Файловые системы. Часть 2

Лекция

Ревизия: 0.1

Последняя модификация: Vlad Kovtun

Дата последней модификации: 03.04.2015 14:44:00

История изменений

07.08.2014 – Версия 0.1. Первичный документ. Ковтун В.Ю.

Содержание

Лекция 7. Файловые системы

Вопросы

1. Строение ФС: CD-ROM.
2. Строение ФС: MS-DOS.
3. Строение ФС: Unix v7.
4. Строение ФС: XFS.
5. Строение ФС: EXT4.

Файловые системы компакт дисков

В качестве первого примера ФС рассмотрим системы, которые используются на компакт-дисках. Это очень простые системы, поскольку они были разработаны для носителей, предназначенных только для чтения данных.

В них не отслеживаются свободные блоки, поскольку файлы на компакт-дисках не могут освобождаться или добавляться после того, как диск был произведён. Далее мы рассмотрим основной тип файловой системы компакт- дисков и два расширения этого типа.

Через несколько лет после появления первого компакт-диска был представлен записываемый компакт-диск — CD-R (**CD Recordable**). В отличие от простого компакт-диска, к нему можно было добавлять файлы после первой записи, но они просто добавлялись к концу записываемого компакт-диска. Файлы никогда не удалялись.

Файловая система ISO 9660

Наиболее распространенный международный стандарт ISO 9660 для файловых систем компакт-дисков был принят в 1988 году. Одна из целей этого стандарта — сделать каждый компакт-диск читаемым на любом компьютере, независимо от использующегося порядка следования байтов и независимо от используемой ОС. Это привело к тому, что на ФС были наложены некоторые ограничения, чтобы она могла читаться в среде слабых ОС, использовавшихся в то время (например, в MS-DOS).

У компакт-диска отсутствуют концентрические цилиндры, имеющиеся у магнитных дисков. Вместо них используется одна протяженная спираль с записью битов в линейной последовательности. Биты на протяжении этой спирали разбиты на логические блоки (которые также называют логическими секторами) по 2352 байта.

Некоторые из них предназначены для преамбул, коррекции ошибок и других служебных данных. Полезная часть каждого логического блока занимает 2048 байт. Когда компакт-диск используется для музыкальных записей, на нем имеются начальные, конечные и промежуточные пустые места, которые не используются для компакт-дисков с данными. Зачастую позиция блока на спирали приводится в минутах и секундах. Она может быть преобразована в номер линейного блока, используя соотношение 1 с = 75 блоков.

Стандарт ISO 9660 поддерживает также наборы компакт дисков до $2^{16}-1$ компакт-диска в наборе. Отдельный компакт-диск также может быть разбит на несколько логических томов (разделов). Но далее мы сконцентрируемся на стандарте ISO 9660 для одного не разбитого на разделы компакт-диска.

Каждый компакт-диск начинается с 16 блоков, чья функция не определена стандартом ISO 9660. Производитель компакт-диска может использовать эту область для программы самозагрузки, позволяющей компьютерам запускаться с компакт-диска, или в каких-нибудь других целях. Далее следует один блок, содержащий основной описатель тома, в котором хранится некоторая общая информация о компакт-диске. В нее включен идентификатор системы (32 байта), идентификатор тома (32 байта), идентификатор издателя (128 байт) и идентификатор того, кто подготовил данные (128 байт). Производитель диска может заполнить эти поля по своему усмотрению, за исключением того, что в них для обеспечения совместимости с различными платформами должны быть буквы в верхнем регистре, цифры и весьма ограниченное количество знаков препинания.

Основной описатель тома также содержит имена трех файлов, в которых могут храниться краткий обзор, уведомление об авторских правах и библиографическая

информация соответственно. Кроме того, в этом блоке также содержатся определенные ключевые числа, включающие размер логического блока (как правило, 2048, однако в определенных случаях могут использоваться более крупные блоки, размер которых равен степеням числа два, например 4096,8192 и т.д.), количество блоков на компакт-диске, а также дата создания и дата окончания срока службы диска. И наконец, основной описатель тома также содержит описатель корневого каталога, что позволяет найти этот каталог на компакт-диске (то есть определить номер блока, содержащего начало каталога). Из этого каталога можно получить местоположение всех остальных элементов файловой системы.

Помимо основного описателя тома, компакт-диск может содержать дополнительный описатель тома. В нем хранится информация, подобная той, что хранится в основном описателе.

Что касается корневого и всех остальных каталогов, то они состоят из переменного количества записей, последняя из которых содержит бит, который помечает ее последней. Сами по себе записи каталогов также имеют переменную длину. Каждая запись каталога состоит из 10-12 полей, некоторые из них имеют ASCII-формат, а остальные имеют формат двоичного числа. Двоичные поля кодируются дважды, один раз в формате прямого порядка байтов (используемого, к примеру, на машинах Pentium), а второй — в формате обратного порядка байтов (используемого, к примеру, на машинах SPARC). Таким образом, 16-разрядное число использует 4 байта, а 32-разрядное — 8 байт.

Использование подобного избыточного кодирования было обусловлено стремлением не ущемить при разработке стандарта ничьих интересов. Если бы стандарт навязывал прямой порядок байтов, то представители компаний, в чьей продукции использовался обратный порядок байтов, почувствовали бы себя гражданами второго сорта и не приняли бы этот стандарт. Таким образом, эмоциональная составляющая компакт-дисков может быть подсчитана и измерена в килобайтах потерянного пространства.

Формат записи каталога стандарта ISO 9660 показан на Рис. 1. Поскольку запись каталога имеет переменную длину, первое поле представлено байтом, сообщающем о длине записи. Чтобы избежать любой неопределенности, установлено, что в этом байте старший бит размещается слева.

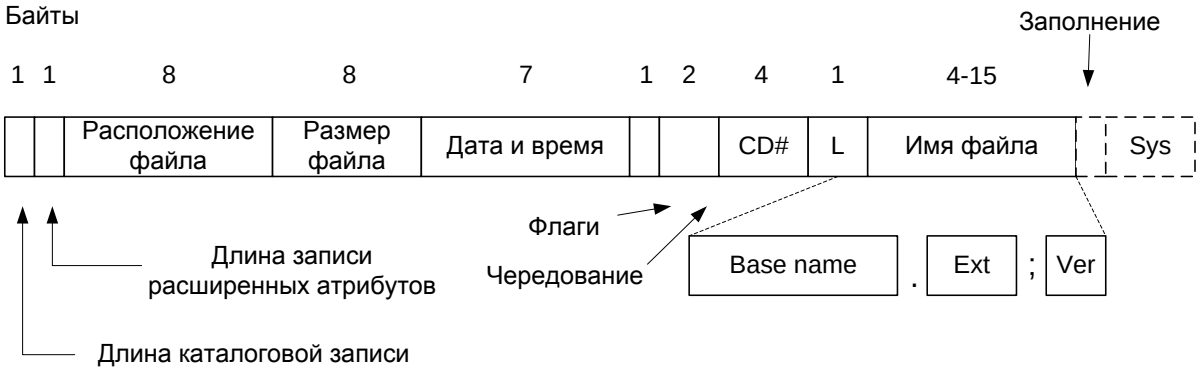

Рис. 1. Запись каталога в стандарте ISO 9660

Записи каталогов могут дополнительно иметь расширенные атрибуты. Если такая возможность используется, то во втором байте указывается длина расширенных атрибутов.

