K.GOWTHAM-192372078

25.Construct a C program to implement the I/O system calls of UNIX (fcntl, seek, stat, opendir, readdir)

Aim:

To implement a C program that demonstrates the usage of UNIX I/O system calls like `fcntl`, `seek`, `stat`, `opendir`, and `readdir`.

Algorithm:

1. Open a file using `open` system call.
2. Use `fcntl` to manipulate file descriptor properties.
3. Use `lseek` to reposition the file offset.
4. Use `stat` to retrieve file status information.
5. Use `opendir` to open a directory and `readdir` to read its contents.
6. Display the results of each operation.

Procedure:

1. Include the necessary headers (`fcntl.h`, `unistd.h`, `sys/stat.h`, etc.).
2. Use appropriate system calls to perform file and directory operations.
3. Handle errors appropriately (e.g., check return values).
4. Display the results of the operations.

Code:

```
#include <fcntl.h>

#include <unistd.h>

#include <sys/stat.h>

#include <dirent.h>

#include <stdio.h>



int main() {

   int fd;

   struct stat fileStat;
```

```
DIR *dir;
struct dirent *entry;

fd = open("testfile.txt", O_RDWR | O_CREAT, 0644);
if (fd == -1) return 1;

if (fcntl(fd, F_SETFL, O_APPEND) == -1) return 1;

off_t offset = lseek(fd, 10, SEEK_SET);
if (offset == -1) return 1;

if (stat("testfile.txt", &fileStat) == -1) return 1;

dir = opendir(".");
if (!dir) return 1;

while ((entry = readdir(dir)) != NULL) {
    printf("%s\n", entry->d_name);
}

closedir(dir);
close(fd);
return 0;
```

}

Result:

1. A file named `testfile.txt` is created or opened.
2. File descriptor properties are modified using `fcntl`.
3. The file offset is repositioned using `lseek`.
4. File details like size and permissions are fetched using `stat`.
5. Directory contents are listed using `opendir` and `readdir`.

Output: