Experiment no: 4

Name: Saloni Vishwakarma
Batch-Roll no: C1-13
Semester-Section:4th-C
Subject: OS(Operating System) Lab
Date of execution: 7 June 2023

Aim: Write a C program to simulate Directory management system calls of Linux

a) Create directory (mkdir command)
b) List the number of files in current directory(ls command)
c) Navigate the directory (cd command)
d) Remove directory (rmdir command)
e) Rename directory

Theory:

Directory system calls are functions or methods provided by an operating system that allow programs to interact with directories and manipulate their contents. These system calls enable operations such as creating, opening, closing, reading, writing, renaming, and deleting directories and files within them. The specific set of directory system calls may vary depending on the operating system, but I will provide some commonly used ones found in Unix-like systems:

1. `mkdir`: This system call is used to create a new directory within the file system. It takes a path as an argument and creates a new directory with the specified name at the given location.
2. `rmdir`: This system call removes an empty directory from the file system. It takes a path as an argument and deletes the directory if it is empty.
3. `opendir`: The `opendir` system call opens a directory for reading. It takes a path as an argument and returns a directory stream, which is used for subsequent directory operations.
4. `readdir`: This system call reads the next entry from an open directory. It takes a directory stream as an argument and returns the name and attributes of the next directory entry.

5. `closedir`: The `closedir` system call closes a directory stream that was previously opened with `opendir`. It frees up system resources associated with the directory stream.
6. `rename`: This system call is used to rename a file or directory. It takes two path arguments, the old name and the new name, and renames the file or directory accordingly.
7. `chdir`: The `chdir` system call changes the current working directory of the process to the specified directory.
8. `getcwd`: This system call retrieves the current working directory for the calling process.
9. `unlink`: The `unlink` system call deletes a file from the file system. It takes a path as an argument and removes the file if it exists.

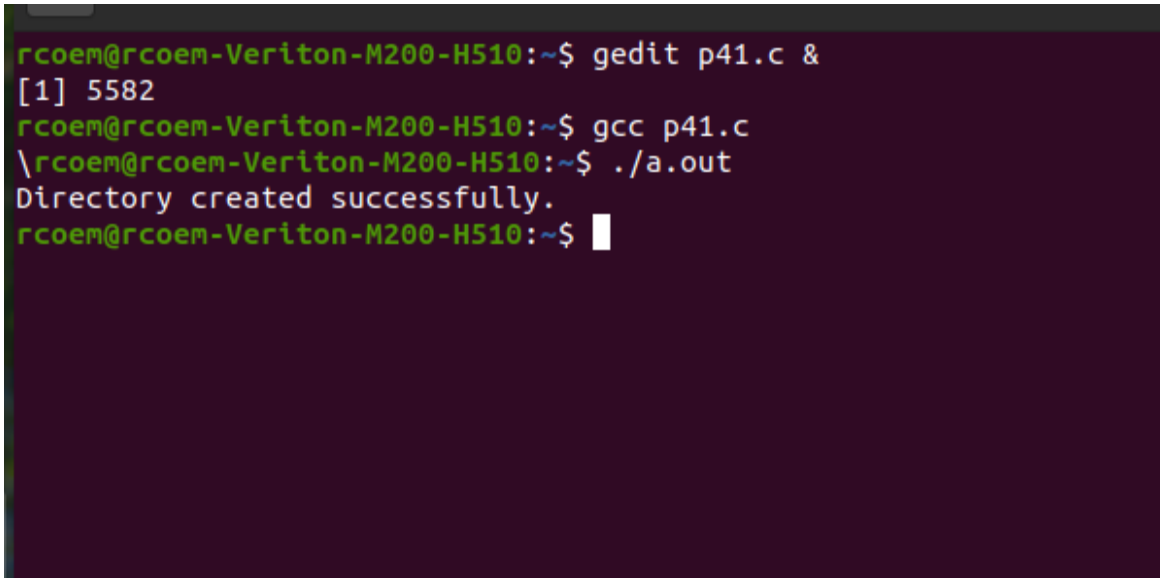
These are just a few examples of directory system calls commonly found in Unix-like operating systems. Other operating systems, such as Windows, may provide similar functionality with different system calls.

Code and Output:

a) Create directory (mkdir command)