Затем следует номер начального блока самого файла. Файлы хранятся в виде непрерывной череды блоков, поэтому размещение файла полностью определяется начальным блоком и размером, который содержится в следующем поле.

Дата и время записи компакт-диска хранятся в следующем поле в отдельных байтах для года, месяца, дня, часа, минуты, секунды и часового пояса (точнее, дата и время создания файла). Летоисчисление для компакт-дисков начинается с 1900 года, а это значит, что компакт-диски подвержены проблеме 2156 года, поскольку за 2155 годом для них последует год 1900. Если бы отсчет велся с 1988 года (когда был принят стандарт), то проблема была бы отложена до 2244 года, и в запасе было бы еще 88 лет.

Поле флажков содержит несколько битов разного назначения, включая бит скрытия записи в выводах каталога (свойство, позаимствованное у MS-DOS), бит, позволяющий отличить запись, относящуюся к файлу, от записи, относящейся к каталогу, бит, позволяющий использование расширенных атрибутов, и бит, помечающий последнюю запись в каталоге. В этом поле имеются и несколько других битов, но здесь они рассматриваться не будут.

Следующее поле сообщает, на каком компакт-диске расположен файл. Допускается, чтобы запись каталога на одном компакт-диске ссылалась на файл, расположенный на другом компакт-диске набора. Таким образом, появляется возможность создания главного каталога на первом компакт-диске набора, в котором содержится список всех файлов на всех компакт-дисках всего набора. Поле, помеченное на Рис. 1 буквой L, задает размер имени файла в байтах. Сразу за ним следует само имя файла, которое состоит из основного имени, точки, расширения, точки с запятой и двоичного номера версии (1 или 2 байта). В основном имени и расширении могут использоваться буквы в верхнем регистре, цифры от 0 до 9 и символ подчеркивания. Все остальные символы запрещены, чтобы обеспечить возможность обработки любого имени файла на любом компьютере. Основное имя может содержать до восьми символов, а расширение — до трех. Этот выбор был продиктован необходимостью сохранения совместимости с MS-DOS. Имя файла может фигурировать в каталоге несколько раз, если только каждый экземпляр отличается номером версии.

Последние два поля присутствуют не всегда. Поле Дополнение используется для того, чтобы каждая запись каталога составляла четное количество байт, чтобы выровнять числовые поля последующих записей по двухбайтовым границам. Если требуется дополнение, то используется нулевой байт. И наконец, у нас есть поле, используемое системой. Его функции и размер не определены, за исключением того, что в нем должно быть четное количество байт. В разных системах оно используется по-разному. К примеру, в системах Macintosh в нем хранятся флажки Finder.

Записи в каталоге, за исключением первых двух, идут в алфавитном порядке. Первая запись предназначена для самого каталога. А вторая — для его родительского каталога. В этом отношении эти записи аналогичны записям «.» и «..» каталога UNIX. Самих файлов для этих записей быть не должно.

Явного ограничения на количество записей в каталоге не установлено. Но есть ограничение на глубину вложенности каталогов. Максимальная глубина вложенности каталогов равна восьми каталогам. Это ограничение было установлено произвольным образом, чтобы упростить реализацию файловой системы.

Стандартом ISO 9660 определены так называемые три уровня. На **уровне 1** применяются самые жесткие ограничения и определяется, что имена файлов ограничиваются уже рассмотренной схемой $8+3$ символа, а также требуется, чтобы все файлы были непрерывными согласно ранее данному описанию. Кроме того, на этом уровне определено, что имена каталогов ограничены восемью символами и не могут иметь расширений. Использование этого уровня предоставляет максимальные шансы на то, что компакт-диск будет читаться на любом компьютере.

На **уровне 2** делаются послабления ограничений на длину. На нем именам файлов и каталогов позволено иметь длину до 31 символа, но при сохранении того же набора символов. На **уровне 3** используются те же ограничения имен, что и на уровне 2, но ослабляется требование по непрерывности файлов. При применении этого уровня файл может состоять из нескольких секций, каждая из которых представляет собой непрерывную последовательность блоков. Одна и та же секция (последовательность блоков) может встречаться в файле несколько раз и даже входить в несколько различных файлов. Если большие фрагменты данных повторяются в нескольких файлах, применение уровня 3 позволяет каким-то образом оптимизировать использование пространства диска, не требуя, чтобы одни и те же данные присутствовали на нем несколько раз.

Расширения Rock Ridge

Стандарт ISO 9660 накладывает целый ряд строгих ограничений. Вскоре после его выпуска представители сообщества UNIX приступили к работе над расширением, позволяющим реализовать файловые системы UNIX на компакт-диске. Эти расширения были названы Rock Ridge, по названию города в фильме Джина Уайлдера «Сверкающие седла» («Blazing Saddles»).

Расширение использует поле, предназначенное для использования системой, чтобы компакт-диск формата Rock Ridge мог читаться на любом компьютере. Все остальные поля сохраняют свое назначение согласно обычному стандарту ISO 9660. Любая система, не работающая с расширениями Rock Ridge, просто игнорирует их и видит обычный компакт-диск.

Расширения делятся на следующие поля:

Поле	Описание
PX	Атрибуты POSIX
PN	Старший и младший номера устройств
SL	Символическая ссылка
NM	Альтернативное имя
CL	Расположение дочернего каталога
PL	Расположение родительского каталога
RE	Перемещение
TF	Отметки времени

Поле *PX* содержит стандартные биты разрешений `rwxrwxrwx` системы UNIX для владельца, группы и всех остальных. Оно также содержит остальные биты слова режима использования, такие как SETUID, SETGID и т.п.

Поле *PN* предназначено для представления на компакт-диске обычных устройств. Оно содержит старший и младший номера устройств, связанных с файлом. Это дает возможность записать на компакт-диск содержимое каталога `/dev` и впоследствии правильно воссоздать его на целевой системе.

Поле *SL* предназначено для символических ссылок. Оно позволяет файлу из одной ФС ссылаться на файл из другой ФС.

Наверное, самым важным является поле *NM*. Оно позволяет связать с файлом второе имя. На это имя не распространяются ограничения по набору используемых символов или длине, накладываемые стандартом ISO 9660, что дает возможность представлять на компакт-диске произвольные имена файлов, принятые в UNIX.

Следующие три поля используются вместе для того, чтобы обойти ограничение ISO 9660 на вложенность каталогов, допускающее только восемь вложений. Использование этих полей позволяет указать на то, что каталог был перемещен, и указать его место в иерархии. Этот способ позволяет эффективно обойти искусственно заданное ограничение на глубину вложенности.

И наконец, поле *TF* содержит три отметки времени, включенные в каждый i-узел UNIX: время создания файла, время его последней модификации и время последнего доступа к этому файлу. Все вместе эти расширения позволяют скопировать файловую систему UNIX на компакт-диск, а затем правильно восстановить ее на другой системе.

Расширения Joliet

Расширить ISO 9660 стремилось не только сообщество UNIX. Компания Microsoft также сочла этот стандарт слишком усеченным (хотя MS-DOS, из-за которой в первую очередь и были введены эти ограничения, принадлежала самой Microsoft). Поэтому Microsoft изобрела ряд расширений, которые получили название Joliet. Они были разработаны, чтобы позволить файловой системе Windows быть скопированной на компакт-диск с последующим восстановлением, точно с такой же целью, с которой расширения Rock Ridge были разработаны для UNIX. По сути, все программы, запускаемые под Windows и использующие компакт-диски, поддерживают Joliet, включая программы, которые копируют данные на записываемые компакт-диски. Обычно такие программы предоставляют выбор между различными уровнями ISO 9660 и Joliet.

В основные расширения, предоставляемые Joliet, входят:

1. Длинные имена файлов.

2. Набор символов Unicode.

3. Вложенность каталогов глубже восьми уровней.

4. Имена каталогов, имеющие расширения.

Первое расширение допускает использование в именах файлов до 64 символов. Второе расширение допускает использование в именах файлов набора символов Unicode. Это расширение играет важную роль для программного обеспечения, предназначенного для использования в тех странах, где не применяется латинский алфавит, например в Японии, Израиле и Греции. Поскольку символы Unicode занимают 2 байта, имя файла в Joliet занимает максимум 128 байт.

Из Joliet, как и из Rock Ridge, удалено ограничение на вложенность каталогов. Каталоги могут быть вложенными настолько глубоко, насколько это нужно. И наконец, имена каталогов могут иметь расширения. Не вполне понятно, зачем именно были включены эти расширения, поскольку каталоги Windows вообще-то никогда не используют расширения, но, возможно, когда-нибудь и станут использовать.

Расширения Romeo для Windows

Стандарт Romeo предоставляет другую возможность записи файлов с длинными именами на компакт-диск. Длина имени может составлять 128 символов, однако использование кодировки Unicode не предусмотрено. Альтернативные имена в этом стандарте не создаются, поэтому программы MS-DOS не смогут прочитать файлы с такого диска.

Вы можете выбрать стандарт Romeo только в том случае, если диск предназначен для чтения приложениями Windows 95 и Windows NT.

Файловая система MS-DOS

ФС MS-DOS — одна из тех систем, с которыми появились первые PC. До появления Windows 98 и Windows ME она была ФС. Она все еще поддерживается на Windows 2000,

Windows XP и Windows Vista, но теперь уже не является стандартом для новых PC, за исключением тех случаев, когда на них используются гибкие диски. Тем не менее она и ее расширение (FAT-32) нашли широкое применение во многих встраиваемых системах. Ее используют большинство цифровых камер. Многие MP3-плееры используют только эту систему. Популярное устройство Apple iPod использует ее в качестве исходной ФС. Таким образом электронных устройств, использующих ФС MS-DOS, стало намного больше, чем когда-либо в прежние времена, и их несомненно больше, чем устройств, использующих более современную файловую систему NTFS.

Чтобы прочитать файл, программа MS-DOS должна сначала сделать СВ `open`, чтобы получить его дескриптор. СВ `open` указывается путь, который может быть как абсолютным, так относительным (от текущего рабочего каталога). В пути ведется покомпонентный поиск до тех пор, пока не будет определено местоположение последнего каталога, после чего происходит его чтение в память. Затем в нем ведется поиск открываемого файла.

Хотя каталоги в ФС MS-DOS переменного размера, в них используются записи фиксированного размера, составляющего 32 байта. Формат записи каталога системы MS-DOS показан на Рис. 2. В этой записи содержится имя файла, его атрибуты, дата и время создания, номер начального блока и точный размер файла. Имена файлов короче $8+3$ символа выравниваются по левому краю полей, и каждое поле по отдельности дополняется пробелами. Поле `Attributes` (атрибуты) представляет собой новое поле, содержащее биты, указывающие, что для файла разрешено только чтение, что файл должен быть заархивирован, что файл является системным или скрытым. Запись в файл, для которого разрешено только чтение, не разрешается. Таким образом осуществляется защита файлов от случайных повреждений. Бит `archived` (архивный) не играет для ОС никакой роли (то есть MS-DOS его не проверяет и не устанавливает). Он предназначен для того, чтобы пользовательские программы архивирования его сбрасывали при создании резервной копии файла, а остальные программы его устанавливали, если файл подвергся модификации. Таким образом, программа резервного копирования получает возможность просто просмотреть состояние этого

бита у каждого файла, чтобы определить, какие именно файлы следует архивировать. Бит `hidden` (скрытый файл) может быть установлен для предотвращения появления файла при отображении содержимого каталога. И наконец, бит `system` (системный) также скрывает файлы и защищает их от случайного удаления командой `del`. Этот бит установлен у всех файлов, содержащих основные компоненты системы MS-DOS.

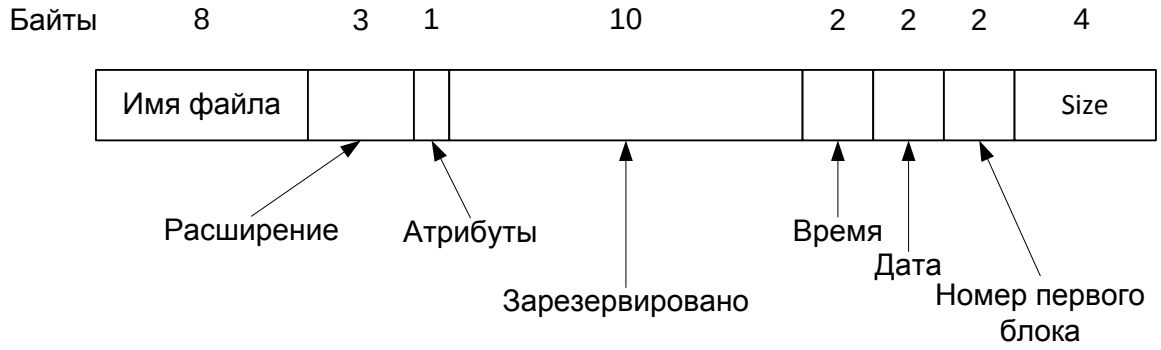

Рис. 2. Запись каталога файловой системы MS-DOS

В записи каталога содержатся также дата и время создания или последнего изменения файла. Точность показания времени составляет ± 2 с, поскольку под него отводится 2-байтовое поле, способное хранить только 65 536 уникальных значений (а в сутках 86 400 с). Это поле времени подразделяется на секунды (5 бит), минуты (6 бит) и часы (5 бит). Счётчики даты в днях также используют три подполя: день (5 бит), месяц (4 бита) и год - 1980 (7 бит). При использовании 7-разрядного числа для года и отсчёта времени с 1980 года самым большим отображаемым годом будет 2107-й. Это означает, что ФС MS-DOS присуща проблема 2108 года. Чтобы избежать катастрофы, пользователи системы MS-DOS должны как можно раньше начать подготовку к 2108 году.

В MS-DOS размер файла хранится в виде 32-разрядного числа, поэтому теоретически файл может иметь размер до 4 Гбайт. Но другие ограничения, рассматриваемые далее, приводят к тому, что максимальный размер файла равен 2 Гбайт или еще меньше. Как ни удивительно, но значительная часть записи А0 байт) остается неиспользуемой.

В MS-DOS файловые блоки отслеживаются через таблицу размещения файлов (FAT — file allocation table), содержащуюся в ОЗУ. В записи каталога хранится номер первого блока файла. Этот номер используется в качестве индекса к одному из 64К элементов FAT в оперативной памяти.

Существуют три версии файловой системы, использующей FAT: FAT-12, FAT-16 и EAT-32, в зависимости от разрядности дискового адреса. Вообще-то, название FAT-32 дано несколько неверно, поскольку для адресов диска используются только 28 бит младшего разряда. Ее следовало бы назвать EAT-28, но число, являющееся степенью двойки, куда более благозвучно.

Для всех вариантов FAT размер дискового блока может быть кратен 512 байтам (это число может быть разным для каждого раздела) и браться из набора разрешенных размеров блока (которые Microsoft называет размерами кластера), разного для каждого варианта. В первой версии MS-DOS использовалась FAT-12 с блоками по 512 байт, задавая максимальный размер раздела $2^{12} \times 512$ байт. При этом максимальный размер дискового раздела составлял около 2 Мбайт, а размер таблицы FAT в памяти был 4096 записей по 2 байта каждая, поскольку при использовании в таблице 12-разрядных записей система работала бы слишком медленно.

Эта система хорошо работает с гибкими дисками, но с появлением жестких дисков возникла проблема. Microsoft решила эту проблему, разрешив дополнительные размеры блоков в 1 Кбайт, 2 Кбайт и 4 Кбайт. Такое изменение сохраняет структуру и размер таблицы FAT-12, но допускает использование размера дисковых разделов вплоть до 16 Мбайт.

Так как MS-DOS поддерживала четыре дисковых раздела на каждый привод, новая ФС FAT-12 работала с дисками объемом до 64 Мбайт. Но кроме этого нужно было еще что-нибудь другое. Поэтому была представлена FAT-16, с 16-разрядными дисковыми указателями. Дополнительно были разрешены размеры блоков в 8 Кбайт, 16 Кбайт и 32 Кбайт. (32 768 — это наибольшее число, кратное степени двойки, которое может

быть представлено 16 разрядами.) Таблица FAT-16 теперь все время занимала 128 Кбайт ОЗУ, но с ростом доступного объема памяти она получила широкое распространение и быстро вытеснила файловую систему FAT-12. Объем наибольшего дискового раздела, поддерживаемого FAT-16, стал равен 2 Гбайт (64 К записей по 32 Кбайт каждая), а наибольший объем диска стал 8 Гбайт, то есть четыре раздела по 2 Гбайт каждый.

Начиная со второго выпуска Windows 95 была представлена ФС FAT-32, использующая 28-разрядные дисковые адреса, а версия MS-DOS, положенная в основу Windows 95, была приспособлена под поддержку FAT-32. В этой системе разделы теоретически могли быть по $2^{28} \times 2^{15}$ байт, но на самом деле они ограничивались размером 2 Тбайт (2048 Гбайт) поскольку внутренне система ведет учет размеров разделов секторами по 512 байт, используя 32-разрядные числа, а $2^{9} \times 2^{32}$ — это 2 Тбайт. Максимальный размер раздела для различных размеров блока и всех трех типов FAT показан в Таблица 1.

Таблица 1. Максимальный размер раздела для различных размеров блока. Пустые клетки означают запрещённые комбинации

Размер блока	FAT-12	FAT-16	FAT-32
0,5 Кбайт	2 Мбайт		
1 Кбайт	4 Мбайт		
2 Кбайт	8 Мбайт	128 Мбайт	
4 Кбайт	16 Мбайт	256 Мбайт	1 Тбайт
8 Кбайт		512 Мбайт	2 Тбайт
16 Кбайт		2048 Мбайт	2 Тбайт
32 Кбайт			2 Тбайт

Кроме поддержки дисков большого объема, ФС FAT-32 имеет два других преимущества над FAT-16. **Во-первых**, диск объемом в 8 Гбайт, на котором используется FAT-32, может быть отформатирован одним разделом. При использовании FAT-16 он должен был иметь четыре раздела, которые появлялись бы для пользователя Windows как логические диски C:, D:, E: и F:. Пользователь должен был сам решать, какой файл на какой диск поместить, и следить за тем, что где находится.

Во-вторых, FAT-32 над FAT-16 заключается в том, что для дискового раздела заданного размера могут использоваться блоки меньшего размера. Например, для 2-гигабайтного дискового раздела система FAT-16 должна использовать 32-килобайтные блоки, иначе при наличии всего 64 К доступных дисковых адресов она не смогла бы покрыть весь раздел. В отличие от нее, FAT-32 может использовать, к примеру, блоки размером 4 Кбайт для 2-гигабайтного дискового раздела. Преимущество блоков меньшего размера заключается в том, что длина большинства файлов менее 32 Кбайт. При размере блока 32 Кбайт даже 10-байтовый файл будет занимать на диске 32 Кбайт. Если средний размер файлов, скажем, равен 8 Кбайт, то при использовании 32-килобайтных блоков около 3/4 дискового пространства будет теряться впустую, то есть эффективность использования диска будет очень низкой. При 8-килобайтных файлах и 4-килобайтных блоках потерь дискового пространства не будет, но зато для хранения таблицы FAT потребуется значительно больше оперативной памяти. При 4-килобайтных блоках 2-гигабайтный раздел будет состоять из 512 К блоков, поэтому таблица FAT должна состоять из 512 К элементов (занимая 2 Мбайт ОЗУ).

ФС MS-DOS использует FAT для учета свободных блоков. Любой нераспределённый на данный момент блок помечается специальным кодом. Когда системе MS-DOS требуется новый блок на диске, она ищет в таблице FAT элемент, содержащий этот код. Поэтому битовый массив или список свободных блоков не нужен.

Файловая система UNIX V7

Даже в ранних версиях системы UNIX применялась довольно сложная многопользовательская файловая система, так как в основе этой системы лежала

операционная система MULTICS. Далее будет рассмотрена ФС V7, разработанная для компьютера PDP-11, сделавшего систему UNIX знаменитой.

ФС представляет собой дерево, начинающееся в корневом каталоге, с добавлением связей, формирующих направленный ациклический граф. Имена файлов могут содержать до 14 любых символов ASCII, кроме слэша (поскольку он служит разделителем компонентов пути) и символа NUL (поскольку он используется для дополнения имен короче 14 символов). Символ NUL имеет числовое значение 0.

Каталог UNIX содержит по одной записи для каждого файла этого каталога. Каждая запись каталога максимально проста, так как в системе UNIX используется система i-узлов. Запись каталога, как показано на Рис. 3, состоит всего из двух полей: имени файла (14 байт) и номера i-узла для этого файла (2 байта). Эти параметры ограничивают количество файлов в файловой системе до 64 К.

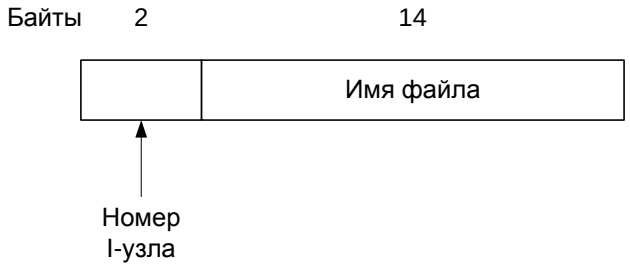

Рис. 3. Запись каталога файловой системы UNIX V7

I-узлы системы UNIX содержат некоторые атрибуты, которые содержат размер файла, три указателя времени (создания, последнего доступа и последнего изменения), идентификатор владельца, группы, информацию о защите и счетчик указывающих на этот i-узел записей в каталогах. Последнее поле необходимо для установления связей. При добавлении к i-узлу новой связи счетчик в i-узле увеличивается на единицу. При удалении связи счетчик в i-узле уменьшается на единицу. Когда значение счетчика достигает нуля, i-узел освобождается, а дисковые блоки возвращаются в список свободных блоков.

Таблица 2. Структура i-узела

Поле	Байты	Описание
Mode	2	Тип файла, биты защиты, биты setuid и setgid
Nlinks	2	Количество каталоговых записей, указывающий на этот i-узел
Uid	2	Идентификатор владельца
Gid	2	Номер группы
Size	4	Размер файла в байтах
Addr	39	Адрес первых 10 дисковых блоков файла и 3 косвенных блока
Gen	1	Счетчик использования i-узла
Atime	4	Время последнего доступа файла
Mtime	4	Время последнего изменения файла
Ctime	4	Время последнего изменения i-узла

Для учета дисковых блоков файла используется общий принцип i-узлов, позволяющий работать с очень большими файлами. Первые 10 дисковых адресов хранятся в самом i-узле, поэтому для небольших файлов вся необходимая информация содержится непосредственно в i-узле, считываемом с диска в оперативную память при открытии

файла. Для файлов большего размера один из адресов в i-узле представляет собой адрес блока диска, называемого **однократным косвенным блоком** (дополнительный блок с адресами блоков файла, если файл не сильно большой, то один из адресов в i-узле указывает на дополнительный блок с адресами). Если и этого недостаточно, используется другой адрес в i-узле, называемый **двукратным косвенным блоком** (дополнительный блок с адресами одинарных косвенных блоков, если одного дополнительного блока не хватает). Если и этого мало, используется **трехкратный косвенный блок** (дополнительный блок с адресами двойных косвенных блоков, если одного одинарного косвенного блока не хватает). Полная схема показана на Рис. 4.

При открытии файла ФС по предоставленному имени файла должна найти его блоки на диске. Рассмотрим, как происходит поиск по абсолютному имени `/usr/ast/mbox`. В этом примере будет использоваться ФС UNIX, хотя для всех иерархических каталоговых систем применяется в основном такой же алгоритм. Сначала ФС определяет местоположение корневого каталога. В системе UNIX его i-узел размещается в фиксированном месте на диске. По этому i-узлу система определяет местоположение корневого каталога, который может находиться в любом месте диска, в данном примере — в блоке 1.

