Лекция 3
Файловая система

Файловая система

- Формат хранения данных на носителе (ф.с. FAT32, NTFS, EXT4)
  - Каждая ф.с. имеет свои особенности: максимальный размер, ограничения на размеры файлов, длину имени и т. п.
- Компонент ядра ОС, отвечающий за доступ к файлам
  - API для работы с файлами и файловой системой
  - Драйвера конкретных файловых систем

Требования к файловым системам

- Постоянство (сохранение данных после окончания процесса и после остановки ОС)
- Поддержка данных огромного размера (одного файла и суммарного)
- Эффективность (скорость поиска файла и обращения к файлу)
- Поддержка разделения прав доступа и квотирования
- Устойчивость к программному сбою
- Устойчивость к аппаратному сбою

Задачи ядра ОС

- Предоставление стандартного интерфейса
  - POSIX API для работы с файлами и файловой системой: user space ↔ kernel space
  - VFS (virtual file system): kernel ↔ FS driver
- Управление ресурсами
  - Разграничение прав доступа к объектам ФС
  - Квотирование ресурсов ФС
  - Арбитраж параллельного доступа к ФС (блокировки, атомарность)

Основные абстракции

- Файл — именованный набор данных (для Unix-систем — регулярный файл)
- Или файл — запись в каталоге. В Unix-системах:
  - Регулярные файлы
  - Файлы-каталоги
  - Файлы-устройства (блочные и символьные)
  - Символические ссылки
  - Именованные каналы (FIFO)
  - Сокеты

Регулярные файлы

- Коллекция записей (record-oriented FS) в MVS (System 370) или VAX VMS
  - Могут быть постоянной или переменной длины
  - Ядро берет на себя функции по поддержанию формата и обеспечению корректности
- Поток байт (stream)
  - Структурирование и поддержание корректности возлагается на user space
  - Но! Когда необходимо ядро рассматривает файл как коллекцию записей (загрузка в память исполняемого образа)

Регулярные файлы

- Гибрид stream и record-oriented: MacOS HFS
  - Файл состоит из data fork и resource fork
  - Data fork — поток байт
  - Resource fork — структурированные данные, ресурсы, метаданные
- Недостаток: проблемы с совместимостью (например, на FAT32 для каждого файла NAME создается спец. файл ._NAME)

Регулярные файлы

- Многопоточные файлы (alternate data stream — Win NTFS)
  - Файл может состоять из нескольких потоков
  - CreateFile("input.txt:stream1")
  - Недостаток: проблемы с совместимостью
- Стандартная модель: регулярный файл — одиночный поток байт

Атрибуты файла

- Тип файла
- Размер
- Флаги
- Права доступа
- Информация о времени

Основные абстракции: каталог

- Каталог — коллекция файлов (в широком смысле)
  - Если каталог может содержать другие каталоги — иерархическая файловая система
  - Иерархическая файловая система — дерево каталогов, корень дерева — корневой каталог
  - Каждая запись в каталоге имеет уникальное имя
  - Каждая запись в каталоге имеет уникальный абсолютный (полный) путь от корневого каталога

Ограничения на имя файла

- Чувствительность к регистру (case sensitive)
- Сохранение регистра (case preservation)
- Разрешенные кодировки (charset)
- Запрещенные символы (reserved chars)
- Максимальная длина (max length)
- Запрещенные имена (reserved names)

Ограничения на имя файла

	Современный UNIX	VFAT	NTFS
Case Sensitive	**YES**	**NO**	**NO**
Case Preserve	**YES**	**YES**	**YES**
Charset	**Any 8-bit (UTF-8)**	**UCS-2**	**UTF-16**
Reserved chars	**\0 /**	**0x00-0x1F 0x7F " * / : < > ? \ |**	**0x0000**
Max Length	**255**	**255**	**255**
Reserved names			**$MFT и другие (только в корне)**

Пути POSIX

- Элементы пути разделяются /
- Несколько / подряд сливаются в один
- "/" - корневой каталог
  - Путь начинается с / - абсолютный
  - Путь начинается с другого символа — относительный
- Если путь оканчивается / - должен быть каталог
- Максимальная длина пути - PATH_MAX

Примеры

/etc/hosts	абсолютный путь
.ssh/authorized_keys	относительно homedir
/	корневой каталог
///////	то же самое
/etc/////hosts	то же, что и первое
/etc///	то же, что /etc
/etc/hosts/	неверно, не каталог
.	текущий каталог
..	родительский каталог

POSIX контроль доступа

- Определяет возможность выполнения операций со стороны процесса над ресурсом
- Пользователь (user) — основная идентификация, user id (uid) — целое число
- Uid == 0 — суперпользователь
- /etc/passwd — файл с отображением uid в user name (login)
- Каждый процесс работает с правами некоторого пользователя (обычно того, кто запустил) — getuid
- Каждый файл имеет владельца (обычно тот, кто создал)

POSIX контроль доступа

- Группа — вторичная идентификация, group id (gid) — целое число
- В группе может находиться несколько пользователей
- Пользователь может принадлежать нескольким группам, но у процесса в любой момент времени есть основная группа (primary group)
- Процесс может свободно переключаться между своими группами (setgid) или менять группу файла между любыми своими, если он — владелец
- Каждый процесс принадлежит множеству групп
- Каждый файл имеет одну группу

Избирательное управление доступом (discretionary access control)

