LAB 2 ASSIGNMENT

Name: Birva Babaria

Roll no.: CE010

ID: 19CEUON064

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

Description of System calls:

1) 'getcwd' system call

Library: #include<unistd.h>

Syntax: char *getcwd(char *buf, size_t size);

char *getwd(char *buf);

char *get_current_dir_name(void);

Description: getcwd system call returns the absolute pathname of the current working directory. Pathname is returned by two ways, one that it is returned as a function result and it is returned via the argument if it is present.

Example: x = getcwd(path, sizeof(path));

2) 'opendir' system call

Library: #include<sys/types.h>

#include<dirent.h>

Syntax: DIR *opendir(const char *name);

Description: opendir function opens the directory stream by the address provided as an argument and it returns a pointer to the directory stream. The pointer points to the first entry in the directory. If error occurs then null is returned or else errorno is set appropriately.

Example: DIR *d = opendir(path);

3) ‘readdir’ system call

Library: #include<dirent.h>

Syntax: struct dirent *readdir(DIR *dirp);

Description: readdir function returns the pointer to the dirent structure that represents the next directory entry in the stream. It increments the stream by one directory each time. It returns null at end of the directory stream. Dirent structure contains following variables: d_ino, d_off, d_reclen, d_type (8 is for a regular file and 4 is for directory), d_name[256]. If an error occurs errorno is set appropriately.

Example: struct dirent dir = readdir(d);

4) ‘closedir’ system call

Library: #include<sys/types.h>

#include<dirent.h>

Syntax: int closedir(DIR *dirp);

Description: closedir is used to close the directory stream. It also closes underlying file descriptor. On success, it returns 0. On error, it returns -1 and errorno is set appropriately.

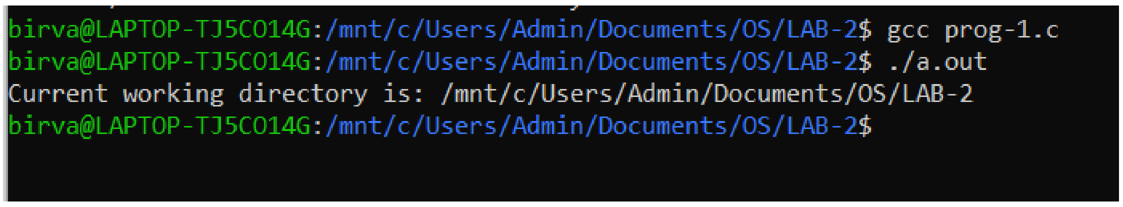
Example: closedir(d)

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(int argc, char *argv[])

{

    char buff[500],*path;

    path = getcwd(buff,sizeof(buff));

    printf("Current working directory is: %s\n",path);

    return 0;
}
```

OUTPUT:



```
birva@LAPTOP-TJ5C014G:/mnt/c/Users/Admin/Documents/OS/LAB-2$ gcc prog-1.c
birva@LAPTOP-TJ5C014G:/mnt/c/Users/Admin/Documents/OS/LAB-2$ ./a.out
Current working directory is: /mnt/c/Users/Admin/Documents/OS/LAB-2
birva@LAPTOP-TJ5C014G:/mnt/c/Users/Admin/Documents/OS/LAB-2$
```

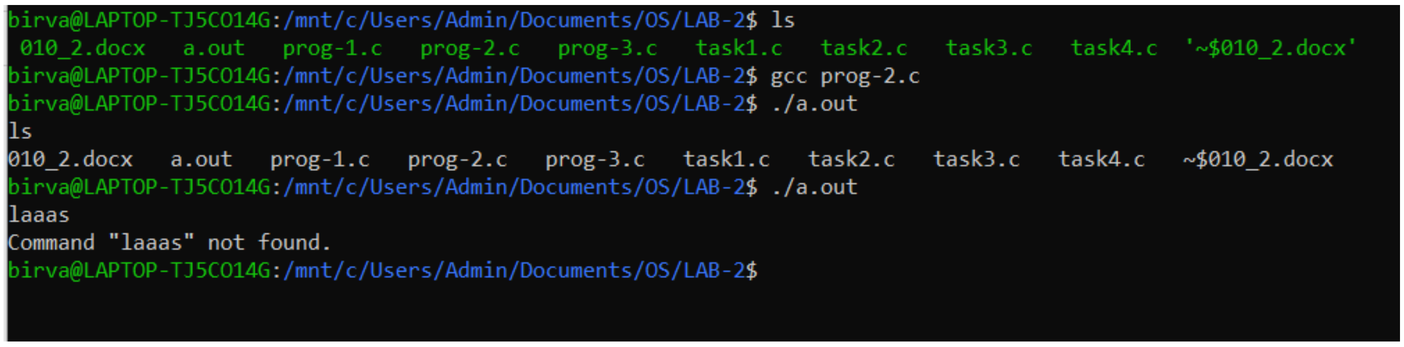
2). Implement a program to list contents of current directory (ls).

CODE:

```
#include<unistd.h>
#include<stdio.h>
#include<dirent.h>
#include<string.h>
int main(int argc, char *argv[])
{
	DIR *d = NULL;
	struct dirent *dir;
	char buff[500];
	scanf("%s",buff);
	if(strcmp(buff,"ls") == 0)
	{
		d = opendir(".");
		if(d != NULL)
		{
			while((dir = readdir(d)) != NULL)
			{
				if((strcmp(dir->d_name,".") == 0) || (strcmp(dir-
>d_name,"..") == 0))
				{
					continue;
```

```
                }
                else
                {
                    printf("%s   ",dir->d_name);
                }
            }
            printf("\n");
            closedir(d);
        }
        else
        {
            printf("Directory can not be opened.\n");
        }
    }
    else
    {
        printf("Command \"%s\" not found.\n",buff);
    }
    return 0;
}
```

OUTPUT:



```
birva@LAPTOP-TJ5CO14G:/mnt/c/Users/Admin/Documents/OS/LAB-2$ ls
 010_2.docx    a.out    prog-1.c    prog-2.c    prog-3.c    task1.c    task2.c    task3.c    task4.c   '~$010_2.docx'
birva@LAPTOP-TJ5CO14G:/mnt/c/Users/Admin/Documents/OS/LAB-2$ gcc prog-2.c
birva@LAPTOP-TJ5CO14G:/mnt/c/Users/Admin/Documents/OS/LAB-2$ ./a.out