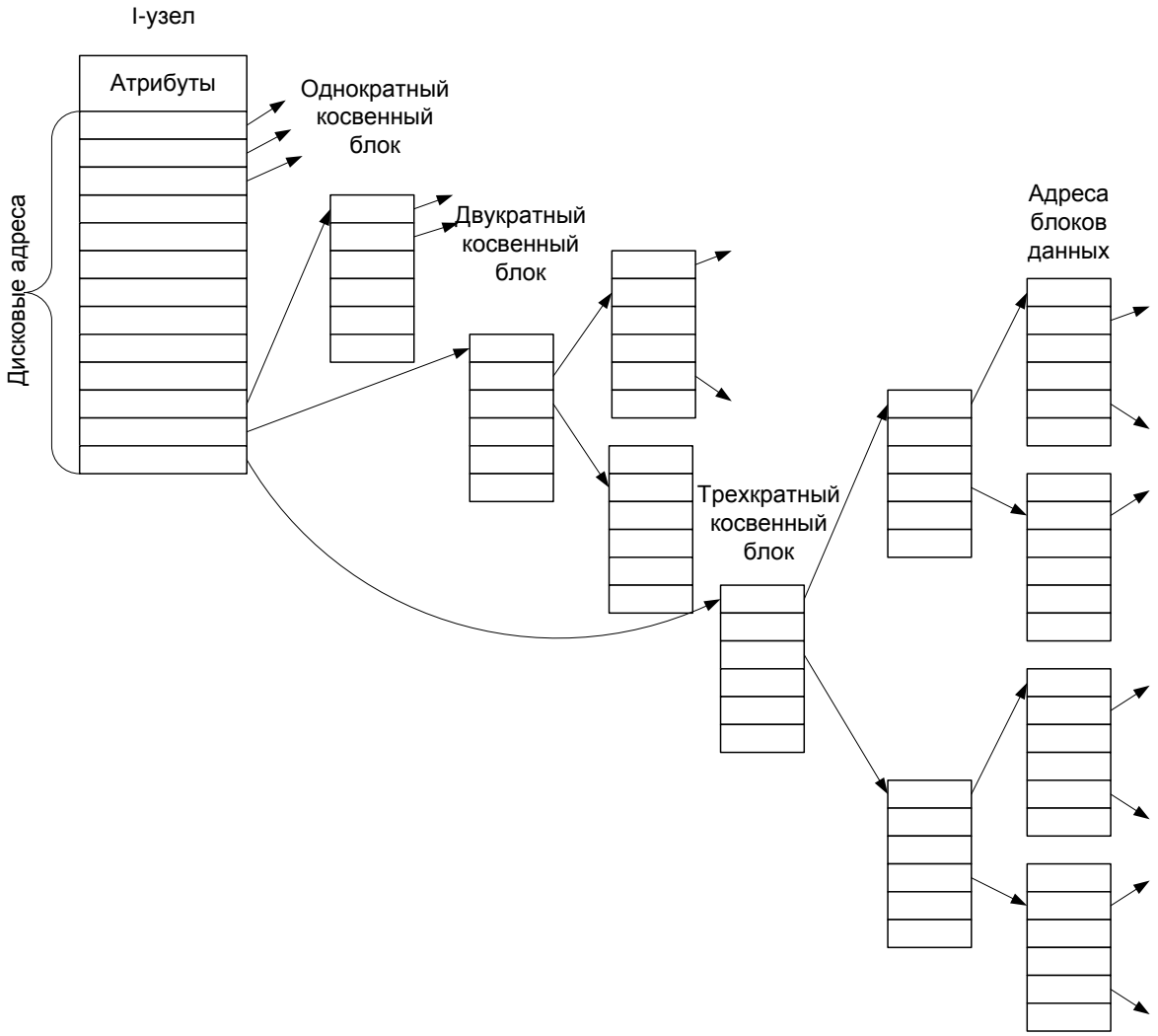

Рис. 4. I-узел UNIX

Затем ФС считывает корневой каталог и ищет в нем первый компонент пути, `usr`, чтобы определить номер i-узла файла `/usr`. Определить местоположение i-узла по его номеру несложно, поскольку у каждого из них есть свое фиксированное место на диске. По этому i-узлу ФС определяет местоположение каталога для `/usr` и ищет в нем следующий компонент, `ast`. Найдя описатель `ast`, ФС получает i-узел для каталога `/usr/ast`. Поэтому i-узлу она может найти сам каталог и искать в нем файл `mbox`. При этом i-узел файла `mbox` считывается в память и остается там, пока файл не будет закрыт. Процесс поиска показан на Рис. 5.

Поиск по относительным именам путей ведется так же, как и по абсолютным, с той лишь разницей, что алгоритм начинает работу не с корневого, а с рабочего каталога. В

каждом каталоге есть элементы «.» и «..», помещаемые в каталог в момент его создания. Элемент «.» содержит номер i-узла текущего каталога, а элемент «..» — номер i-узла родительского каталога. Таким образом, процедура, ведущая поиск файла `../dick/prog.c`, просто находит «..» в рабочем каталоге, разыскивает в нем номер i-узла родительского каталога, в котором ищет описатель каталога `dick`. Для обработки этих имен не требуется никакого специального механизма. Что касается системы каталогов, она представляет собой обыкновенные ASCII-строки, ничем не отличающиеся от любых других имен. Единственная тонкость в том, что элемент «..» в корневом каталоге указывает на сам этот каталог.

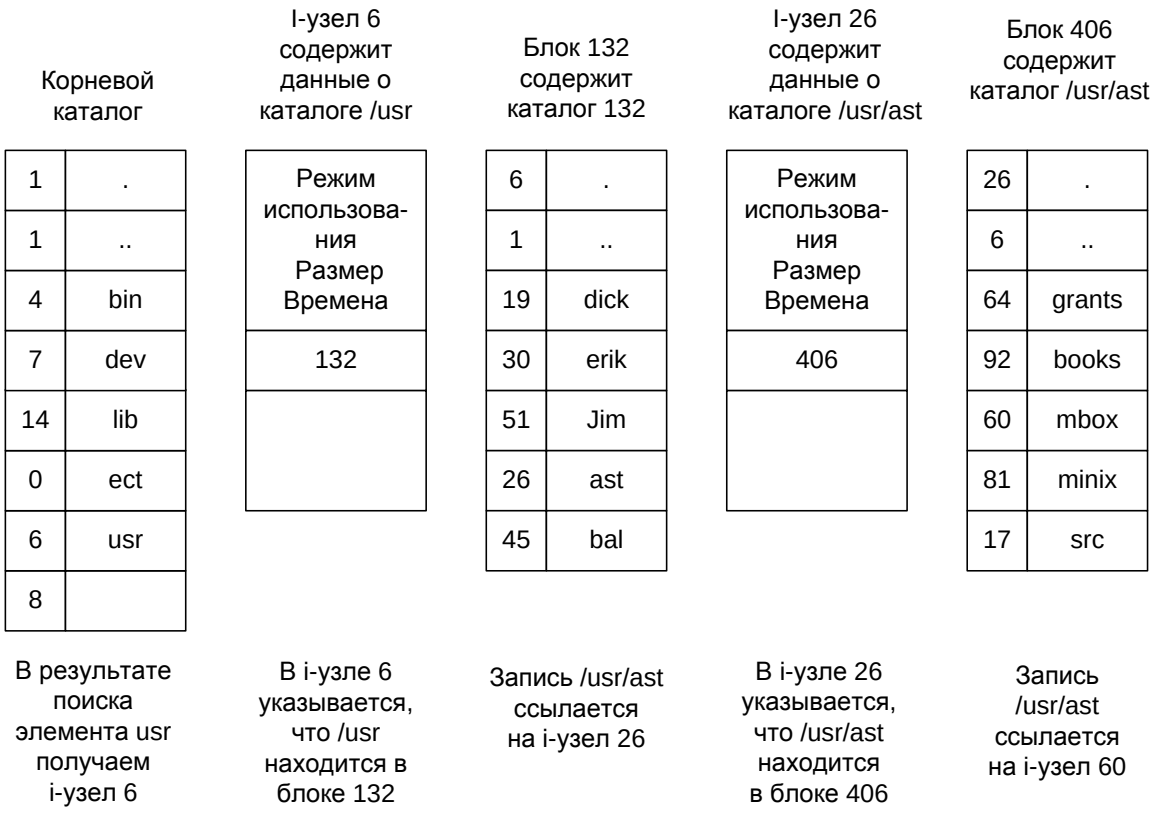

Рис. 5. Этапы поиска /usr/ast/mbox

Файловая система XFS

1 мая 2001 года SGI (SGI, ранее — Silicon Graphics Computer Systems или SGCS) выпустила первый релиз своей журналируемой ФС для Linux.

Особенности XFS - поддержка больших дисков и очень высокая скорость ввода-вывода (на тестировании доходило до 7 ГБайт в секунду). XFS была разработана для ОС IRIX 5.3 SGI Unix, и первая версия этой ФС была представлена в декабре 1994 года. Цель этой ФС - поддержка очень больших файлов и высокая пропускная способность для проигрывания и записи видео в реальном времени.

Для увеличения масштабируемости ФС XFS обширно использует B-деревья. Они используются для пометки свободных расширителей (extents), индексации директорий и для контроля за динамически размещаемыми i-nodes, разбросанными по всей ФС. В дополнение ко всему этому XFS использует асинхронную схему логгирования с журналированием (для защиты meta-данных от сбоев после обновления), и позволяет быстро восстановить ФС.

XFS использует систему распределения места, основанную на расширителях, и имеет такие возможности, как `delayed allocation`, `space pre-allocation` и `space coalescing on deletion`. Данные располагаются с использованием самых больших расширителей из имеющихся, и это позволяет записывать файлы огромной длины. Чтобы сделать управление огромными количествами непрерывных фрагментов эффективным, XFS использует очень большие описания для расширителей. Каждое описание может включать в себя информацию о блоках файловой системы в количестве до 2 миллионов. Описание больших количеств блоков разгружает процессор от такой работы, как поиск записей в `extent map`, чтобы определить, являются ли блоки непрерывными. Теперь можно просто прочитать длину всего расширителя, который уже

является непрерывным, а не смотреть на каждую запись, определяя, является ли она продолжением предыдущей записи.

