**LAB\_02\_OS\_Assignment**

**CE-054**

**Aim: Implementation of “pwd” and “ls” commands. (Use of getcwd, opendir, closedir, readdir functions)**

1. **opendir:**

The opendir() function opens a directory stream corresponding to the directory name, and returns a pointer to the directory stream.

**Syntax:**

#include <sys/types.h>

#include <dirent.h>

DIR \*opendir (const char\* name );

The stream is positioned at the first entry in the directory.

On error, NULL is returned, and errno is set appropriately.

1. **closedir:**

The closedir() function closes the directory stream associated with dirp. The directory stream descriptor dirp is not available after this call.

**Syntax:**

#include <sys/types.h>

#include <dirent.h>

int closedir(DIR \*dirp);

The closedir() function returns 0 on success.

On error, -1 is returned, and errno is set appropriately.

1. **readdir:**

The readdir() function returns a pointer to a dirent structure representing the next directory entry in the directory stream pointed to by dirp.

**Syntax :**

#include <dirent.h> struct dirent \*readdir(DIR \*dirp);

It returns NULL on reaching the end of the directory stream.

On success, readdir() returns a pointer to a dirent structure.

On Linux, the dirent structure is defined as follows:

struct dirent {

|  |  |
| --- | --- |
| ino\_t d\_ino; | /\* inode number \*/ |
| off\_t d\_off; | /\* offset to the next dirent \*/ |
| unsigned short d\_reclen; | /\* length of this record \*/ |
| unsigned char d\_type; | /\* type of file\*/ |
| char d\_name[256]; | /\* filename \*/ |

};

If the end of the directory stream is reached, NULL is returned and errno is not changed. If an error occurs, NULL is returned and errno is set appropriately.

1. **getcwd and current dir name:**

This function returns the absolute pathname that is the current working directory of the calling process.

**Syntax** :

#include <unistd.h> char \*getcwd(char \*buf, size\_t size); char \*get\_current\_dir\_name(void);

Pathname is returned as the function result and via the argument buf, if present.

1. **getwd:**

The getwd() function shall determine an absolute pathname of the current working directory of the calling process, and copy a string containing that pathname into the array pointed to by the path\_name argument.

**Syntax :**

char \*getwd(char \*buf);

• Assignments:

1. Implementation of pwd cmd.

Code:

#include<unistd.h>

#include<stdio.h>

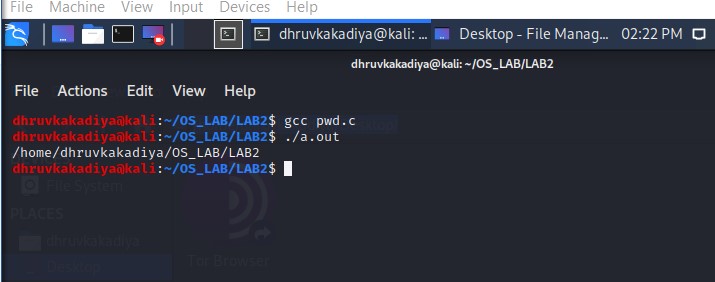
void main()

{

char buf[1024]; getcwd(buf, sizeof(buf)); printf("%s",buf); printf("\n");

}

**Output :**



2. Implementation of ls cmd. Code:

#include<unistd.h>

#include<dirent.h>

#include<stdio.h>

#include<sys/types.h>

#include<string.h> #include<stdlib.h>

void recursion(char path[], char name[], int mode); void recursion(char path[], char name[], int mode)

{

struct dirent \*dirp; DIR \*dir; char path\_in[1000]; strcpy(path\_in, path); strcat(path\_in, "/"); strcat(path\_in, name); if((dir = opendir(path\_in)) == 0) {

printf("Error"); exit(0);

}

while(dirp = readdir(dir))

{

if(strcmp(dirp->d\_name, ".") != 0 && strcmp(dirp->d\_name, "..") != 0)

{

for(int i=0 ; i < mode ; i++)

{

printf(" ");

}

printf("%s\n", dirp->d\_name);

if(dirp->d\_type == DT\_DIR)