ls
010_2.docx    a.out    prog-1.c    prog-2.c    prog-3.c    task1.c    task2.c    task3.c    task4.c    ~$010_2.docx
birva@LAPTOP-TJ5CO14G:/mnt/c/Users/Admin/Documents/OS/LAB-2$ ./a.out
laaas
Command "laaas" not found.
birva@LAPTOP-TJ5CO14G:/mnt/c/Users/Admin/Documents/OS/LAB-2$
```

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

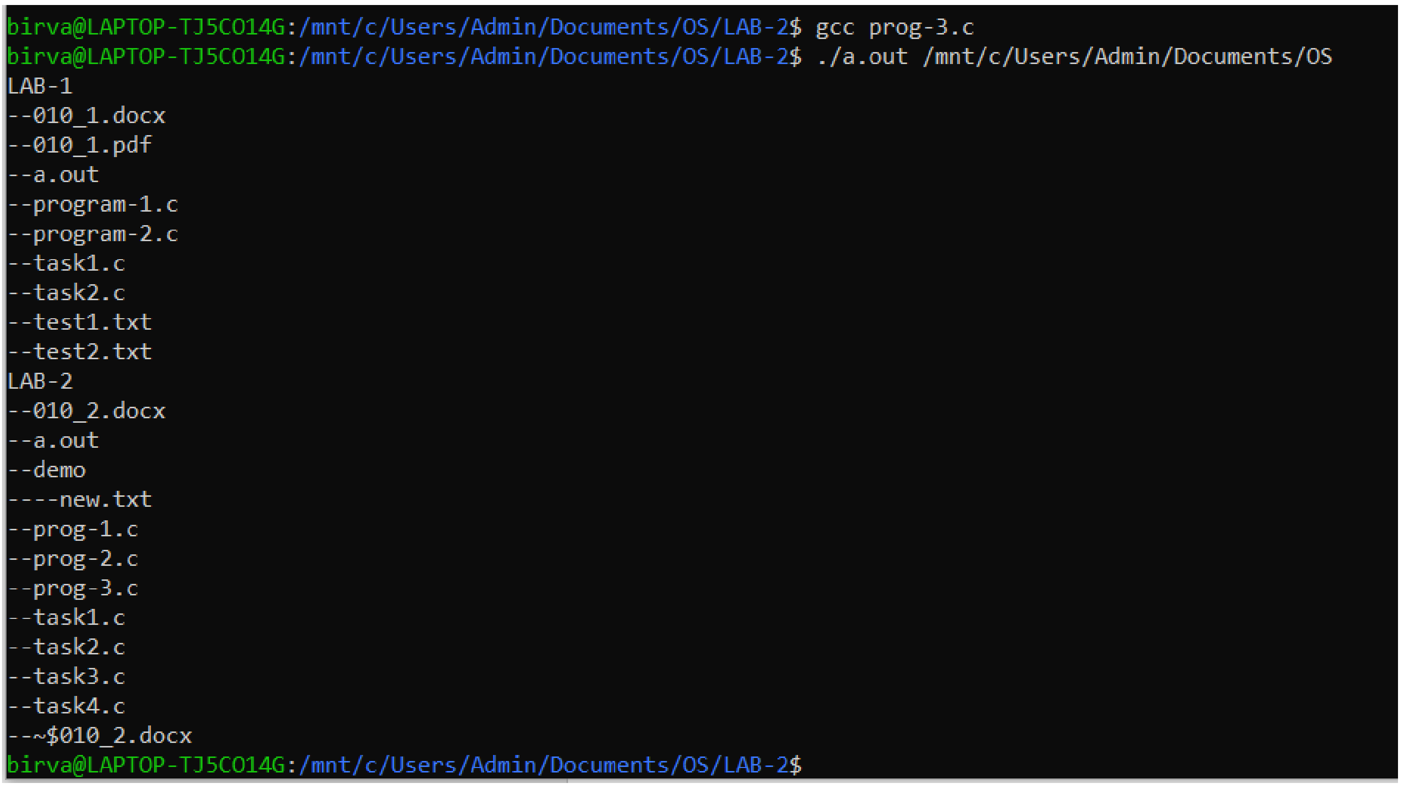
CODE:

```
#include<unistd.h>
#include<sys/types.h>
#include<stdio.h>
#include<dirent.h>
#include<string.h>
int count = 0;
void printFiles(char *path)
{
     DIR *dir;
     struct dirent *dptr;
     char temp[100];
     dir = opendir(path);
     strcpy(temp,path);
     if(dir != NULL)
     {
          while ((dptr = readdir(dir)) != NULL)
          {
               if((dptr->d_name)[0] != '.' && ((dptr->d_type) == 4
|| (dptr->d_type) == 8))
               {
                    int i=0;
                    while(i<count)
                    {
                         printf("--");
                         i++;
                    }
                    printf("%s\n",dptr->d_name);
               }
```