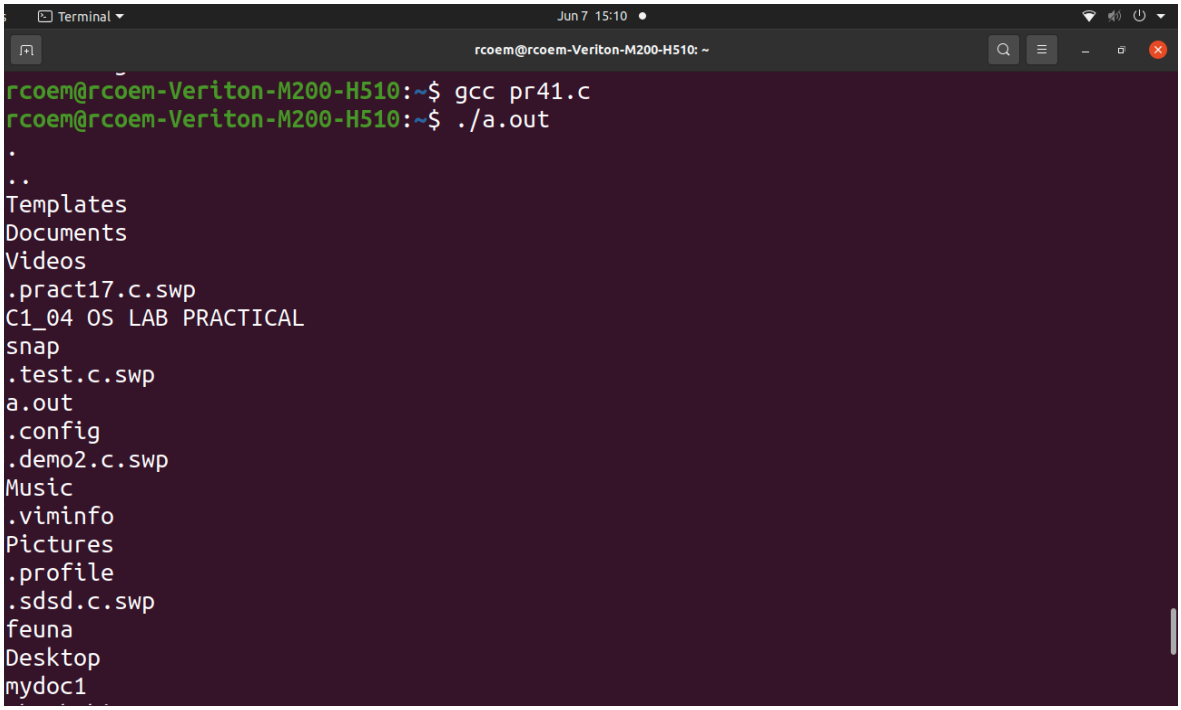
```
#include <stdlib.h>
#include<stdio.h>

int main() {
        char* directoryName = "kingdom";
        int result = system("mkdir kingdom");
        if (result == 0) {
        printf("Directory created successfully.\n");
        } else {
        printf("Failed to create directory.\n");
        }
        return 0;
}
```



b) Listing files in a directory

```
#include <dirent.h>
#include <stdio.h>
int main(void)
{
  DIR *d;
  struct dirent *dir;
  d = opendir(".");
  if (d)
  {
          while ((dir = readdir(d)) != NULL)
          {
          printf("%s\n", dir->d_name);
          }
          closedir(d);
  }
  return(0);
}
```



c) Navigate the directory (cd command)

```
#include <stdio.h>
#include <unistd.h>
int main() {
        char directory[100];
        printf("Enter the directory path: ");
        scanf("%s", directory);
        if (chdir(directory) == 0) {
        printf("Directory changed successfully.\n");
        } else {
        printf("Failed to change directory.\n");
        }
        return 0;
}
```

d)Remove directory (rmdir command)

```
#include <stdio.h>
#include <unistd.h>
int main() {
        char directory[100];
        printf("Enter the directory path to remove: ");
        scanf("%s", directory);
        int status = rmdir(directory);
        if (status == 0) {
        printf("Directory removed successfully.\n");
        } else {
        printf("Failed to remove directory.\n");
        }
        return 0;
}
```

e) Rename a directory:

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

int main() {
    char oldName[100], newName[100];

    printf("Enter the current name of the directory: ");
    scanf("%s", oldName);

    printf("Enter the new name for the directory: ");
    scanf("%s", newName);

    // Construct the command to rename the directory using the mv command
    char command[200];
    sprintf(command, "mv %s %s", oldName, newName);

    // Execute the command using system()
    int status = system(command);

    if (status == -1) {
        printf("Failed to rename the directory.\n");
        exit(EXIT_FAILURE);
```

```
    }
    printf("Directory renamed successfully!\n");
    return 0;
}
```

```
[2] 7649
rcoem@rcoem-Veriton-M200-H510:~$ gcc rename.c
rename.c: In function ‘main’:
rename.c:15:29: warning: ‘%s’ directive writing up to 99 bytes into a regi
between 97 and 196 [-Wformat-overflow=]
   15 |     sprintf(command, "mv %s %s", oldName, newName);
      |                             ^~           ~~~~~~~
rename.c:15:5: note: ‘sprintf’ output between 5 and 203 bytes into a desti
ize 200
   15 |     sprintf(command, "mv %s %s", oldName, newName);
      |     ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[2]+  Done                    gedit rename.c
rcoem@rcoem-Veriton-M200-H510:~$ ./a.out
Enter the current name of the directory: birth
Enter the new name for the directory: death
Directory renamed successfully!
rcoem@rcoem-Veriton-M200-H510:~$
```

Conclusion: We have successfully studied and implemented the directory system calls using C. By using these system calls, developers can perform various operations on directories, such as creating new directories, listing the contents of a directory, changing the current working directory, and removing directories and files.