{

mode += 1;

recurse(path\_in, dirp->d\_name, mode);

mode -= 1;

}

}

}

closedir(dir);

return;

}

int main() {

struct dirent \*dirp; DIR \*dir; char path[1000]; scanf("%s", path);

if((dir = opendir(path)) == 0)

{

printf("Error: open dir");

exit(0);

}

while(dirp = readdir(dir))

{

if(strcmp(dirp->d\_name, ".") != 0 && strcmp(dirp->d\_name, "..") != 0)

{

printf("%s\n", dirp->d\_name);

if(dirp->d\_type == DT\_DIR)

{

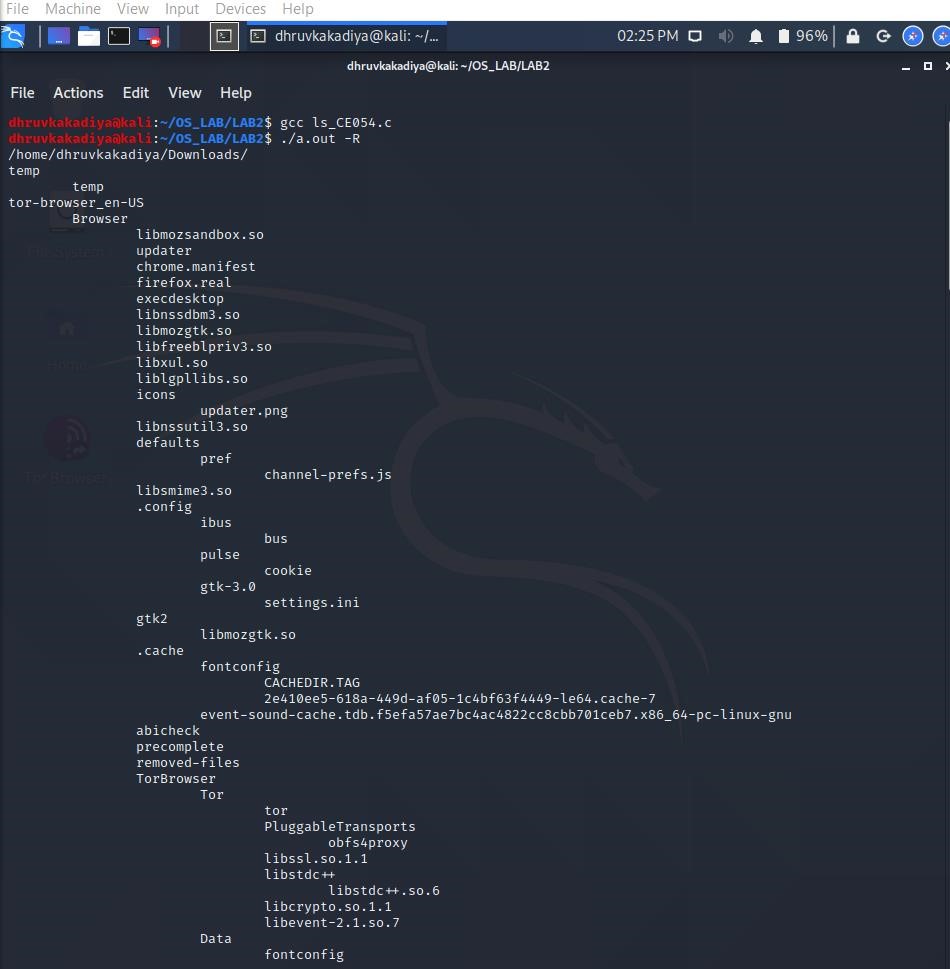
recursion(path, dirp->d\_name, 1);

