc  boot
usr  sys  sbin  root  opt  media  lib64  home  dev  bin
```

ls -al :- ls -al show in list format with details.

```
[root@localhost /]# ls -al
total 92
drwxrwxrwx  21 root root   550 Jan 26 07:03 .
drwxrwxrwx  21 root root   550 Jan 26 07:03 ..
lrwxrwxrwx   1 root root    7 Aug 12 23:54 bin -> usr/bin
drwxrwxr-x   2 root root   37 Dec  8 11:00 boot
drwx-----  2 root root   37 Dec  8 12:45 .cache
drwxr-xr-x   3 root root 12180 Jan  1 1970 dev
drwxr-xr-x 125 root root  6673 Dec 26 15:41 etc
-rw-rw-rw-   1 root root    0 Jan 26 07:03 .fscmd
drwxr-xr-x   2 root root   37 Aug 12 23:54 home
lrwxrwxrwx   1 root root    7 Aug 12 23:54 lib -> usr/lib
lrwxrwxrwx   1 root root    9 Aug 12 23:54 lib64 -> usr/lib64
drwx-----  2 root root   37 Dec  8 11:00 lost+found
```

cat :- cat (concatenate) command is very frequently used in linux. It reads data from the file and gives their content as output. It helps us to create, view, concatenate file. So let us see some frequent used cat command.

1.cat [option] [file]...

2. cat [file] | more

3. cat [file] | less

4. cat -n

```
[root@localhost /]# cat >> shishu.txt
kfhkshfkjdshdfkj
slfjlkjsjflkj
slfjlskjf^Z
[4]+  Stopped(SIGTSTP)      cat >> shishu.txt
[root@localhost /]# cat >> shishu1.txt
hello1
hello2
hello3^Z
[5]+  Stopped(SIGTSTP)      cat >> shishu1.txt
[root@localhost /]# cat shishu.txt shishu1.txt
kfhkshfkjdshdfkj
slfjlkjsjflkj
hello1
hello2
[root@localhost /]# cat -n shishu
[root@localhost /]# cat -n shishu.txt
  1 kfhkshfkjdshdfkj
  2 slfjlkjsjflkj
[root@localhost /]#
```

[illegible]

man :- It is used to display the user manual of any command that we can run on the terminal. It provides a detailed view of the command.

```
LS(1)                                User Commands                                LS(1)

NAME
    ls - list directory contents

SYNOPSIS
    ls [OPTION]... [FILE]...

DESCRIPTION
    List information about the FILES (the current directory by default).
    Sort entries alphabetically if none of -cftuvSUX nor --sort is speci-
    fied.

    Mandatory arguments to long options are mandatory for short options
    too.

    -a, --all
        do not ignore entries starting with .

    -A, --almost-all
        do not list implied . and ..

    --author
        with -l, print the author of each file

    -b, --escape
        print C-style escapes for nongraphic characters

    --block-size=SIZE

Manual page ls(1) line 1 (press h for help or q to quit)
```


help :- help command as told before just displays information about shell built-in commands

```
[root@localhost /]# help
GNU bash, version 5.0.17(1)-release (riscv64-redhat-linux-gnu)
These shell commands are defined internally. Type `help' to see this list.
Type `help name' to find out more about the function `name'.
Use `info bash' to find out more about the shell in general.
Use `man -k' or `info' to find out more about commands not in this list.

A star (*) next to a name means that the command is disabled.

job_spec [&]
(( expression ))
. filename [arguments]
:
[ arg... ]
[[ expression ]]
alias [-p] [name[=value] ... ]
bg [job_spec ...]
bind [-lpsvPSVX] [-m keymap] [-f file]
break [n]
builtin [shell-builtin [arg ...]]
caller [expr]
case WORD in [PATTERN [| PATTERN]...)>
cd [-L|[-P [-e]] [-@]] [dir]
command [-pVv] command [arg ...]
compgen [-abdefghjksuv] [-o option] [>
complete [-abdefghjksuv] [-pr] [-DEI]>
compopt [-o|+o option] [-DEI] [name .>
continue [n]
coproc [NAME] command [redirections]
history [-c] [-d offset] [n] or hist>
if COMMANDS; then COMMANDS; [ elif C>
jobs [-lnprs] [jobspec ...] or jobs >
kill [-s sigspec | -n signum | -sigs>
let arg [arg ...]
local [option] name[=value] ...
logout [n]
mapfile [-d delim] [-n count] [-O or>
popd [-n] [+N | -N]
printf [-v var] format [arguments]
pushd [-n] [+N | -N | dir]
pwd [-LP]
read [-ers] [-a array] [-d delim] [->
readarray [-d delim] [-n count] [-O >
readonly [-aAf] [name[=value] ...] o>
return [n]
select NAME [in WORDS ... ;] do COMM>
set [-abefhkmnptuvxBCHP] [-o option->
shift [n]
shopt [-pqsu] [-o] [optname ...]
```

cp :- It is used to copy file one destination to another destination.

