

OPERATING SYSTEMS

AIM: To understand getcwd, opendir, readdir and closedir functions and implement ls and pwd command.

ROLL NO: CE056

LAB NO: 02

❖ **getcwd function:**

- This function is used to get current working Directory.
- The getcwd() function copies an absolute pathname of the current working directory to the array pointed to by buf, which is of length size.

LIBRARY INCLUDED: unistd.h

SYNTAX:

```
char *getcwd(char *buf, size_t size);  
char *getwd(char *buf);  
char *get_current_dir_name(void);
```

- Syntax 1 is most widely used syntax.
- Getcwd command returns the current working directory of the user.
- It also stores the current working directory of the user in buf.

EXAMPLE: `x = getcwd(buffer,sizeof(buffer));`

❖ readdir function:

- The readdir() function returns a pointer to a dirent structure representing the next directory entry in the directory stream pointed to by dirp.
- It returns NULL on reaching the end of the directory stream or if an error occurred.
- On success, readdir() returns a pointer to a *dirent* structure.

- 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 to indicate the error.
- the dirent structure is defined as follows:

```
struct dirent {  
    ino_t      d_ino;    /* Inode number */  
    off_t      d_off;    /* Not an offset; see below */  
    unsigned short d_reclen; /* Length of this record */  
    unsigned char d_type;  /* Type of file; not  
                           supported by all filesystem types */  
    char d_name[256]; /* Null-terminated filename */  
};
```

- d_ino: This is the inode number of the file.
- d_off: The value returned in d_off is the same as would be returned by calling telldir(3) at the current position in the directory stream.
- d_reclen: This is the size (in bytes) of the returned record.
- d_type: This field contains a value indicating the file type
- d_name :This field contains the null terminated filename.

LIBRARY INCLUDED:, dirent.h

SYNTAX:

```
struct dirent *readdir(DIR *dirp);
```

- The readdir() function returns a pointer to a dirent structure representing the next directory entry in the directory stream pointed to by dirp.
- It returns NULL on reaching the end of the directory stream.
- On success, readdir() returns a pointer to a dirent structure.
- 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.

EXAMPLE: dirent *dptr = readdir(ptr);

Hereby, ptr is a directory pointer.

❖ **opendir function:**

- Opendir() function is used to open a directory.
- The opendir() function opens a directory stream corresponding to the directory name, and returns a pointer to the directory stream. The stream is positioned at the first entry in the directory.

LIBRARY INCLUDED: sys/types.h, dirent.h

SYNTAX:

DIR *opendir(const char *name);

- The opendir() function opens a directory stream corresponding to the directory name, and returns a pointer to the directory stream.
- The stream is positioned at the first entry in the directory.
- On error, NULL is returned, and errno is set appropriately.

EXAMPLE: DIR *ptr = opendir("/home");

❖ **closedir function:**

- `closedir()` function is used to close a directory.
- The `closedir()` function closes the directory stream associated with `dirp`.
- A successful call to `closedir()` also closes the underlying file descriptor associated with `dirp`.
- The directory stream descriptor `dirp` is not available after this call.

LIBRARY INCLUDED: `sys/types.h`, `dirent.h`

SYNTAX:

```
int closedir(DIR *dirp);
```

- The `closedir()` function returns 0 on success.
- On error, -1 is returned, and `errno` is set to indicate the error.

EXAMPLE: `close(ptr)`
 , Where `ptr` is directory pointer.