- Модель владелец-группа-прочие
- Если uid процесса и uid файла совпадают, берется множество прав доступа владельца (user)
- Если один из gid процесса совпадает с gid файла, берется множество прав доступа группы (group)
- Иначе берется множество прав доступа прочих (other)

Множество прав доступа

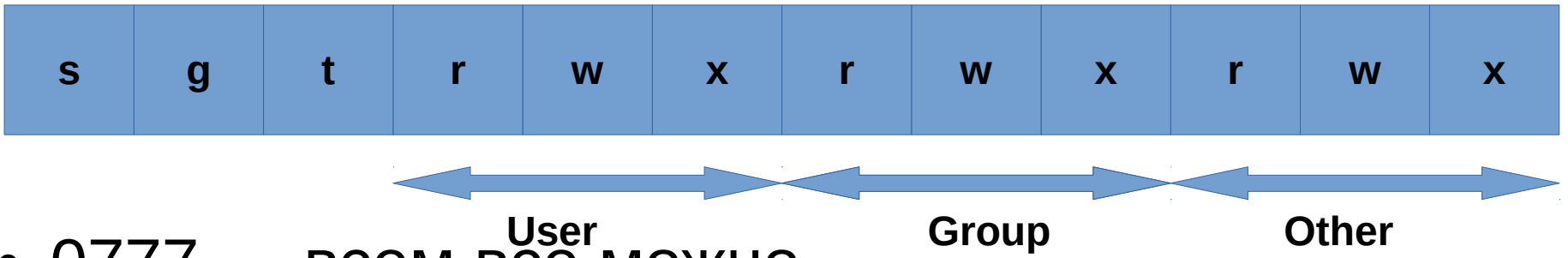
- r,w,x — интерпретация зависит от того, является ли файл каталогом или нет
- Права для файлов:
  - "r" — право на чтение из файла (вызов системного вызова read или lseek)
  - "w" — право на запись в файл (вызов write)
  - "x" — право на выполнение файла (вызов exec*)

Права доступа

- Права для каталогов
  - "r" — право читать список файлов в каталоге (вызовы opendir/readdir/…)
  - "w" — право модифицировать список файлов в каталоге (создавать, удалять, переименовывать)
  - "x" — право на поиск заданного имени в каталоге
- Права "--x" — пользователь не может посмотреть какие файлы есть в каталоге, но если он знает имя файла в нем, с этим файлом может работать

Права доступа

- Полные права — 12 бит (9 основных + 3 доп.)



- 0777 — всем все можно
- 0664 — чтение/запись для владельца и группы, только чтение для остальных
- 0700 — все права только для владельца

Дополнительные биты

Бит	Для файлов	Для каталогов
S (04000)	При выполнении процесс, запущенный из данного файла, может изменить свой uid на uid файла	Не используется
G (02000)	При выполнении процесс, запущенный из данного файла, может изменить свой gid на gid файла	При создании новых файлов и каталогов группа наследуется из родительского каталога, а не из процесса
T (01000)	Не используется	Только владелец может удалить созданный им файл

Время Unix

- Календартное время: <time.h>, тип time_t
- Число секунд от «сотворения мира Unix» (1 января 1970 г. UTC)
- Упрощенная модель — в сутках всегда 86400 секунд
- Если time_t — знаковый 32-битный, то «конец света Unix» случится 19 января 2038 г.
- Если time_t — знаковый 64-битный, то «конец света Unix» случится 4 декабря 292,277,026,596

Файловый дескриптор

- Файловый дескриптор — идентификатор открытого файла в процессе
- Каждый процесс имеет свой набор файловых дескрипторов, независимый от других процессов
- Неотрицательное целое число
- Обычно при старте процесса:
  - 0 — stdin
  - 1 — stdout
  - 2 — stderr
- При выделении нового ф. д. всегда выбирается свободный с минимальным номером
- ф. д. - индекс в таблицу открытых файлов процесса

Открытие файла

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

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

- Возвращает файловый дескриптор при успехе и -1 при ошибке
- При ошибке код ошибки в errno (<errno.h>)
- path — путь к открываемому файлу

Флаги открытия

- Основные режимы открытия:
  - O_RDONLY — только чтение
  - O_WRONLY — только запись
  - O_RDWR — чтение-запись
- Флаги управления файловым дескриптором
  - O_CLOEXEC — файловый дескриптор закрывается автоматически при exec
- Модификаторы режима записи
  - O_APPEND — режим добавления в конец файла
  - O_TRUNC — очистить файл

Флаги открытия

- Модификаторы создания файла
  - O_CREAT — создание файла
  - O_EXCL — создание файла только в случае, если он еще не существует
- Типичные комбинации флагов
  - O_RDONLY
  - O_WRONLY | O_CREAT | O_TRUNC
  - O_WRONLY | O_CREAT | O_APPEND

Режим создания файла

- При указании флага O_CREAT используется параметр mode — права доступа на создаваемый файл
- Права доступа накладываются на параметр umask: mode & ~umask
- Например, mode == 0666, umask == 0022, права на создаваемый файл 0644
- Mode == 0700, umask == 0007, права 0700

Закрытие файла

int close(int fd);

- При успехе возвращается 0, при неудаче - -1.
- Причины неудачи:
  - EBADF — неправильный файловый дескриптор
  - EINTR — операция была прервана
  - EIO — ошибка записи
- В любом случае, ничего разумного при ошибке сделать нельзя!