**Code :**

#include <stdio.h>

#include <stdlib.h>

#include <fcntl.h>

#include <unistd.h>

#include <string.h>

#include <errno.h>

/\* Path to the device file in /dev \*/

#define DEVICE\_PATH "/dev/dev1”

/\* Buffer size for read/write operations \*/

#define BUFFER\_SIZE 1024

int main() {

int fd;

char write\_data[BUFFER\_SIZE] = "Hello, Character Device!";

char read\_data[BUFFER\_SIZE];

ssize\_t bytes\_written, bytes\_read;

/\* Open the device file \*/

fd = open(DEVICE\_PATH, O\_RDWR);

if (fd < 0) {

perror("Failed to open the device");

return errno;

}

printf("Device opened successfully.\n");

/\* Write data to the device \*/

bytes\_written = write(fd, write\_data, strlen(write\_data));

if (bytes\_written < 0) {

perror("Failed to write to the device");

close(fd);

return errno;

}

printf("Data written to the device: %s\n", write\_data);

/\* Reset read\_data buffer and file offset \*/

memset(read\_data, 0, BUFFER\_SIZE);

/\* Reset offset to the beginning of the device \*/

lseek(fd, 0, SEEK\_SET);

/\* Read data from the device \*/

bytes\_read = read(fd, read\_data, BUFFER\_SIZE);

if (bytes\_read < 0) {

perror("Failed to read from the device");

close(fd);

return errno;

}

printf("Data read from the device: %s\n", read\_data);

/\* Close the device file \*/

close(fd);

printf("Device closed successfully.\n");

return 0;

}

**Explanation of the Application Code**

1. **Opening the Device**: The application opens the device file located at /dev/dev1. Ensure that the device file exists and has the correct permissions. If it doesn’t open successfully, an error message is printed.
2. **Writing to the Device**: The application writes a string, "Hello, Character Device!", to the device using the write system call. The number of bytes written is checked to ensure that the write was successful.
3. **Reading from the Device**: The lseek function resets the file offset to the beginning so that we can read back what was written. The read system call reads data from the device into a buffer.
4. **Closing the Device**: The application closes the device after performing the read and write operations.

**Compilation and Execution**

1. Compile the application with:

gcc -o char\_driver\_app char\_driver\_app.c

1. Run the application with appropriate permissions (root, if necessary, since it interacts with a device file):

sudo ./char\_driver\_app

1. **Expected Output**:

The application will print messages indicating that it successfully opened the device, wrote data, read it back, and then closed the device. You should see output similar to:

Device opened successfully.

Data written to the device: Hello, Character Device!

Data read from the device: Hello, Character Device!

Device closed successfully.

Ensure the device file /dev/dev1 is properly created by the driver and accessible before running the application. This code provides a basic example of how to interface with a character device driver from user space.