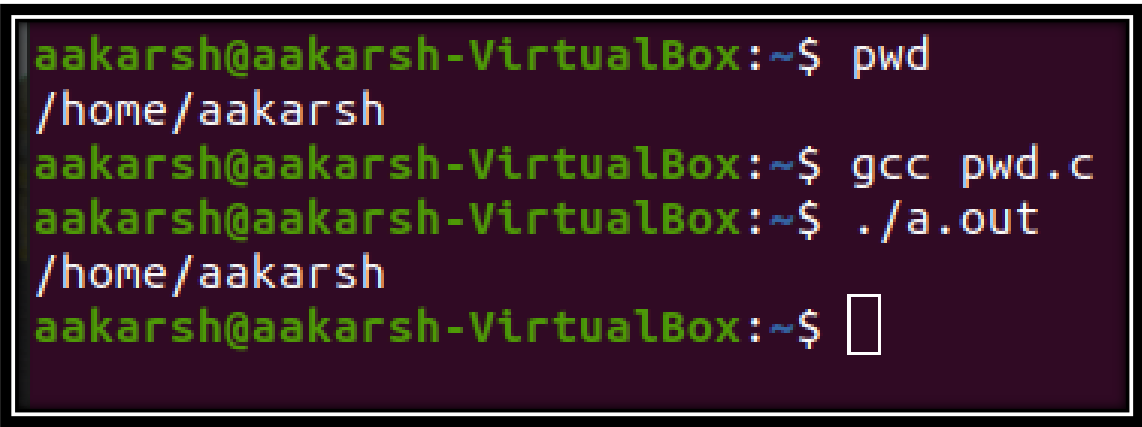
❖ PROGRAMS:

1. Write a program to get current working directory name of the current process. (“pwd” command).

➤ CODE:

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

int main()
{
    char buffer[50];
    char *x;
    x = getcwd(buffer, sizeof(buffer));
    //printf("%s\n", buffer);
    printf("%s\n", x);    //display present working
    directory
}
```

A terminal window with a dark purple background and green text. It shows the execution of a C program. The user runs 'pwd' and gets '/home/aakarsh'. Then they compile 'pwd.c' with 'gcc' and run the resulting 'a.out' file, which also outputs '/home/aakarsh'. The prompt is currently at the end of the last line.

```
aakarsh@aakarsh-VirtualBox:~$ pwd
/home/aakarsh
aakarsh@aakarsh-VirtualBox:~$ gcc pwd.c
aakarsh@aakarsh-VirtualBox:~$ ./a.out
/home/aakarsh
aakarsh@aakarsh-VirtualBox:~$
```

2. Implement a program to list contents of current directory (ls).

➤ CODE:

```
#include<stdio.h>
#include<sys/types.h>
#include<dirent.h>

int main()
{
    char path[100]; //character array to store path
    printf("Please enter path of directory: ");
    scanf("%s",path);
    DIR *ptr = opendir(path); //opening directory
    struct dirent *dptr;
    if(ptr == NULL)
        printf("error");
    while((dptr = readdir(ptr)) != NULL){ //read
directory
        printf("%s\n",dptr->d_name);
    }
    Closedir(ptr);
}
```



```
aakarsh@aakarsh-VirtualBox:~$ gcc opendir.c
aakarsh@aakarsh-VirtualBox:~$ ./a.out
Please enter path of directory: /home
..
aakarsh
.
```

```
aakarsh@aakarsh-VirtualBox:~$ ./a.out
Please enter path of directory: /home/aakarsh
Videos
lsrec.c
a.out
.mozilla
..
.local
.ssh
.cache
Desktop
Music
.sudo_as_admin_successful
.bashrc
Public
Downloads
.bash_logout
.config
opendir.c
lsrecursive.c
Documents
.bash_history
.profile
snap
net.c
Pictures
Folder
.pki
Templates
ls_rcs.c
.
pwd.c
.gnupg
OS
new.c
rec_ls.c
.mas.c.swp
aakarsh@aakarsh-VirtualBox:~$
```

3. Implement a program to demonstrate “ls – R” command.

➤ CODE:

```
#include<unistd.h>
#include<stdio.h>
#include <dirent.h>
#include<string.h>

void display_dir(char path[]);

int main()
{
    char path[100];
    printf("Please enter the path:");
    scanf("%s",path);
    display_dir(path);
}

void display_dir(char path[])
{
```

```

//printf("Inside function");

DIR *open_directory =
opendir(path),*open_subdirectory;

char pth[100];

pth[0]='/';

pth[1]='\0';