```
[root@localhost /]# ls
bin  etc  lib64  mnt  root  shishu  shishu.txt  test  var
boot home lost+found opt  run  shishu1  srv        tmp
dev  lib  media  proc sbin  shishu1.txt sys        usr

[root@localhost /]# cp shishu.txt test
[root@localhost /]# cd test
[root@localhost test]# ls
shishu.txt
[root@localhost test]#
```

echo hello world : This command display a line of test/string on standard output or a file.

```
[root@localhost home]# echo hello world
hello world
```

passwd :- This command is used to changing current password for user root.

```
[root@localhost home]# passwd
Changing password for user root.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

uname :- uname is a command line utility that prints basic information about the operating system name and system hardware

```
[root@localhost /]# uname
Linux
[root@localhost /]#
```

grep :- It stands for global regular expression print. The grep filter searches a file for a particular pattern of characters, and displays all lines that contain that pattern. The pattern that is searched in the file is referred to as the regular expression.

```
localhost:~# cat readme.txt
Some tests:

- Compile hello.c with gcc (or tcc):

gcc hello.c -o hello
./hello

- Run QuickJS:

qjs hello.js

- Run python:

python3 bench.py
localhost:~# grep -i "run" readme.txt
- Run QuickJS:
- Run python:
```

find :-The find command in UNIX is a command line utility for walking a file hierarchy. It can be used to find files and directories and perform subsequent operations on them. It supports searching by file, folder, name, creation date, modification date, owner and permissions. By using the '-exec' other UNIX commands can be executed on files or folders found.

```
localhost:~# find ./leo -name new.txt
./leo/new.txt
```

last :-This command is used to display the list of all the users logged in and out since the file **/var/log/wtmp** was created. One or more usernames can be given as an argument to display their login (and out) time and their host-name.

```
localhost:~# last
sh: last: not found
```

top :-This command is used to show the Linux processes. It provides a dynamic real-time view of the running system. Usually, this command shows the summary information of the system and the list of processes or threads which are currently managed by the Linux Kernel.

```
Mem: 5772K used, 181472K free, 8K shrd, 0K buff, 864K cached
CPU:  0% usr  0% sys  0% nic 95% idle  0% io  0% irq  3% sirq
Load average: 0.00 0.00 0.00 1/31 92
```

PID	PPID	USER	STAT	VSZ	%VSZ	CPU	%CPU	COMMAND
75	62	root	R	1516	1%	0	1%	top
62	1	root	S	1552	1%	0	0%	sh -l
1	0	root	S	1512	1%	0	0%	{init} /bin/sh /sbin/init
57	1	root	S	1260	1%	0	0%	dhcpcd -q
55	1	root	S	744	0%	0	0%	settime -d /
7	2	root	SW	0	0%	0	0%	[ksoftirqd/0]
2	0	root	SW	0	0%	0	0%	[kthreadd]
6	2	root	SW<	0	0%	0	0%	[mm_percpu_wq]
8	2	root	SW	0	0%	0	0%	[kdevtmpfs]
10	2	root	SW<	0	0%	0	0%	[writeback]
11	2	root	SW	0	0%	0	0%	[kcompactd0]
12	2	root	SW<	0	0%	0	0%	[crypto]
13	2	root	SW<	0	0%	0	0%	[bioset]
14	2	root	SW<	0	0%	0	0%	[kblockd]
4	2	root	SW<	0	0%	0	0%	[kworker/0:0H]
16	2	root	SW	0	0%	0	0%	[kswapd0]
17	2	root	SW<	0	0%	0	0%	[bioset]
34	2	root	SW	0	0%	0	0%	[khvcd]
35	2	root	SW<	0	0%	0	0%	[bioset]
36	2	root	SW<	0	0%	0	0%	[bioset]
37	2	root	SW<	0	0%	0	0%	[bioset]
38	2	root	SW<	0	0%	0	0%	[bioset]
39	2	root	SW<	0	0%	0	0%	[bioset]
40	2	root	SW<	0	0%	0	0%	[bioset]