```
                if((dptr->d_name)[0] != '.' && (dptr->d_type) == 4)
                {
                    strcat(path,"/");
                    strcat(path,dptr->d_name);
                    strcat(path,"\0");
                    count++;
                    printFiles(path);
                    count--;
                    strcpy(path,temp);
                }
            }
            closedir(dir);
        }
}
int main(int argc, char *argv[])
{
    char buff[200];
    getcwd(buff,sizeof(buff));
    if(argc == 1)
    {
        printFiles(buff);
    }
    else if(argc == 2)
    {
        printFiles(argv[1]);
    }
    else
    {
        printf("Incorrect arguments provided.");
    }
    return 0;
}
```

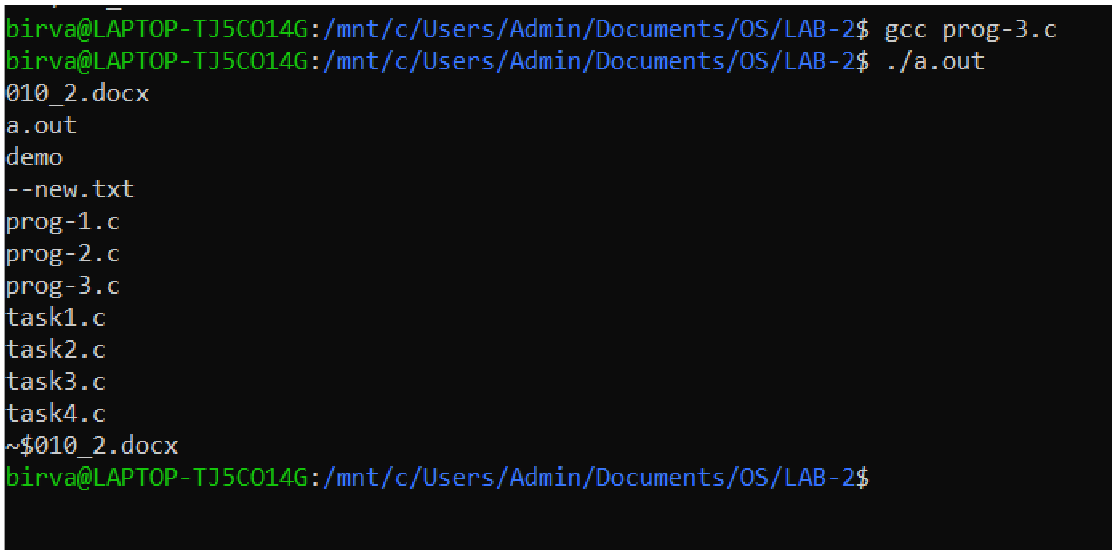
OUTPUT:

If a path is provided in command line argument.



```
birva@LAPTOP-TJ5CO14G:/mnt/c/Users/Admin/Documents/OS/LAB-2$ gcc prog-3.c
birva@LAPTOP-TJ5CO14G:/mnt/c/Users/Admin/Documents/OS/LAB-2$ ./a.out /mnt/c/Users/Admin/Documents/OS
LAB-1
--010_1.docx
--010_1.pdf
--a.out
--program-1.c
--program-2.c
--task1.c
--task2.c
--test1.txt
--test2.txt
LAB-2
--010_2.docx
--a.out
--demo
----new.txt
--prog-1.c
--prog-2.c
--prog-3.c
--task1.c
--task2.c
--task3.c
--task4.c
--~$010_2.docx
birva@LAPTOP-TJ5CO14G:/mnt/c/Users/Admin/Documents/OS/LAB-2$
```

If path is not provided in command line argument (current path is considered).



```
birva@LAPTOP-TJ5CO14G:/mnt/c/Users/Admin/Documents/OS/LAB-2$ gcc prog-3.c
birva@LAPTOP-TJ5CO14G:/mnt/c/Users/Admin/Documents/OS/LAB-2$ ./a.out
010_2.docx
a.out
demo
--new.txt
prog-1.c
prog-2.c
prog-3.c
task1.c
task2.c
task3.c
task4.c
~$010_2.docx
birva@LAPTOP-TJ5CO14G:/mnt/c/Users/Admin/Documents/OS/LAB-2$
```

4). Implement a program that works as both ls and ls –R command.

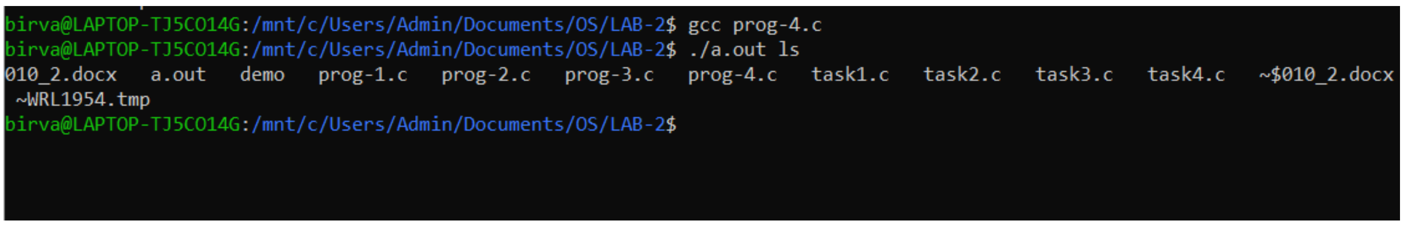
CODE:

```
#include<unistd.h>

#include <sys/types.h>

#include<stdio.h>

#include<dirent.h>

#include<string.h>

int count = 0;

void displayLsR(char *path)

{

     DIR *dir;

     struct dirent *dptr;

     char temp[100];

     dir = opendir(path);

     strcpy(temp,path);

     if(dir != NULL)

     {

          while ((dptr = readdir(dir)) != NULL)

          {

               if((dptr->d_name)[0] != '.' && ((dptr->d_type) == 4
|| (dptr->d_type) == 8))

               {

                    int i=0;

                    while(i<count)

                    {

                         printf("--");

                         i++;

                    }

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

               }
```

```
                if((dptr->d_name)[0] != '.' && (dptr->d_type) == 4)
                {
                    strcat(path,"/");
                    strcat(path,dptr->d_name);
                  strcat(path,"\0");
                  count++;
                    displayLsR(path);
                    count--;
                    strcpy(path,temp);
                }
            }
            closedir(dir);
        }
}
void displayLs(char *path)
{
    DIR *dir;
    struct dirent *dptr;
    dir = opendir(path);
    if(dir != NULL)
        {
            while((dptr = readdir(dir)) != NULL)
            {
                if((strcmp(dptr->d_name,".") == 0) ||
(strcmp(dptr->d_name,"..") == 0))
                {
                    continue;
                }
                else
                {
                    printf("%s   ",dptr->d_name);
                    }
                }
                printf("\n");
                closedir(dir);