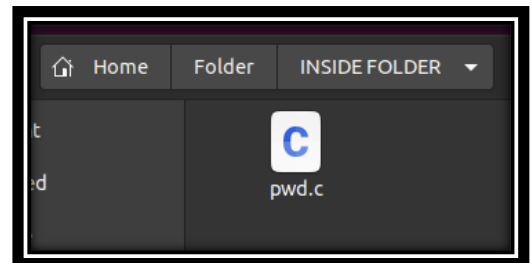
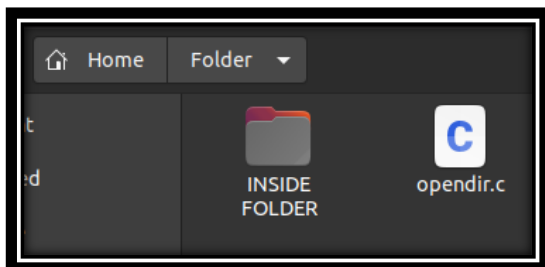
struct dirent *read_directory;

while((read_directory = readdir(open_directory))
!= NULL )
{
    if(read_directory->d_type == DT_DIR)
    {
        if((strcmp(read_directory->d_name, "." ))
&& (strcmp(read_directory->d_name, ".." )))
        {
            strcat(pth,read_directory->d_name);
            strcat(path,pth);
            printf("%s\t\t",read_directory-
>d_name);

            open_subdirectory = opendir(path);
            display_dir(path);

```

```
        closedir(open_subdirectory);
    }
}
else
{
    printf("%s\n",read_directory->d_name);
}
}
closedir(open_directory);
}
```



My directory structure

```
aakarsh@aakarsh-VirtualBox:~/Desktop$ gcc recursivelscmd.c
aakarsh@aakarsh-VirtualBox:~/Desktop$ ./a.out
Please enter the path:/home/aakarsh/Folder
*opendir.c
INSIDE FOLDER          *pwd.c
aakarsh@aakarsh-VirtualBox:~/Desktop$
```

4.Implement combined version of ls command.

➤ CODE:

```
#include<unistd.h>

#include<stdio.h>

#include <dirent.h>

#include<string.h>

#include<stdlib.h>

void display_dir(char path[],int argc,char **argv);

int main(int argc, char **argv)
{
    char path[100];
    printf("Please enter the path:");
    scanf("%s",path);
    //printf("%d\n",argc);
```

```

        display_dir(path,argc,argv);
    }

void display_dir(char path[],int argc,char **argv)
{
    //printf("Inside function");
    DIR *open_directory =
opendir(path),*open_subdirectory;
    char pth[100];
    pth[0]='/';
    pth[1]='\0';
    struct dirent *read_directory;
    //printf("%d\n",argc);
    //printf("%s\n",argv[1]);
    if(argc == 2 && strcmp(argv[1],"-R"))
    {
        printf("Invalid option given");
        exit(0);
    }

    while((read_directory =
readdir(open_directory)) != NULL )

```

```

        {
            if(argc == 2 && !strcmp(argv[1],"-R") &&
read_directory->d_type == DT_DIR)
            {
                if((strcmp(read_directory-
>d_name, "." )) && (strcmp(read_directory-
>d_name, ".." )))
                {
                    strcat(pth,read_directory-
>d_name);

                    strcat(path,pth);
                    printf("%s\t\t",read_directory-
>d_name);

                    open_subdirectory =
opendir(path);

                    display_dir(path,argc,argv);
                    closedir(open_subdirectory);
                }
            }

            else if((strcmp(read_directory-
>d_name, "." )) && (strcmp(read_directory-
>d_name, ".." )))

```

```

        {
            printf("%s\n",read_directory-
>d_name);
        }
        /*else
        {
            continue;
        }*/
    }
    closedir(open_directory);
}

```

➤ Taking directory structure same as previous one.

```

aakarsh@aakarsh-VirtualBox:~/Desktop$ gcc lscombined.c
aakarsh@aakarsh-VirtualBox:~/Desktop$ ./a.out
Please enter the path:/home/aakarsh/Folder
*opendir.c
*INSIDE FOLDER
aakarsh@aakarsh-VirtualBox:~/Desktop$ 

```

```

aakarsh@aakarsh-VirtualBox:~/Desktop$ ./a.out -R
Please enter the path:/home/aakarsh/Folder
*opendir.c
INSIDE FOLDER                *pwd.c

```



```
aakarsh@aakarsh-VirtualBox:~/Desktop$ ./a.out -d
Please enter the path:/
Invalid option givenaakarsh@aakarsh-VirtualBox:~/Desktop$ ./a.out
```