useradd :-This command in Linux that is used to add user accounts to your system. It is just a symbolic link to adduser command in Linux and the difference between both of them is that useradd is a native binary compiled with system whereas adduser is a Perl script which uses useradd binary in the background.

```
localhost:~# useradd leo
sh: useradd: not found
localhost:~# sudo useradd leo
sudo: useradd: command not found
```

passwd :-passwd command in Linux is used to change the user account passwords. The root user reserves the privilege to change the password for any user on the system, while a normal user can only change the account password for his or her own account.


```
localhost:/# passwd
Changing password for root
New password:
Bad password: too weak
Retype password:
passwd: password for root changed by root
```

userdel :- This command in Linux system is used to delete a user account and related files. This command basically modifies the system account files, deleting all the entries which refer to the username LOGIN. It is a low-level utility for removing the users.

```
localhost:~# userdel
sh: userdel: not found
localhost:~#
```

clear :- **clear** is a standard Unix computer operating system command that is used to clear the

```
localhost:~# ls
bench.py    hello.c    hello.js   readme.txt
localhost:~# pwd
/root
localhost:~# clear
```

terminal screen.

```
localhost:~#
```

cal 2000 :- By default, the cal command shows the calendar of the current month. With options, we can view the calendar of the an year or particular month of ay year. This particular command will display the calendar of the year 2000.

```
localhost:~# cal 2000
```

2000

January

Su	Mo	Tu	We	Th	Fr	Sa
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

February

Su	Mo	Tu	We	Th	Fr	Sa
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29				

March

Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

April

Su	Mo	Tu	We	Th	Fr	Sa
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

May

Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

June

Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

July

Su	Mo	Tu	We	Th	Fr	Sa
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

August

Su	Mo	Tu	We	Th	Fr	Sa
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

September

Su	Mo	Tu	We	Th	Fr	Sa
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

October

Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21

November

Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18

December

Su	Mo	Tu	We	Th	Fr	Sa
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16

history :-This command shows all the command used in the current login.

```
localhost:~# history
 0 ls
 1 top
 2 useradd leo
 3 sudo useradd leo
 4 sudo useraddtest_user
 5 userdel
 6 ls
 7 pwd
 8 clear
 9 cal
10 clear
11 cal 2000
12 clear
13 cal 9 1752
14 bc 1
15 bc -1
16 bc 5+4
17 bc 5+4|bc -1
18 echo 5+4|bc -1
19 yes please
20 clear
21 yes please
22 history
```

Chown: chow command gives us permission of ownership

Read: This permission allows the user to read files and in directories, it lets the user read directories and subdirectories stores in it.

Write: This permission allows a user to modify and delete a file. Also it allows a user to modify its contents (create, delete and rename files in it) for the directories. Unless the execute permission is not given to directories changes does do affect them.

Execute: The write permission on a file allows it to get executed. For example, if we have a file named *php.sh* so unless we don't give it execute permission it won't run.

chown [OPTION]... [OWNER][:[GROUP]] FILE...

chown [OPTION]... -reference=RFILE FILE...

cal: cal command show calendar of any year which we want

Finger command is a user information lookup command which gives details of all the users logged in. This tool is generally used by system administrators. It provides details like login name, user name, idle time, login time, and in some cases their email address even

```
shishu@shishu-VirtualBox:~/os/assignment1$ finger shishu@shishu-VirtualBox
finger: command not found
shishu@shishu-VirtualBox:~/os/assignment1$ finger shishu@shishu-VirtualBox
finger: command not found
shishu@shishu-VirtualBox:~/os/assignment1$ █
```

