2.1. Файлы.

Требования к хранению информации:

- возможность хранения больших объемов данных
- информация должна сохраняться после прекращения работы процесса
- несколько процессов должны иметь одновременный доступ к информации

2.1.1. Именование файлов.

Длина имени файла зависит от ОС, может быть от 8 (MS-DOS) до 255 (Windows, LINUX) символов.

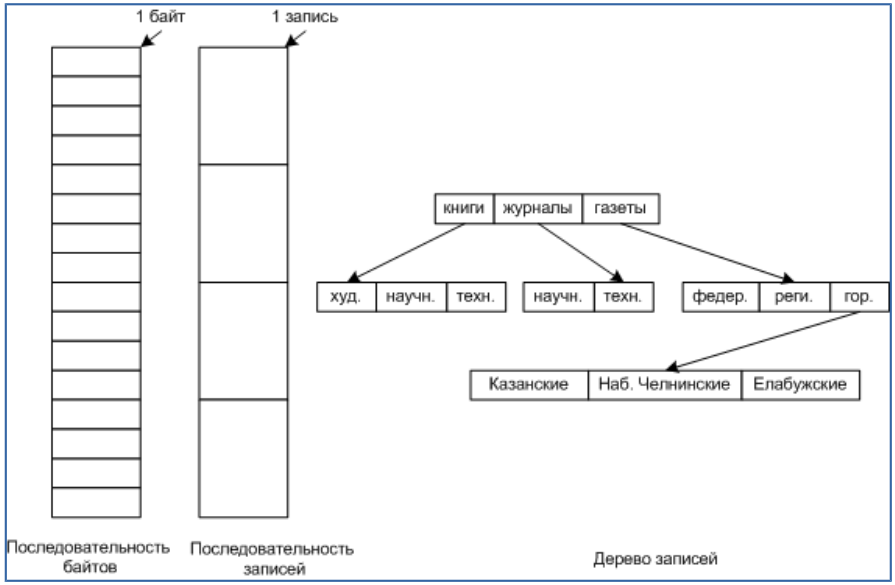
ОС могут различать прописные и строчные символы. Например, WINDOWS и windows для MS-DOS одно и тоже, но для UNIX это разные файлы.

Во многих ОС имя файла состоит из двух частей, разделенных точкой, например windows.exe. Часть после точки называют **расширением файла**. По нему система различает тип файла.

У MS-DOS расширение составляет 3 символа. По нему система различает тип файла, а также можно его исполнять или нет.

У UNIX расширение ограничено размером имени файла в 255 символов, также у UNIX может быть несколько расширений, но расширениями пользуются прикладные программы, а не ОС. По расширению UNIX не может определить исполняемый это файл или нет. Тип файла UNIX определяет по MagicNumber — последовательности символов в начале файла.

2.1.2. Структура файла.



Три основных типа структур файла.

1. **Последовательность байтов** - ОС не интересуется содержимым файла, она видит только байты. Основное преимущество такой системы, это гибкость использования. Используются в Windows и UNIX.
2. **Последовательность записей** - записей фиксированной длины (например, перфокарта), считываются последовательно. Сейчас в ФС не используются. Развитием подхода стали базы данных.
3. **Дерево записей** - каждая запись имеет ключ, записи считываются по ключу. Основное преимущество такой системы, это скорость поиска. Пока еще используется на мэйнфреймах.

2.1.3. Типы файлов.

Основные типы файлов:

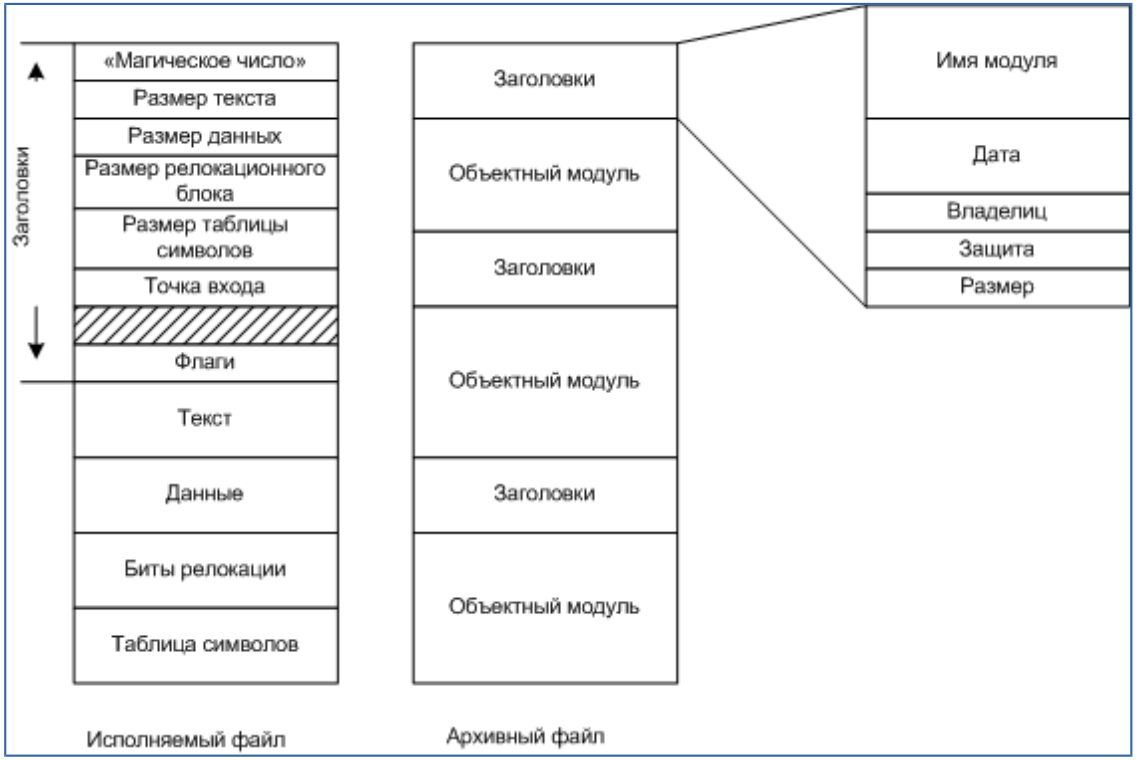
- **Регулярные** - содержат информацию пользователя. Используются в Windows и UNIX.
- **Каталоги** - системные файлы, обеспечивающие поддержку структуры файловой системы. Используются в Windows и UNIX.
- **Символьные** - для моделирования ввода-вывода. Используются только в UNIX.
- **Блочные** - для моделирования дисков. Используются только в UNIX.

Основные типы регулярных файлов:

- **ASCII файлы** - состоят из текстовых строк. Каждая строка завершается возвратом каретки (Windows ASCII-13), символом перевода строки (UNIX ASCII-10) и используются оба варианта (MS-DOS). Поэтому если открыть текстовый файл, написанный в UNIX, в Windows, то все строки сольются в одну большую строку, но под MS-DOS они не сольются (*это достаточно частая ситуация*). Основные преимущества ASCII файлов:
  - могут отображаться на экране, и выводится на принтер без преобразований
  - могут редактироваться почти любым редактором
- **Двоичные файлы** - остальные файлы (не ASCII). Как правило, имеют внутреннею структуру.

Основные типы двоичных файлов:

- **Исполняемые** - программы, их может обрабатывать сама операционная система, хотя они записаны в виде последовательности байт.
- **Неисполняемые** - все остальные.



Примеры исполняемого и не исполняемого файла

«Магическое число» - идентифицирующее файл как исполняющий.

2.1.4. Доступ к файлам.

Основные виды доступа к файлам:

- **Последовательный** - байты читаются по порядку. Использовались, когда были магнитные ленты.
- **Произвольный** - файл можно читать с произвольной точки. Основное преимущество возникает, когда используются большие файлы (например, баз данных) и надо считать только часть данных из файла. Все современные ОС используют этот доступ.

2.1.5. Атрибуты файла.

Основные атрибуты файла:

- Защита - кто, и каким образом может получить доступ к файлу (пользователи, группы, чтение/запись). Используются в Windows и UNIX.
- Пароль - пароль к файлу
- Создатель - кто создал файл
- Владелец - текущий владелец файла
- Флаг "только чтение" - 0 - для чтения/записи, 1 - только для чтения. Используются в Windows.
- Флаг "скрытый" - 0 - виден, 1 - невиден в перечне файлов каталога (по умолчанию). Используются в Windows.
- Флаг "системный" - 0 - нормальный, 1 - системный. Используются в Windows.
- Флаг "архивный" - готов или нет для архивации (не путать сжатием). Используются в Windows.
- Флаг "сжатый" - файл сжимается (подобие zip архивов). Используются в Windows.

