In Linux, `open()` and `write()` are fundamental system calls used for file input/output (I/O) operations.

1. `open()` System Call:

The `open()` system call is used to open or create a file and obtain a file descriptor. A file descriptor is a non-negative integer that uniquely identifies an open file within a process.

C
```c
#include <fcntl.h> // For open flags like O_RDONLY, O_WRONLY, O_CREAT, etc.
#include <sys/stat.h> // For file permissions (mode argument)

int open(const char *pathname, int flags, ... /* mode_t mode */);
```

- `pathname`: The path to the file to be opened or created.
- `flags`: Specifies how the file should be opened (e.g., read-only, write-only, read-write, create if not exists, truncate if exists). Common flags include `O_RDONLY`, `O_WRONLY`, `O_RDWR`, `O_CREAT`, `O_TRUNC`, `O_APPEND`. These flags can be combined using the bitwise OR operator (`|`).
- `mode`: (Optional) If `O_CREAT` is included in `flags`, this argument specifies the file permissions for the newly created file (e.g., `0644` for read/write for owner, read-only for group and others).

`open()` returns a file descriptor on success, or -1 if an error occurs.

2. `write()` System Call:

The `write()` system call is used to write data from a buffer to a file associated with a file descriptor.

C
```c
#include <unistd.h> // For write()

ssize_t write(int fd, const void *buf, size_t count);
```

- `fd`: The file descriptor of the file to which data will be written, obtained from a successful `open()` call.
- `buf`: A pointer to the buffer containing the data to be written.
- `count`: The number of bytes to write from the buffer.

`write()` returns the number of bytes actually written on success, which may be less than `count` in certain cases (e.g., disk full, signal interruption). It returns -1 if an error occurs.

Example Usage:

C
```c
#include <fcntl.h>
#include <unistd.h>
#include <string.h>

int main() {
    int fd;
    const char *data = "Hello, Linux I/O!";
```

```
    ssize_t bytes_written;

    // Open a file for writing, create if it doesn't exist, truncate if it
does
    fd = open("output.txt", O_WRONLY | O_CREAT | O_TRUNC, 0644);
    if (fd == -1) {
        // Handle error
        return 1;
    }

    // Write data to the file
    bytes_written = write(fd, data, strlen(data));
    if (bytes_written == -1) {
        // Handle error
        close(fd);
        return 1;
    }

    // Close the file
    close(fd);

    return 0;
}
```