【Linux Programming】Day12(2)

Tags	
Date	@June 4, 2022
Summary	lseek, stat, and dup

【Ch2】Work with Files

3.4 Other System Calls for Managing Files

3.4.1 lseek

The `lseek` system call sets the read/write pointer of a file descriptor.

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

off_t lseek(int fildes, off_t offset, int whence);
```

`whence` can be one of the following:

- `SEEK_SET`: offset is an absolute position
- `SEEK_CUR`: offset is relative to the current position
- `SEEK_END`: offset is relative to the end of the file

`lseek` returns the offset measured in bytes from the beginning of the file that the file pointer is set to, or -1 on failure.

The type `off_t` is an implementation-dependent integer type defined in `sys/types.h`.

3.4.2 fstat, stat, and lstat

The `fstat` system call returns status information about the file associated with an open file descriptor.

The information is written to a structure, `buf`, the address of which is passed as a parameter.

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

int fstat(int filedes, struct stat *buf);
int stat(const char *path, struct stat *buf);
int lstat(const char *path, struct stat *buf);
```

The related functions stat and lstat return status information for a named file. They produce the same results, except when the file is a symbolic link.

`lstat` returns information about the link itself, and `stat` returns information about the file to which the link refers.

The members of the structure, stat, include those in the following table:

stat Member	Description
st_mode	File permissions and file-type information
st_ino	The inode associated with the file
st_dev	The device the file resides on
st_uid	The user identity of the file owner
st_gid	The group identity of the file owner
st_atime	The time of last access
st_ctime	The time of last change to permissions, owner, group, or content
st_mtime	The time of last modification to contents
st_nlink	The number of hard links to the file

The `st_mode` flags have a number of associated macros defined in the header file `sys/stat.h`.

The permission flags are the same as for the open system call. File-type flags include:

- S_IFBLK: Entry is a block special device
- S_IFDIR: Entry is a directory
- S_IFCHR: Entry is a character special device
- S_IFIFO: Entry is a FIFO(named pipe)
- S_IFREG: Entry is a regular file
- S_IFLNK: Entry is a symbolic link

Other mode flags include:

- S_ISUID: Entry has setUID on execution
- S_ISGID: Entry has setGID on execution

Masks to interpret the `st_mode` flags include:

- S_IFMT: File type
- S_IRWXU: User read/write/execute permissions
- S_IRWXG: Group read/write execute permissions
- S_IRWXO: Others' read/write/execute permissions.

There are some macros defined to help with determining file types:

- S_ISBLK: Test for block special file
- S_ISCHR: Test for character special file
- S_ISDIR: Test for directory
- S_ISFIFO: Test for FIFO
- S_ISREG: Test for regular file
- S_ISLINK: Test for symbolic link

For example, to test that a file doesn't represent a directory and has execute permission set for the owner but no other permissions.

```
struct stat statbuf;
stat("filename", &statbuf);

mode_t modes = statbuf.st_mode;
if (!S_ISDIR(modes) && (modes & S_IRWXU) == S_IXUSR)
```

3.4.3 dup and dup2

The `dup` system calls provide a way of duplicating a file descriptor, giving two or more different descriptors access the same file.

These might be used for reading and writing to different locations in the file.

The `dup` system call duplicates a file descriptor returning a new descriptor.

The `dup2` system call effectively copies on file descriptor to another by specifying the descriptor to use for the copy:

```
#include <unistd.h>

int dup(int fildes);
int dup2(int fildes, int fildes2);
```