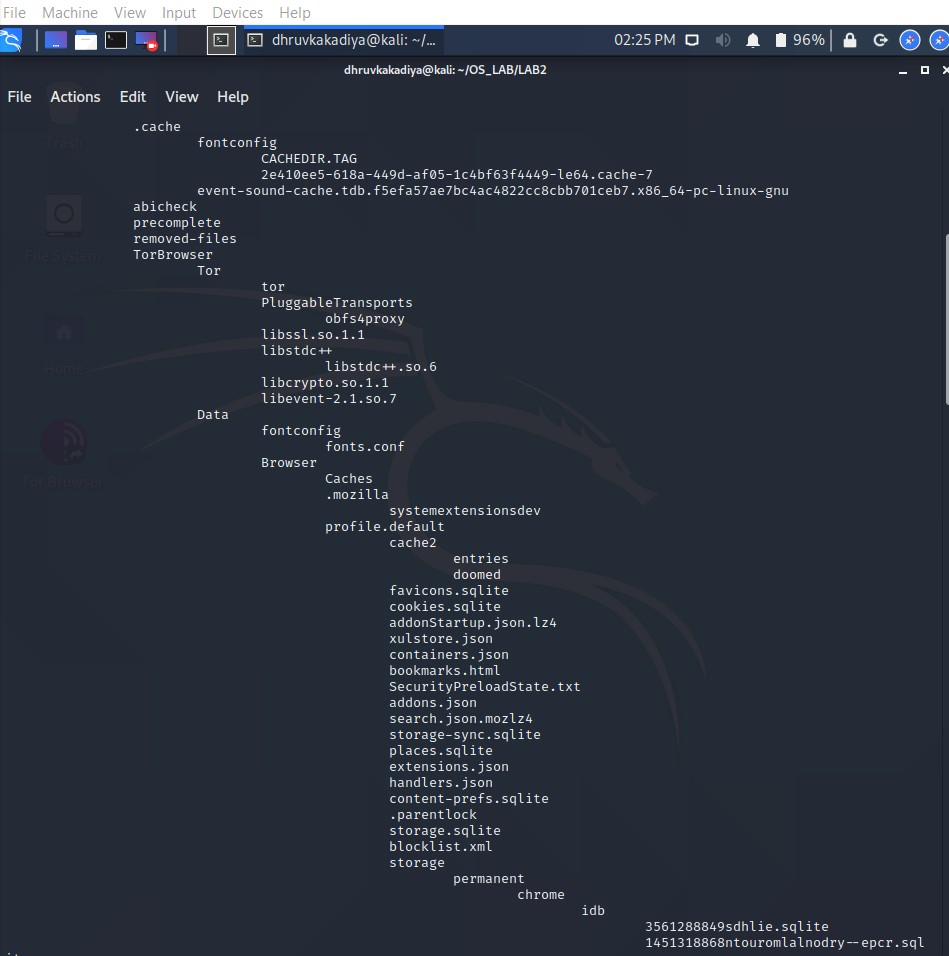
}

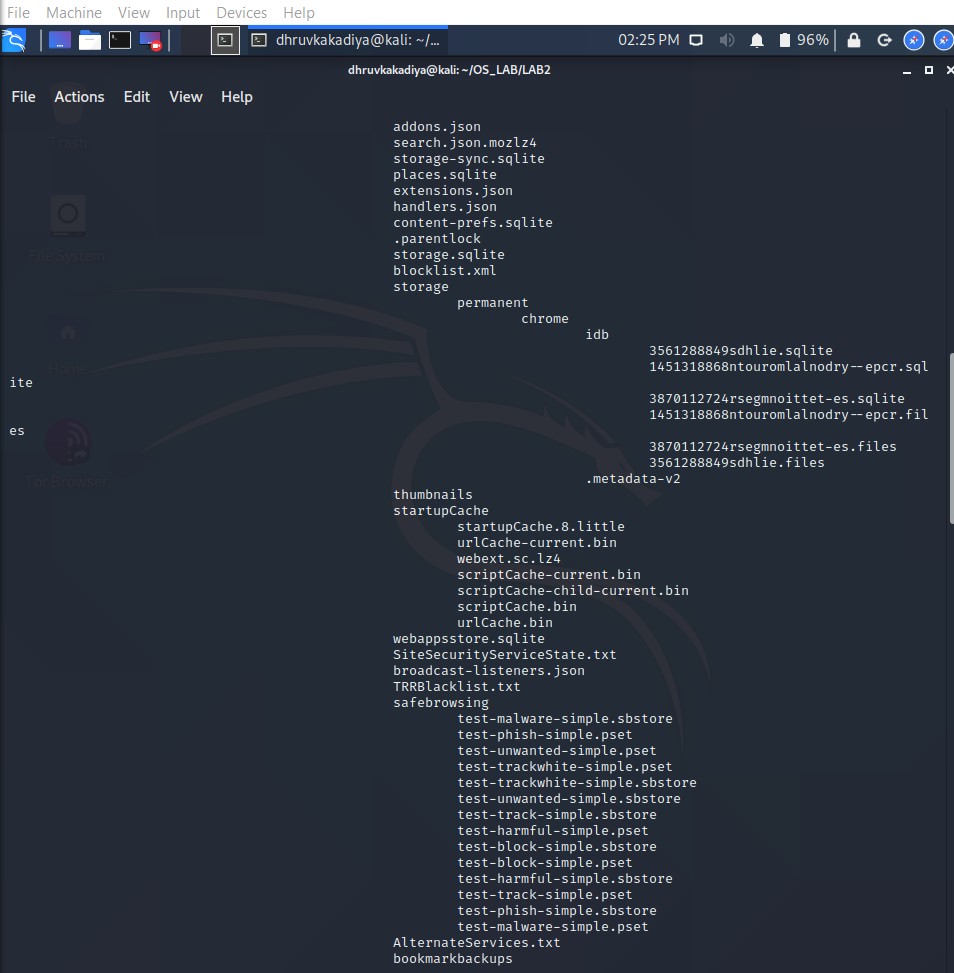
}

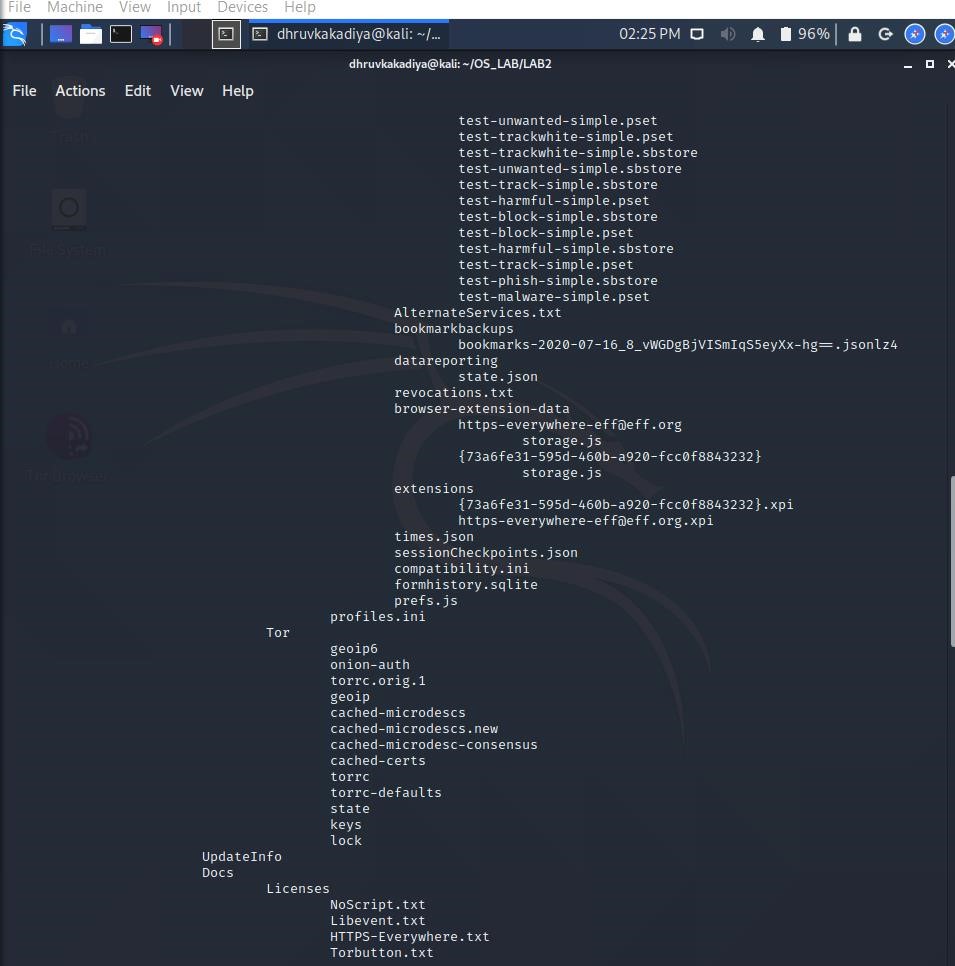
}

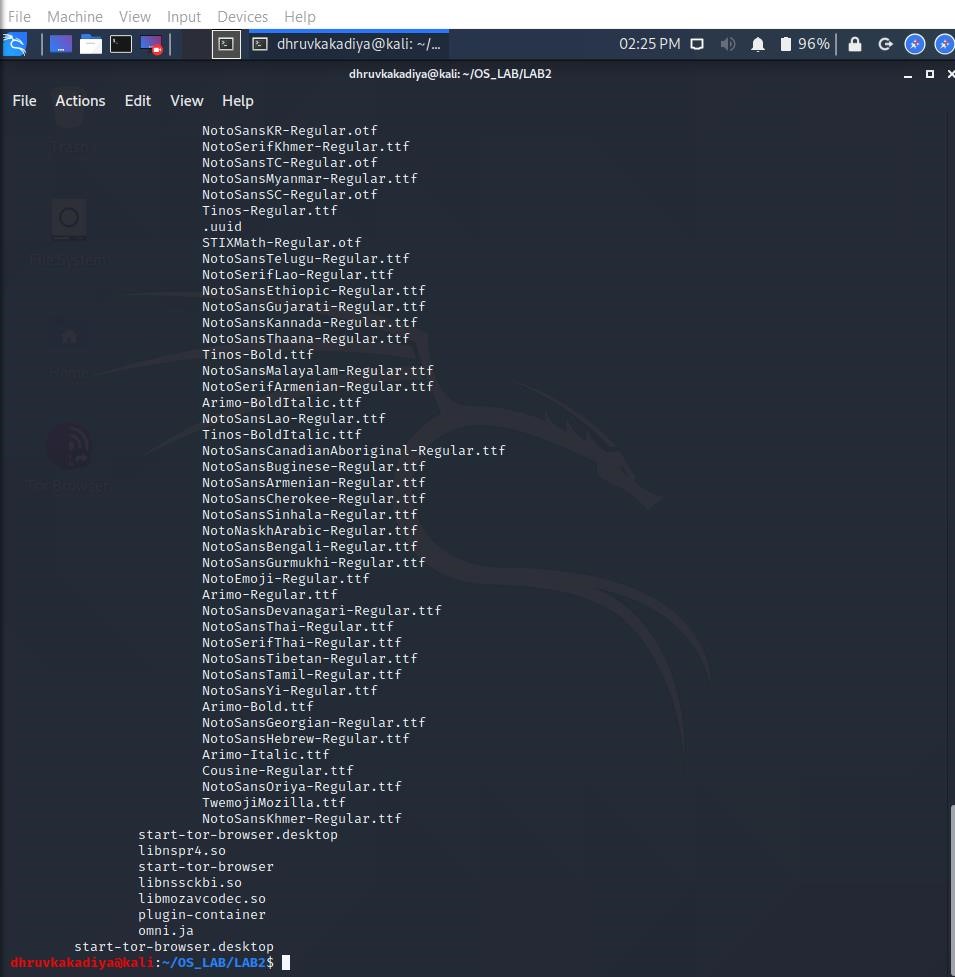
closedir(dir);}











3. Implementation of simple ls cmd.

#include<stdio.h>

#include<dirent.h>

int main()

{

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(1);

   }

while(d=readdir(p))

{

if (strcmp(d->d\_name, “.”) != 0 && strcmp(d->d\_name, “..”) != 0)

   printf("%s\n",d->d\_name);

}