XFS позволяет использовать различные размеры блоков – от 512 байт до 64 килобайт. Изменение размеров блоков может повлиять на фрагментацию. Файловые системы с большим количеством маленьких файлов обычно используют маленькие размеры блоков, чтобы избежать лишних трат свободного места в случаях, когда размер файла меньше, чем размер блока, а системы с большими файлами обычно делают противоположный выбор и используют большие размеры блоков, чтобы уменьшить фрагментацию ФС.

XFS - это комплексная разработка, основанная на IRIX, и она очень связана с IRIX, поэтому при портировании на Linux все было полностью переработано и написано с нуля. Результат - модуль pagebuf для Linux, обеспечивающий интерфейс между XFS и подсистемой виртуальной памяти, а так же между XFS и Linux block device layer.

XFS поддерживает ACL'ы (Access Control List – список контроля доступа), и transactional quotes. В Linux она поддерживает квотирование для группы, в отличии от IRIX квотирования для проекта, потому что именно так квотирование реализуется в файловых системах для Linux. Более необычные функции XFS из оригинальной версии для IRIX, позволяющие оказывать специальные сервисы различным приложениям (например, обслуживание video в реальном времени) еще не были перенесены в Linux-версию.

Нормальный режим работы для XFS - это использование асинхронно ведущегося логгирования. Из-за этого XFS приобретает 2 вещи:

1. Множественные обновления могут быть произведены одной операцией записи в лог. Это увеличивает эффективность ведения лога.

2. Быстродействие изменения meta-данных обычно делается независимым от скорости устройства, на котором записаны данные. Вообще-то, эта независимость ограничена количеством буферизации, используемой для ведения лога, но все равно - это намного лучше, чем синхронные обновления, используемые в старых файловых системах.

XFS также имеет большое количество инструментов для dump'инга, восстановления, наращивания, использования ACL'ов и дисковых квот, и т.д.

Файловая система EXT4

Четвертая версия расширенной ФС (Fourth Extended File System – EXT4) появилась на свет в связи с тем, что разработчики требовали добавления новых и расширения старых функций ext3. С этим шагом был связан ряд проблем:

1. Некоторые из новых функций нарушали обратную совместимость.

2. Код ext3 мог стать слишком сложным и трудным для поддержки.

3. Эти изменения могли сделать весьма надежную систему ext3 нестабильной.

Исходя из этих соображений, в июне 2006 года разработчики приняли решение выделить работу над ext4 в отдельную ветку. С этого времени началась разработка ext4, которая оставалась незаметной для большинства обычных пользователей и администраторов Linux. С выходом ядра 2.6.19 в ноябре 2006 года ext4 впервые появилась в стабильном ядре, хотя она и была помечена как экспериментальная (и остается таковой до сих пор), поэтому большинство пользователей проигнорировали ее.

Особенности

1. Совместимость.

Любая ФС типа Ext3 может быть конвертирована в Ext4 путём простой процедуры, состоящей из запуска пары команд в режиме «только чтение». Это означает, что вы можете повысить производительность и вместимость и улучшить возможности вашей имеющейся ФС без переформатирования и без переустановки ОС и программ. Если вы хотите получить преимущества Ext4 в production-системе, вы также можете обновить ФС.

Ext4 будет использовать новые структуры только для новых данных, а старые при этом останутся неизменными. При необходимости их можно будет читать и изменять. Это

безусловно означает, что, единожды сменив ФС на Ext4, вернуть Ext3 будет уже невозможно.

2. Больший размер файлов и файловой системы.

На сегодняшний день максимальный размер ФС Ext3 равен 16 Тбайт, а размер файла ограничен 2 Тбайт. В Ext4 добавлена 48-битная адресация блоков, что означает, что максимальный размер этой ФС равен 1 Эбайту, и файлы могут быть размером до 16 Тбайт. Почему 48-битная, а не 64-битная? Имелся ряд ограничений, которые необходимо было бы снять, чтобы сделать Ext4 полностью 64-битной, и такой задачи перед Ext4 не ставилось. Структуры данных в Ext4 проектировались с учётом требуемых изменений, поэтому однажды в будущем поддержка 64 бит в Ext4 появится. Пока же придётся довольствоваться одним экзабайтом.

3. Масштабируемость подкаталогов.

В настоящий момент один каталог Ext3 не может содержать более, чем 32000 подкаталогов. Ext4 снимает это ограничение и позволяет создавать неограниченное количество подкаталогов.

4. Экстенты.

Традиционные файловые системы, произошедшие от Unix, такие как Ext3, используют схему непрямого отображения блоков для отслеживания каждого блока, отвечающего за хранение данных файла. Такой подход неэффективен для больших файлов, особенно при операциях удаления и усечения таких файлов, поскольку карта соответствия содержит по одной записи на каждый отдельный блок. В больших файлах блоков много, их карты соответствия большие, и обрабатываются они медленно.

В современных ФС применяется иной подход, основанный на так называемых экстентах. **Экстент** — это набор последовательных физических блоков. Он как бы говорит нам: «Эти данные находятся в следующих n блоках». Например, файл размером в 100 Мбайт может храниться в единственном экстенте такого же размера, вместо того, чтобы быть разбитым на 25600 4-килобайтных блоков, адресуемых путём непрямого отображения. Огромные файлы можно разделить на несколько экстентов.

Благодаря применению экстентов улучшается производительность, а также уменьшается фрагментированность, поскольку экстенты способствуют непрерывному размещению данных.

5. Многоблочное распределение.

Если в Ext3 нужно записать на диск новые данные, специальный механизм распределения блоков определяет, какие блоки из числа свободных будут для этого использованы. Проблема в том, что в Ext3 этот механизм распределяет в один присест только один блок (4Кбайта). Это означает, что, если нужно записать, скажем, ранее упомянутые 100Мбайт данных, нужно будет обратиться к механизму распределения 25600 раз. Мало того, что это неэффективно, это к тому же не позволяет оптимизировать политику распределения, поскольку соответствующий механизм не имеет понятия о реальном объёме данных, подлежащем записи, а знает только об одном-единственном блоке.

Ext4 использует механизм многоблочного распределения (multiblock allocator, mballoc), который позволяет распределить любое количество блоков с помощью единственного вызова и избежать огромных накладных расходов. Благодаря этому производительность существенно вырастает, что особенно заметно при отложенном распределении с использованием экстентов. Эта возможность никак не влияет на формат данных.

6. Отложенное распределение

Отложенное распределение представляет собой способ повышения производительности, не влияющий на формат данных и представленный в современных файловых системах, таких как XFS, ZFS, btrfs и Reiser 4.

