【Linux Programming】Day14(2)

Tags	
Date	@June 6, 2022
Summary	Stream Errors and File and Directory Maintenance

【Ch3】Work with Files

3.6.3 Stream Errors

To indicate error, many `stdio` library functions return out-of-ranges values, such as null pointers or the constant EOF.

In these cases, the error is indicated in the external variable `errno`:

```
#include <errno.h>

extern int errno;
```

We can also interrogate the state of a file stream to determine whether an error occurred:

```
#include <stdio.h>

int ferror(FILE *stream);
int feof(FILE *stream);
void clearerr(FILE *stream);
```

The `ferror` function tests the error indicator for a stream and return nonzero if it's set, but zero otherwise.

The `feof` function tests the end-of-file indicator within a stream and returns nonzero if it is set, zero otherwise.

```
if (feof(some_stream))
  do something...
```

3.7 File and Directory Maintenance

3.7.1 chmod

We can change the permissions on a file or directory using the chmod system call.

```
#include <sys/stat.h>

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

For example,

```
chmod("./tmp.txt", S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH);
```

Allows read and write by the owner and read by the group user and others.

3.7.2 chown

A superuser can change the owner of a file using the `chown` system call.

```
#include <sys/types.h>
#include <unistd.h>

int chown(const char *path, uid_t owner, gid_t group);
```

3.7.3 unlink, link, and symlink

We can remove a file using `unlink`. The unlink system call removes the directory entry for a file and decrements the link count for it.

It returns 0 if the unlinking was successful, -1 on an error. We must have write and execute permissions in the directory where the file has its directory entry for this call to function

```
#include <unistd.h>

int unlink(const char *path);
int link(const char *path1, const char *path2);
int symlink(const char *path1, const char *path2);
```

We can create new links to a file by using the `link` system call. The link to an existing file is path1 and the new directory entry is specified by path2.

3.7.4 mkdir and rmdir

We can create and remove directories using the `mkdir` and `rmdir` system calls.

```
#include <sys/types.h>
#include <sys/stat.h>

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

```
#include <unistd.h>

int rmdir(const char* path);
```

The `rmdir` removes the directory only when they are empty.

3.7.5 chdir and getcwd

A program can navigate directories in the same way as a user moves around the file system.

```
#include <unistd.h>

int chdir(const char *path);
```

And can determine the current working directory

```
#include <unistd.h>
```

```
char *getcwd(char *buf, size_t size);
```

The `getcwd` function writes the name of the current directory into the given buffer, `buf`.

It returns NULL if the directory name would exceed the size of the buffer, given as the parameter `size`.