- Флаг "шифрованный" - используется алгоритм шифрования. Если кто-то попытается прочесть файл, не имеющий на это прав, он не сможет его прочесть. Используются в Windows.
- Флаг ASCII/двоичный - 0 - ASCII, 1 - двоичный
- Флаг произвольного доступа - 0 - только последовательный, 1 - произвольный доступ
- Флаг "временный" - 0 - нормальный, 1 - для удаления файла по окончании работы процесса
- Флаг блокировки - блокировка доступа к файлу. Если он занят для редактирования.
- Время создания - дата и время создания. Используются UNIX.
- Время последнего доступа - дата и время последнего доступа
- Время последнего изменения - дата и время последнего изменения. Используются в Windows и UNIX.
- Текущий размер - размер файла. Используются в Windows и UNIX.

2.1.6. Операции с файлами.

Основные системные вызовы для работы с файлами:

- Create - создание файла без данных.
- Delete - удаление файла.
- Open - открытие файла.
- Close - закрытие файла.
- Read - чтение из файла, с текущей позиции файла.

- Write - запись в файл, в текущею позицию файла.
- Append - добавление в конец файла.
- Seek - устанавливает файловый указатель в определенную позицию в файле.
- Get attributes - получение атрибутов файла.
- Set attributes - установить атрибутов файла.
- Rename - переименование файла.

2.1.7. Файлы, отображаемые на адресное пространство памяти.

Иногда удобно файл отобразить в памяти (не надо использовать системные вызовы ввода-вывода для работы с файлом), и работать с памятью, а потом записать измененный файл на диск.

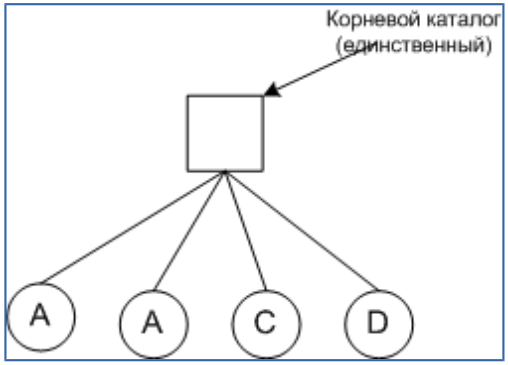
При использовании страничной организации памяти, файл целиком не загружается, а загружаются только необходимые страницы.

При использовании сегментной организации памяти, файл загружают в отдельный сегмент.

2.2. Каталоги.

2.2.1. Одноуровневые каталоговые системы.

В этой системе все файлы содержатся в одном каталоге.



Однокаталоговая система, содержащая четыре файла, файлов A два, но разных владельцев

Преимущества системы:

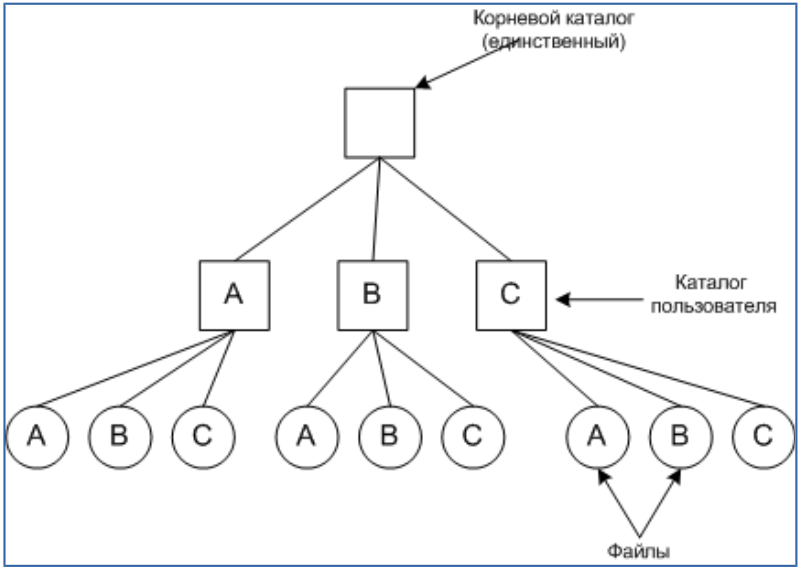
- Простота
- Возможность быстро найти файл, не надо лазить по каталогам

Недостатки системы:

- Различные пользователи могут создать файлы с одинаковыми именами.