            }
            else
            {
                printf("Directory can not be opened.\n");
            }
}
int main(int argc, char *argv[])
{
        char buff[200];
        getcwd(buff,sizeof(buff));
        if(argc == 2 && strcmp(argv[1],"ls")==0)
        {
            displayLs(buff);
        }
        else if(argc == 3 && (strcmp(argv[1],"ls")==0 &&
strcmp(argv[2],"-R")==0))
        {
            displayLsR(buff);
        }
        else
        {
            printf("Invalid command.\n");
        }
        return 0;
}
```

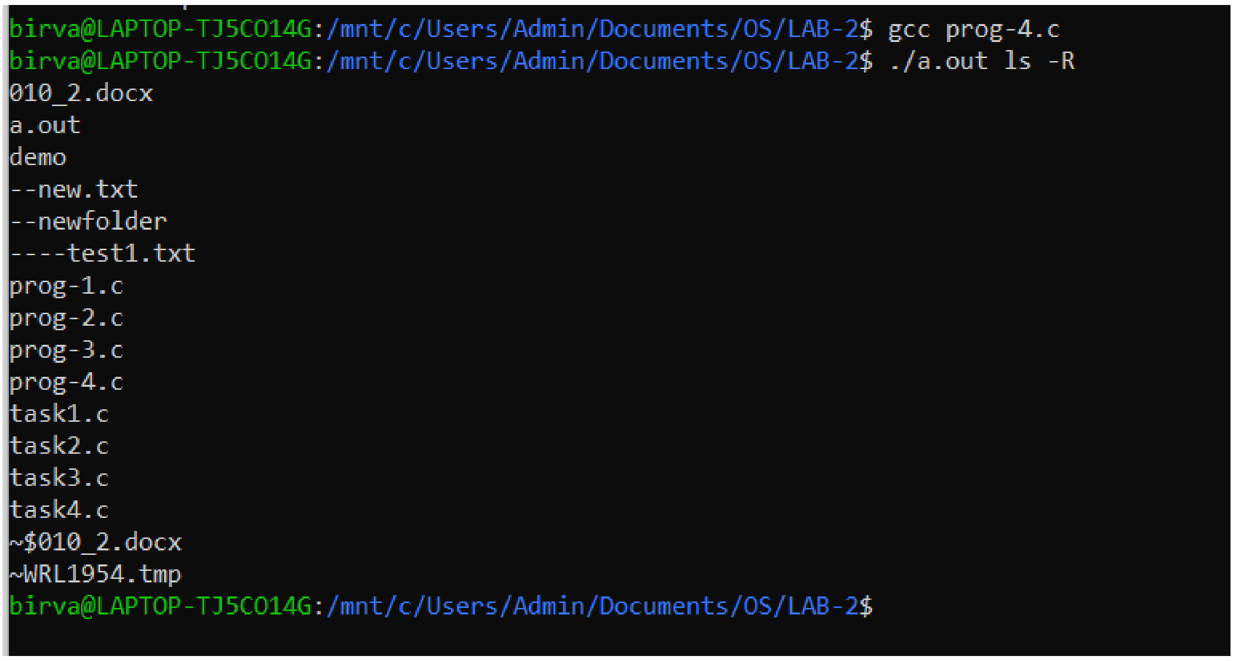
OUTPUT:

If ‘ls’ is provided as command.



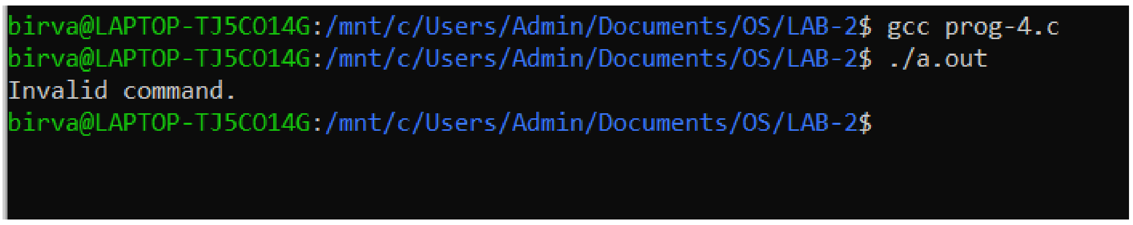
```
birva@LAPTOP-TJ5CO14G:/mnt/c/Users/Admin/Documents/OS/LAB-2$ gcc prog-4.c
birva@LAPTOP-TJ5CO14G:/mnt/c/Users/Admin/Documents/OS/LAB-2$ ./a.out ls
010_2.docx    a.out    demo    prog-1.c    prog-2.c    prog-3.c    prog-4.c    task1.c    task2.c    task3.c    task4.c    ~$010_2.docx
 ~WRL1954.tmp
birva@LAPTOP-TJ5CO14G:/mnt/c/Users/Admin/Documents/OS/LAB-2$
```

If ‘ls –R’ is provided as command.



```
birva@LAPTOP-TJ5CO14G:/mnt/c/Users/Admin/Documents/OS/LAB-2$ gcc prog-4.c
birva@LAPTOP-TJ5CO14G:/mnt/c/Users/Admin/Documents/OS/LAB-2$ ./a.out ls -R
010_2.docx
a.out
demo
--new.txt
--newfolder
----test1.txt
prog-1.c
prog-2.c
prog-3.c
prog-4.c
task1.c
task2.c
task3.c
task4.c
~$010_2.docx
~WRL1954.tmp
birva@LAPTOP-TJ5CO14G:/mnt/c/Users/Admin/Documents/OS/LAB-2$
```

Any other command.



```
birva@LAPTOP-TJ5CO14G:/mnt/c/Users/Admin/Documents/OS/LAB-2$ gcc prog-4.c
birva@LAPTOP-TJ5CO14G:/mnt/c/Users/Admin/Documents/OS/LAB-2$ ./a.out
Invalid command.
birva@LAPTOP-TJ5CO14G:/mnt/c/Users/Admin/Documents/OS/LAB-2$
```