ПОНЯТИЕ ФАЙЛА: ИМЕНА, СТРУКТУРА, АТРИБУТЫ, ОПЕРАЦИИ

Файл является механизмом абстрагирования. Он предоставляет способ сохранения информации на диске и последующего ее считывания, который должен оградить пользователя от подробностей о способе и месте хранения информации и деталей фактической работы дисковых устройств.

Имена) Когда процесс создает файл, он присваивает ему имя. Когда процесс завершается, файл продолжает существовать, и к нему по этому имени могут обращаться другие процессы.

Конкретные правила составления имен файлов варьируются от системы к системе, но все ныне существующие операционные системы в качестве допустимых имен файлов позволяют использовать от одной до восьми букв.

Многие операционные системы поддерживают имена файлов, состоящие из двух частей, разделенных точкой, как, например, prog.c. Та часть имени, которая следует за точкой, называется **расширением имени файла** и, как правило, несет в себе некото- рую информацию о файле.

|  |  |
| --- | --- |
| **Расширение** | **Значение** |
| .bak | Резервная копия файла |
| .c | Исходный текст программы на языке C |
| .gif | Изображение формата GIF |
| .hlp | Файл справки |
| .html | Документ в формате HTML |
| .jpg | Статическое растровое изображение в формате JPEG |
| .mp3 | Музыка в аудиоформате MPEG layer 3 |
| .mpg | Фильм в формате MPEG |
| .o | Объектный файл (полученный на выходе компилятора, но еще не прошедший компоновку) |
| .pdf | Документ формата PDF |
| .ps | Документ формата PostScript |
| .tex | Входной файл для программы форматирования TEX |
| .txt | Обычный текстовый файл |
| .zip | Архив, сжатый программой zip |

Структура)

Файлы могут быть структурированы несколькими различными способами.

Есть три наиболее вероятные структуры

1. представляет со- бой бессистемную последовательность байтов
2. В данной модели файл представляет собой последовательность записей фиксированной длины, каждая из которых имеет собственную внутреннюю структуру.
3. При такой организации файл состоит из дерева записей, необязательно одинаковой длины, каждая из которых в конкретной позиции содержит **ключевое** поле. Дерево сортируется по ключевому полю, позволяя выполнять ускоренный поиск по конкретному ключу.

АТРИБУТЫ)

У каждого файла есть свои имя и данные. Вдобавок к этому все операционные си- стемы связывают с каждым файлом и другую информацию, к примеру дату и время последней модификации файла и его размер. Мы будем называть эти дополнительные сведения **атрибутами файла**. Также их называют **метаданными**.

|  |  |
| --- | --- |
| **Атрибут** | **Значение** |
| Защита | Кто и каким образом может получить доступ к файлу |
| Пароль | Пароль для получения доступа к файлу |
| Создатель | Идентификатор создателя файла |
| Владелец | Текущий владелец |
| Флаг «только для чтения» | 0 — для чтения и записи; 1 — только для чтения |
| Флаг «скрытый» | 0 — обычный; 1 — не предназначенный для отображения в перечне файлов |
| Флаг «системный» | 0 — обычный; 1 — системный |
| Флаг «архивный» | 0 — прошедший резервное копирование; 1 — нуждающийся в резервном копировании |
| Флаг «ASCII/двоичный» | 0 — ASCII; 1 — двоичный |
| Флаг произвольного доступа | 0 — только последовательный доступ; 1 — произвольный до- ступ |
| Флаг «временный» | 0 — обычный; 1 — удаляемый по окончании работы процесса |
| Флаги блокировки | 0 — незаблокированный; ненулевое значение — заблокиро- ванный |
| Длина записи | Количество байтов в записи |
| Позиция ключа | Смещение ключа внутри каждой записи |
| Длина ключа | Количество байтов в поле ключа |
| Время создания | Дата и время создания файла |
| Время последнего доступа | Дата и время последнего доступа к файлу |
| Время внесения последних изменений | Дата и время внесения в файл последних изменений |
| Текущий размер | Количество байтов в файле |
| Максимальный размер | Количество байтов, до которого файл может увеличиваться |

ОПЕРАЦИИ)

Файлы предназначены для хранения информации с возможностью ее последующего извлечения. Разные системы предоставляют различные операции, позволяющие со-

4.1. Файлы **311** хранять и извлекать информацию. Далее рассматриваются наиболее распространенные

системные вызовы, относящиеся к работе с файлами.

* ⎝  *Create* (Создать). Создает файл без данных. Цель вызова состоит в объявлении о появлении нового файла и установке ряда атрибутов.
* ⎝  *Delete* (Удалить). Когда файл больше не нужен, его нужно удалить, чтобы освобо- дить дисковое пространство. Именно для этого и предназначен этот системный вызов.
* ⎝  *Open* (Открыть). Перед использованием файла процесс должен его открыть. Цель системного вызова *open* — дать возможность системе извлечь и поместить в опера- тивную память атрибуты и перечень адресов на диске, чтобы ускорить доступ к ним при последующих вызовах.
* ⎝  *Close* (Закрыть). После завершения всех обращений к файлу потребность в его атрибутах и адресах на диске уже отпадает, поэтому файл должен быть закрыт, чтобы освободить место во внутренней таблице. Многие системы устанавливают максимальное количество открытых процессами файлов, определяя смысл суще- ствования этого вызова. Информация на диск пишется блоками, и закрытие файла вынуждает к записи последнего блока файла, даже если этот блок и не заполнен.
* ⎝  *Read* (Произвести чтение). Считывание данных из файла. Как правило, байты по- ступают с текущей позиции. Вызывающий процесс должен указать объем необхо- димых данных и предоставить буфер для их размещения.
* ⎝  *Write* (Произвести запись). Запись данных в файл, как правило, с текущей позиции. Если эта позиция находится в конце файла, то его размер увеличивается. Если теку- щая позиция находится где-то в середине файла, то новые данные пишутся поверх существующих, которые утрачиваются навсегда.
* ⎝  *Append* (Добавить). Этот вызов является усеченной формой системного вызова *write*. Он может лишь добавить данные в конец файла. Как правило, у систем, предо- ставляющих минимальный набор системных вызовов, вызов *append* отсутствует, но многие системы предоставляют множество способов получения того же результата, и иногда в этих системах присутствует вызов *append*.
* ⎝  *Seek* (Найти). При работе с файлами произвольного доступа нужен способ указания места, с которого берутся данные. Одним из общепринятых подходов является при- менение системного вызова *seek*, который перемещает указатель файла к определен- ной позиции в файле. После завершения этого вызова данные могут считываться или записываться с этой позиции.
* ⎝  *Get attributes* (Получить атрибуты). Процессу для работы зачастую необходимо считать атрибуты файла. К примеру, имеющаяся в UNIX программа make обычно используется для управления проектами разработки программного обеспечения, состоящими из множества сходных файлов. При вызове программа make проверяет время внесения последних изменений всех исходных и объектных файлов и для обновления проекта обходится компиляцией лишь минимально необходимого ко- личества файлов. Для этого ей необходимо просмотреть атрибуты файлов, а именно время внесения последних изменений.
* ⎝  *Set attributes* (Установить атрибуты). Значения некоторых атрибутов могут уста- навливаться пользователем и изменяться после того, как файл был создан. Такую возможность дает именно этот системный вызов. Характерным примером может   
  послужить информация о режиме защиты. Под эту же категорию подпадает большинство флагов.

⎝ *Rename* (Переименовать). Нередко пользователю требуется изменить имя существу- ющего файла. Этот системный вызов помогает решить эту задачу. Необходимость в нем возникает не всегда, поскольку файл может быть просто скопирован в новый файл с новым именем, а старый файл затем может быть удален.