// sir what happening in finger command I don't Known

Log out : using this command operating system exit from terminal

2. Write C programs to simulate UNIX commands like ls, grep, etc.

```
#include<stdio.h>
```

```
#include<stdlib.h>
```

```
#include<dirent.h>
```

```
#define DATA_SIZE 1000
```

```
void createf()
```

```
{ char data[DATA_SIZE];
```

```
    char n[100];
```

```
    FILE * fPtr;
```

```
    int i;
```

```
    for ( i=0;i<2;i++){
```

```
printf("enter a file name:");  
  
gets(n);  
  
fPtr = fopen(n,"w");  
  
if(fPtr == NULL)  
{ printf("Unable to create file.\n");  
  exit(EXIT_FAILURE);  
}  
  
printf("Enter contents to store in file : \n");  
  
fgets(data, DATA_SIZE, stdin);  
  
fputs(data, fPtr);  
  
fclose(fPtr);  
  
printf("File created and saved successfully. ?? \n");  
  
}  
  
}  
  
void lsandgrep(){  
  char fn[10],pat[10],temp[200];  
  
  FILE *fp;  
  
  char dirname[10];  
  
  DIR*p;  
  
  struct dirent *d;
```



```
printf("Enter directory name\n");
```

```
scanf("%s",dirname);
```

```
p=opendir(dirname);
```

```
if(p==NULL)
```

```
{
```

```
    perror("Cannot find directory");
```

```
    exit(0);
```

```
}
```

```
while(d=readdir(p))
```

```
    printf("%s\n",d->d_name);
```

```
}
```

```
int main(){
```

```
    createf();
```

```
    lsandgrep();
```






```
}
```

```

enter a file name:file1.txt
Enter contents to store in file :
this is file1
File created and saved successfully.
enter a file name:file2.txt
Enter contents to store in file :
this is file2
File created and saved successfully.
Enter directory name
dir1
.
..
a.txt
b.txt
c.txt
dir1
dir2

```

Name	Date modified	Type	Size
dir1	11/13/2021 1:27 PM	File folder	
dir2	11/13/2021 1:28 PM	File folder	
a	11/13/2021 1:27 PM	Text Document	0 KB
b	11/13/2021 1:27 PM	Text Document	0 KB
c	11/13/2021 1:27 PM	Text Document	0 KB

				
file1	file2	lsgrep	lsgrep	lsgrep.o

3. Write a program to implement

1. Create a file
2. Read contents of a file
3. Write to a file
4. Link and unlink a file
5. Copy file
6. Read contents of a file in a reverse order

Using the system calls: `open()`, `close()`, `read()`, `write()`, `lseek()`, `link()`, `unlink()`.

```
#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<dirent.h>

#define DATA_SIZE 1000

#define MAX 100

void createf()

{ char data[DATA_SIZE];

    char n[100];

    FILE * fPtr;

    printf("enter a file name:");

    scanf("%s",n);

    fPtr = fopen(n,"w");

    if(fPtr == NULL)

    { printf("Unable to create file.\n");

        exit(EXIT_FAILURE);

    }

    fflush(stdin);

    printf("Enter contents to store in file : \n");

    fgets(data, DATA_SIZE, stdin);
```

```
fputs(data, fPtr);

fclose(fPtr);

printf("File created and saved successfully. \n");

}

void ReadFile(){

    char name[20];

    printf("Enter name of file:");

    scanf("%s",name);

    FILE *fp;

    fp=fopen(name,"r");

    if(fp==NULL)

    {

        printf("File does not exist");

        exit(1);

    }

    char ch = fgetc(fp);

    while (ch != EOF)

    {

        printf ("%c", ch);

        ch = fgetc(fp);

    }

}
```

```
}  
  
fclose(fp);  
  
}  
  
void CopyFile()  
{  
  
    FILE *fptr1, *fptr2;  
  
    char filename[100], c;  
  
  
    printf("Enter the filename to open for reading \n");  
    scanf("%s", filename);  
  
    fptr1 = fopen(filename, "r");  
  
    if (fptr1 == NULL)  
    {  
        printf("Cannot open file %s \n", filename);  
        exit(0);  
    }  
  
  
    printf("Enter the filename to open for writing \n");  
    scanf("%s", filename);  
  
    fptr2 = fopen(filename, "w");
```



```
if (fptr2 == NULL)
{
    printf("Cannot open file %s \n", filename);
    exit(0);
}

c = fgetc(fptr1);
while (c != EOF)
{
    fputc(c, fptr2);
    c = fgetc(fptr1);
}

printf("\nContents copied to %s", filename);

fclose(fptr1);
fclose(fptr2);

return 0;
}

void reverseContent()
{
    char x[100];

    printf("Enter file name:");
```

```
scanf("%s",x);  
  
FILE* fp = fopen(x, "a+");  
  
if (fp == NULL) {  
    printf("Unable to open file\n");  
    return;  
}
```

```
char buf[100];  
  
int a[MAX], s = 0, c = 0, l;  
  
fprintf(fp, " \n");  
  
rewind(fp);  
  
while (!feof(fp)) {  
    fgets(buf, sizeof(buf), fp);  
    l = strlen(buf);  
    strcpy(a,s+l);  
}
```

```
rewind(fp);  
  
c -= 1;  
  
while (c >= 0) {  
    fseek(fp, a, 0);  
    fgets(buf, sizeof(buf), fp);
```

```
    printf("%s", buf);

    c--;

}

return ;

}

int menu(){

    printf("\n1.CREATE FILE\n");

    printf("\n2.READ FILE\n");

    printf("\n3.COPY FILE\n");

    printf("\n4.REVERSE FILE\n");

    printf("\n5.LINK FILE\n");

    printf("\n6.UNLINK FILE\n");

    printf("7.EXIT\n");

    int ch;

    scanf("%d",&ch);

    return ch;

}

int main(){

    while(1){

        system("cls");
```

```
switch(menu()){  
case 1:  
    createf();break;  
case 2:  
    ReadFile();break;  
case 3:  
    CopyFile();break;  
case 4:  
    reverseContent();break;  
case 5:  
    //linkFile();  
    break;  
case 6:  
    //unlinkFile();break;  
case 7:  
    exit(0);  
    break;  
default:  
    printf("Enter valid choice!!");  
}
```

```
    getch();  
}  
return 0;  
}
```



```

1.CREATE FILE
2.READ FILE
3.COPY FILE
4.REVERSE FILE
5.EXIT
1
Enter a file name:file3.txt
Enter contents to store in file :
This is file 3 created by me
File created and saved successfully.

