Семинар 12: файлы в Linux

25 февраля, 2020

Интерфейс Linux для файлов [x2]

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

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

lseek

- С каждым файловым дескриптором внутри ядра ассоциируется `struct file`
- У этой структуры есть поле, отвечающее за текущее положение в файле (`f_pos`)
- За несколькими файловыми дескрипторами может быть одна `struct file`!
- lseek позволяет изменять позицию в файле

lseek: whence

- **SEEK_SET**: `f_pos = offset`
- **SEEK_CUR**: `f_pos += offset`
- **SEEK_END**: `f_pos = file_size + offset`

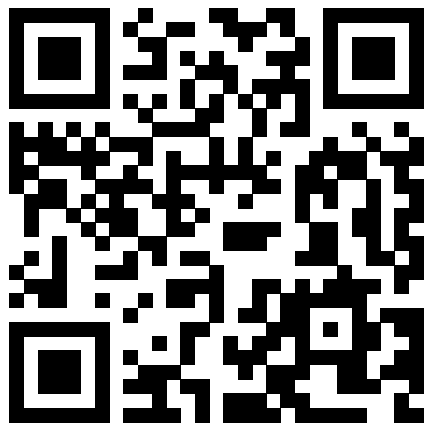
lseek: off_t

- Специальный тип, который предназначен для хранения оффсетов
- Может быть 32ух битным
- Чтобы было 64 бита: `-D_FILE_OFFSET_BITS=64`

Имена файлов

- Относительный путь отсчитывается от текущей директории
- Абсолютный путь — от корня
- В реальности длина ничем не ограничена
- Путь, передаваемый в сисколлы не может превосходить `PATH_MAX`

PATH_MAX Is Tricky



Мета-информация файлов

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

int stat(const char* pathname, struct stat* buf);
int lstat(const char* pathname, struct stat* buf);
int fstat(int fd, struct stat *buf);
ssize_t readlink(const char *pathname, char *buf, size_t bufsiz)
```

struct stat

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

struct stat {
    dev_t     st_dev;     /* ID of device containing file */
    ino_t     st_ino;     /* Inode number */
    mode_t    st_mode;    /* File type and mode */
    nlink_t   st_nlink;   /* Number of hard links */
    uid_t     st_uid;     /* User ID of owner */
    gid_t     st_gid;     /* Group ID of owner */
    dev_t     st_rdev;    /* Device ID (if special file) */
    off_t     st_size;    /* Total size, in bytes */
    blksize_t st_blksize; /* Block size for filesystem I/O */
    blkcnt_t  st_blocks;  /* Number of allocated blocks */
    time_t st_atime;      /* Time of last access */
    time_t st_mtime;      /* Time of last modification */
    time_t st_ctime;      /* Time of last status change */
};
```

Типы файлов

- 7 типов файлов
- Регулярные (**S_ISREG**)
- Директории (**S_ISDIR**)
- Символьные устройства (**S_ISCHR**)
- Блочные устройства (**S_ISBLK**)
- Именованные пайпы (**S_ISFIFO**)
- Символические ссылки (**S_ISLNK**)
- Сокеты (**S_ISSOCK**)

Права доступа

- 3 группы по 3 бита определяют права на чтение (r), запись (w) и исполнение (x)
- Чаще всего записываются в виде `rwxrwxrwx` или восьмеричных чисел
- Примеры:
- `777` = `rwxrwxrwx` — права доступа кому угодно
- `666` = `rw-rw-rw-` — права доступа на чтение запись кому угодно
- `700` = `rw-------` — читать/писать может только владелец
- Сначала проверяются права на владельца, потом на группу и потом для остальных
- Если у владельца `---`, но у группы `rwx`, то у владельца доступа всё равно не будет, даже если он будет входить в группу-владельца

inode и файловые ссылки

- inode — структура, которая содержит в себе мета-информацию (аттрибуты) и аллоцированные блоки
- Также под inode часто понимают её уникальный номер

inode и файловые ссылки

- inode — структура, которая содержит в себе мета-информацию (аттрибуты) и аллоцированные блоки
- Также под inode часто понимают её уникальный номер
- Символические ссылки — файлы, которые содержат в себе пути до других файлов
- Жёсткие ссылки — различные «имена» одного и того же файла
- Жёсткие ссылки на один и тот же файл имеют одинаковые inode в отличие от символических
- Жёсткие ссылки должны быть на одной файловой системе
- Символические ссылки могут ссылаться на несуществующий файл
- `lstat` не следует по символическим ссылкам в отличие от `stat`

Проверка прав доступа

```c
#include <unistd.h>

#define F_OK ...
#define R_OK ...
#define W_OK ...
#define X_OK ...

int access(const char *pathname, int mode);
```

File allocations

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

int truncate(const char* path, off_t length);
int ftruncate(int fd, off_t length);
int fallocate(int fd, int mode, off_t offset, off_t len);
```

ftruncate

- Позволяет изменять *размер* файла, не используя `write`
- Если `file_size < length`, то добивает файл `'\0'`
- Если `file_size > length`, то данные теряются
- Не меняет `f_pos`

fallocate

- Аллоцирует *дисковое пространство*
- После этого вызова гарантируется, что для $[offset; offset + len)$ будет выделено реальное место на диске

Создание и удаление файлов

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

int link(const char *oldpath, const char *newpath);
int symlink(const char *oldpath, const char *newpath);
int mkdir(const char *pathname, mode_t mode);

int rmdir(const char *pathname);
int unlink(const char *oldpath, const char *newpath);
int rename(const char* oldpath, const char* newpath);
```

Работа с директориями

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

DIR *opendir(const char *pathname);

struct dirent {
    ino_t d_ino;
    char d_name[NAME_MAX + 1];
}
struct dirent *readdir(DIR *dp);

void rewinddir(DIR *dp);
int closedir(DIR *dp);
long telldir(DIR *dp);
void seekdir(DIR *dp, long loc);
```

Работа с рабочей директорией

```c
#include <unistd.h>

char *getcwd(char *buf, size_t size);
int chdir(const char *pathname);
int fchdir(int filedes);
```

Работа со временем

Gratias ago!