Суть этого метода состоит в отсрочке выделения блоков насколько это возможно — по контрасту с подходом, применямым в традиционных файловых системах : распределять блоки сразу, при первой возможности. Например, если процесс осуществляет запись вызовом `write()`, файловая система распределит блоки под запись немедленно — даже если данные пока не будут записываться на диск, а будут находиться какое-то время в кэше. **Недостатки** такого подхода, например, в том, что, если процесс непрерывно

осуществляет запись в растущий файл, последовательные вызовы `write()` постоянно распределяют блоки данных, и при этом неизвестно, будет ли файл расти далее.

При использовании отложенного распределения блоки сразу не выделяются при обращении к `write()`. Вместо этого распределение откладывается до момента, когда файл будет записан из кэша на диск. Благодаря этому механизм получает возможность оптимизировать процесс распределения. Наибольший выигрыш получается при использовании двух ранее упомянутых возможностей — экстентов и многоблочного распределения, поскольку часто встречается ситуация, когда окончательный файл пишется на диск в виде экстентов, распределённых с помощью `mballoc`. Это даёт существенный прирост производительности, и иногда сильно снижает фрагментированность данных.

7. Быстрый fsck

Fsck — это очень медленная операция, особенно это касается её первой стадии, проверки всех inodes в ФС.

В Ext4 после inode-таблицы каждой группы хранится список неиспользованных inodes (снабжённый для надёжности контрольной суммой), поэтому fsck такие inodes не будет проверять. Результатом является уменьшение времени проверки от 2 до 20 раз, что зависит от количества используемых inodes.

То, что список неиспользуемых inodes составляется fsck, а не Ext4, будет хорошо заметно, если вы запустите fsck, чтобы он построил список неиспользуемых inodes, и когда только следующий запуск fsck пройдёт быстрее (запуск fsck всё равно необходим при конвертировании Ext3 в Ext4).

Кроме того, на ускорение fsck влияет и другая особенность — «гибкие группы блоков», также они ускоряют и другие файловые операции.

8. Контрольные суммы журнала

Журнал является наиболее часто используемой частью диска, вследствие чего блоки, из которых он состоит, становятся особенно чувствительными к отказам оборудования. Более того, попытка восстановления при повреждённом журнале может привести к ещё более массовым повреждениям в данных. Ext4 подсчитывает контрольные суммы журнальных данных, что позволяет определить факт их повреждения. У этого есть и ещё одно преимущество: благодаря контрольным суммам можно превратить систему двухфазной фиксации журнала Ext3 в однофазную, что ускоряет файловые операции в отдельных случаях до 20 %, таким образом, улучшаются одновременно и надёжность, и производительность.

Примечание: часть, отвечающая за производительность — асинхронное протоколирование, — сейчас по умолчанию отключена, и будет включена в одном из последующих релизов, когда удастся добиться надёжной его работы.

9. Режим без журналирования

Журналирование обеспечивает целостность ФС путём протоколирования всех происходящих на диске изменений. Но оно также вводит дополнительные накладные расходы на дисковые операции. В некоторых особых ситуациях журналирование и предоставляемые им преимущества могут оказаться излишними. Ext4 позволяет отключить журналирование, что приводит к небольшому приросту производительности.

10. Онлайн-дефрагментация

Эта функция пока в разработке и будет включена в один из будущих релизов.

Хотя отложенное и многоблочное распределение и экстенты помогают уменьшить фрагментированность ФС, со временем она всё-таки может вырасти.

Например: вы создаёте три файла в одном каталоге и они расположены на диске друг за другом. Потом, однажды вы решаете обновить второй файл, и при этом файл становится несколько больше — так, что места для него становится недостаточно. При этом нет никаких других решений, кроме как отделить не вмещающийся фрагмент файла и положить его на другое место диска или выделить файлу последовательную область диска большего размера в другом месте, вдалеке от первых двух файлов, что приведёт к перемещениям головки диска, если приложению потребуется считать все файлы в каталоге (скажем, менеджер файлов будет создавать эскизы для файлов изображений).

Помимо этого, ФС может заботиться только об определённых типах фрагментации и она не может знать, например, что она должна хранить все файлы, требуемые при загрузке, рядом друг с другом, поскольку она просто не знает, какие из них требуются при загрузке. Чтобы решить эту проблему, Ext4 будет поддерживать онлайн-дефрагментацию.

Также имеется утилита `e4defrag`, которая позволяет дефрагментировать как отдельные файлы, так и всю ФС.

11. Улучшения, связанные с inode

Бóльшие `inodes`: Ext3 поддерживает `inodes` настраиваемого размера (путём указания mkfs параметра -I), однако размер `inode` по умолчанию — 128 байт. В Ext4 он будет 256 байт. Это потребовалось, чтобы вместить несколько дополнительных полей (таких как наносекундные временные метки и версии `inode`), а оставшееся в `inode` место будет использовано для хранения тех расширенных атрибутов, которые достаточно малы, чтобы там поместиться. Это позволит сделать доступ к таким атрибутам намного быстрее и улучшит производительность приложений, использующих их, в 3-7 раз.

Суть резервирования `inode` состоит в выделении нескольких `inodes` при создании каталога в ожидании того, что они будут использованы в будущем. Это улучшает производительность, потому что вновь создаваемые в этом каталоге файлы смогут использовать зарезервированные `inodes`. Поэтому создание и удаление файлов производится более эффективно.

12. Устойчивое прераспределение

Эта возможность, доступная уже в Ext3 в последних версиях ядра и эмулируемая `glibc` в файловых системах, которые её не поддерживают, позволяет приложениям заранее распределять дисковое пространство, сообщая о своих потребностях ФС. Та, в свою очередь, выделяет необходимое количество блоков и структур данных, но они пусты до тех пор, пока приложение в реальности не осуществит в них запись.

Применений этому несколько: **во-первых**, чтобы предотвратить выполнение того же самого приложениями неэффективно заполняющими файлы нулями — нужные блоки будут выделены разом.

Во-вторых, чтобы снизить фрагментацию — опять же потому, что блоки выделяются однократно, настолько непрерывно, насколько это возможно.

В-третьих, чтобы гарантировать, что приложение будет иметь столько места, сколько ему требуется, что особенно важно для приложений, работающих в реальном времени, поскольку файловая система может вдруг переполниться в процессе выполнения важной операции.

13. Механизм «шлагбаумов» по умолчанию включен

Это опция, обеспечивающая целостность ФС ценой некоторой потери производительности.

Литература

1. .Э. Таненбаум, А. Вудхалл. Операционные системы: разработка и реализация. Классика CS. –СПб.: Питер, 2006. –576 с.

2. Олифер В., Олифер Н. Сетевые операционные системы: Учебник для вузов. 2-е изд. — СПб.: Питер, 2009. — 669 с.: ил.

3. Таненбаум Э. Современные операционные системы. 3-е изд. — СПб.: Питер, 2010. — 1120 с: ил. — (Серия «Классика computer science»).

4. Microsoft Development Network. URL: http://msdn.com