2.2.2. Двухуровневые каталоговые системы.

Для каждого пользователя создается свой собственный каталог.



Двухуровневая каталоговая система

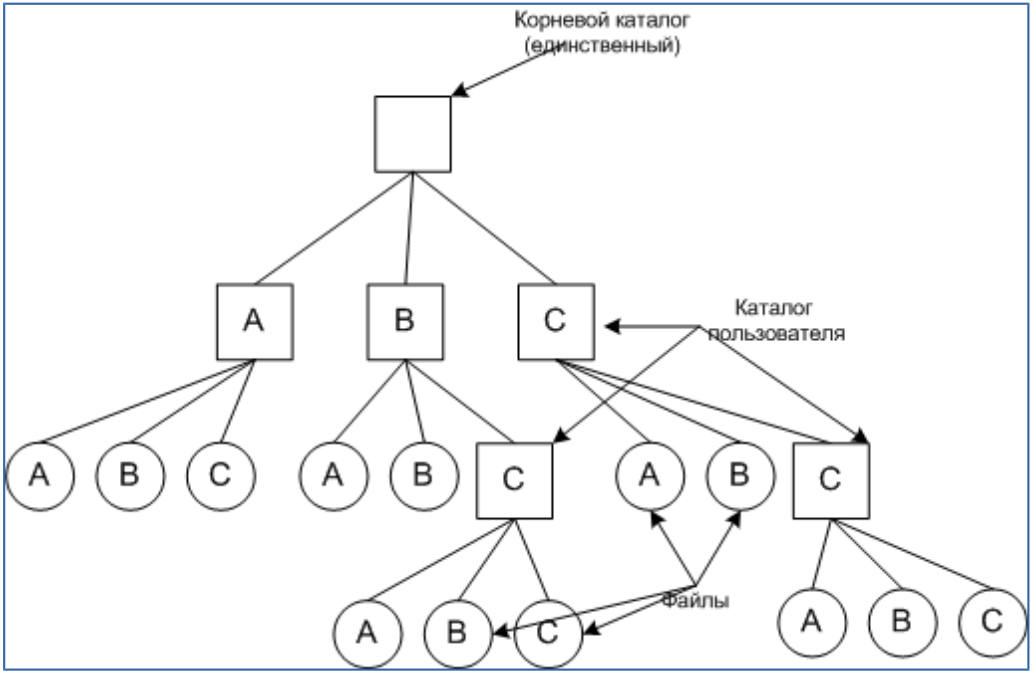
Пользователь, при входе в систему, попадает в свой каталог и работает только с ним. Это делает проблематичным использование системных файлов.

Эту проблему можно решить созданием системного каталога, с общим доступом.

Если у одного пользователя много файлов, то у него тоже может возникнуть необходимость в файлах с одинаковыми именами.

2.2.3. Иерархические каталоговые системы.

Каждый пользователь может создавать столько каталогов, сколько ему нужно.



Иерархическая каталоговая система

Почти все современные универсальные ОС, организованы таким образом. Специализированным ОС это может быть не нужным.

В UNIX также используется **сетевая иерархия** файлов и каталогов за счёт технологии hardlink.

2.2.4. Имя пути.

Для организации дерева каталогов нужен некоторый способ указания файла. Два основных метода указания файла:

- **абсолютное имя пути** - указывает путь от корневого каталога, например:
  - для Windows \usr\ast\mailbox
  - для UNIX /usr/ast/mailbox
  - для MULTICS >usr>ast>mailbox
- **относительное имя пути** - путь указывается от текущего каталога (рабочего каталога), например:
  - если текущий каталог /usr/, то абсолютный путь /usr/ast/mailbox перепишется в ast/mailbox
  - если текущий каталог /usr/ast/, то абсолютный путь /usr/ast/mailbox перепишется в mailbox
  - если текущий каталог /var/log/, то абсолютный путь /usr/ast/mailbox перепишется в ../../usr/ast/mailbox

./ - означает текущий каталог
../ - означает родительский каталог

2.2.5. Операции с каталогами.

Основные системные вызовы для работы с каталогами:

- Create - создать каталог
- Delete - удалить каталог
- OpenDir - закрыть каталог
- CloseDir - закрыть каталог
- ReadDir - прочитать следующий элемент открытого каталога
- Rename - переименование каталога
- Link - создание жесткой ссылки, позволяет файлу присутствовать сразу в нескольких каталогах.
- Unlink - удаление ссылки из каталога