# 【Linux Programming】Day12

| | |
|---|---|
| Tags | |
| Date | @June 4, 2022 |
| Summary | open, close, read, write, and ioctl |

## 【Ch3】Working with Files

### 3.3 Low-level File Access

#### 3.3.1 write

The `write` system call arranges for the first `nbytes` bytes from `buf` to be written to the file associated with the file descriptor `fildes`.

It returns the number of bytes actually written. If it returns -1, there has been an error in the write call, and the error will be specified in the `errno` global variable.

Here's the syntax:

```
#include <unistd.h>

size_t write(int fildes, const void *buf, size_t nbytes);
```

*Note: write might report that it wrote fewer bytes than we asked it to. This is not necessarily an error.*

#### 3.3.2 read

The `read` system call reads up to `nbytes` bytes of data from the file associated with the file descriptor `fildes` and places them in the data area `buf`.

```
#include <unistd.h>

size_t read(int fildes, void *buf, size_t nbytes);
```

### 3.3.3 open

To create a new file descriptor, we need to use the `open` system call:

```
#include <fcntl.h>

int open(const char* path, int oflags);
int open(const char* path, int oflags, mode_t mode);
```

If open is successful, it returns a file descriptor that can be used in read, write, and other system calls.

The name of the file or device to be opened is passed as `path`; the `oflags` parameter is used to specify actions to be taken on opening the file.

| Mode | Description |
|---|---|
| O_RDONLY | Open for read-only |
| O_WRONLY | Open for write-only |
| O_RDWR | Open for reading and writing |

The call may also include a combination of the following optional modes in the `oflags` parameter:

- `O_APPEND`: Place written data at the end of the file
- `O_TRUNC`: Set the length of the file to zero, discarding existing contents
- `O_CREAT`:  Creates the file with permissions given in mode.
- `O_EXCL`: Used with `O_CREAT`, ensures that the caller create the file.

There is also a `creat` system call, which is equivalent of calling open with `oflags` equal to `O_CREAT|O_WRONLY|O_TRUNC`.

*Initial Permissions*

When we create a file using `O_CREAT` flag with open, we must use the three-parameter form.

mode, the third parameter, is made from a bitwise OR of the flags defined in the header file `sys/stat.h`. These are:

- S_IRUSR: Read permission, owner
- S_IWUSR: Write permission, owner
- S_IXUSR: Execute permission, user
- S_IRGRP: Read permission, group
- S_IWGRP: Write permission, group
- S_IXGRP: Execute permission, group
- S_IROTH: Read permission, others
- S_IWOTH: Write permission, others
- S_IXOTH: Execute permission, others

For example,

```
open("my_file", O_CREAT, S_IRUSR | S_IXOTH);
```

has the effect of creating a file called `my_file`, with read permission for the owner and execute permission for others, and only these permission:

### 3.3.4 unmask

The `unmask` is a system variable that encodes a mask for file permissions to be used when a file is created.

We can change the variable by executing the unmask command to supply a new value.

| Digit | Value | Meaning |
|---|---|---|
| 1 | 0 | No user permissions are to be disallowed. |
| | 4 | User read permission is disallowed. |
| | 2 | User write permission is disallowed. |
| | 1 | User execute permission is disallowed. |
| 2 | 0 | No group permissions are to be disallowed. |
| | 4 | Group read permission is disallowed. |

2



| Digit | Value | Meaning |
|---|---|---|
| | 2 | Group write permission is disallowed. |
| | 1 | Group execute permission is disallowed. |
| 3 | 0 | No other permissions are to be disallowed |
| | 4 | Other read permission is disallowed. |
| | 2 | Other write permission is disallowed. |
| | 1 | Other execute permission is disallowed. |



When we create a file via an `open` or `creat` call, the mode parameter is compared with the current unmask. Any bit setting in the mode parameter that is also set in the unmask is removed.

### 3.3.5 close

We ucse close to terminate the association between a file descriptor and its file.

```
#include <unistd.h>

int close(int filedes);
```

### 3.3.6 ioctl

`ioctl` performs the function indicated by `cmd` on the object referenced by the descriptor `fildes`.

```
#include <unistd.h>

int ioctly(int fildes, int cmd, ...);
```