```

```

1.CREATE FILE
2.READ FILE
3.COPY FILE
4.REVERSE FILE
5.EXIT
2
Enter name of file:file1.txt
this is file1 lknfklfgnknfklnlngkgng
knjgogfnjkngoegvngof
jfgnfojgerogvfgonfneorgjf fjgflngoiegj egjf
jkngoerglf egjfekgjer gfjklgje rge;keg erg

```

```

1.CREATE FILE
2.READ FILE
3.COPY FILE
4.REVERSE FILE
5.EXIT
3
Enter the filename to open for reading
file1.txt
Enter the filename to open for writing
file2.txt
Contents copied to file2.txt

```

file1 - Notepad

```

File Edit Format View Help
this is file1 lknfklfgnknfklnlngkgng
knjgogfnjkngoegvngof
jfgnfojgerogvfgonfneorgjf fjgflngoiegj egjf
jkngoerglf egjfekgjer gfjklgje rge;keg erg

```

file2 - Notepad

```

File Edit Format View Help
this is file1 lknfklfgnknfklnlngkgng
knjgogfnjkngoegvngof
jfgnfojgerogvfgonfneorgjf fjgflngoiegj egjf
jkngoerglf egjfekgjer gfjklgje rge;keg erg

```

4. Determine the size of a file using the lseek command. Once you found out the size, calculate the number of blocks assigned for the file. Compare these results with the similar results obtained when using the function **stat**.

```
#include <unistd.h>
```

```
#include <stdio.h>
```

```
#include <fcntl.h>
```

```
#include <sys/stat.h>
```

```
void lseekfun(int ac, char *name[])
```

```
{
```

```
    if ( ac < 2 ) return 0;
```

```
    int fd = open(name,O_RDONLY);
```

```
    int size = lseek(fd, 0, SEEK_END);
```

```
    printf("Size using lseek:%d\n", size);
```

```
    close(fd);
```

```
    return 0;
```

```
}
```

```
void statfun(int ac, char *name[])
```

```
{  
    if ( ac < 2 ) return 0;  
  
    struct stat stbuf;  
  
    stat( name, &stbuf);  
  
    printf ("Size using stat:%lld\n", stbuf.st_size);  
}  
  
int main(){  
    printf("Enter file name:");  
    char name[20];  
    scanf("%s",name);  
    int n=strlen(name);  
    lseekfun(n,name);  
    statfun(n,name);  
    return 0;  
}
```

Enter file name:file1.txt

Size using lseek:157

Size using stat:157

Process returned 0 (0x0) execution time : 5.698 s

Press any key to continue.