Оглавление

1 Файловые системы ОС Linux

Изначально в операционной системе Linux использовалась файловая система операционной системы MINIX.

Однако в системе MINIX длина имен файлов ограничивалась 14 символами (для совместимости с UNIX Version 7), а максимальный размер файла был равен 64 Мбайт. Поэтому у разработчиков операционной системы Linux практически сразу появился интерес к усовершенствованию системы.

Первым шагом вперед стала файловая система **ext** в которой длина имен файлов была увеличена до 255 символов, а размер файлов - до 2 Гбайт. Однако эта система была медленнее файловой системы MINIX. поэтому исследования некоторое время продолжались.

Наконец, была разработана файловая система **ext2**. с длинными именами файлов, длинными файлами и высокой производительностью. Эта файловая система и стала основной файловой системой Linux.

Однако операционная система Linux также поддерживает еще более десятка файловых систем, используя для этого файловую систему **NFS** .

При компоновке операционной системы Linux можно сделать выбор файловой системы, которая будет **встроена в ядро**.

Другие файловые системы при необходимости могут динамически подгружаться во время исполнения в виде модулей.

Файловая система **ext2** очень похожа на файловую систему Berkeley Fast File system с небольшими изменениями.

Вместо того чтобы использовать группы цилиндров, что практически ничего не значит при современных дисках с виртуальной геометрией, она делит диск на группы блоков, независимо от того, где располагаются границы между цилиндрами.

Каждая группа блоков начинается с **суперблока,** в котором хранится информация о том, сколько блоков и **i-узлов** находятся в данной группе, о размере группы блоков и т. д.

Затем следует **описатель группы**, содержащий информацию о расположении битовых массивов, количестве свободных блоков и i-узлов в группе, а также количестве каталогов в группе. Эта информация важна, так как файловая система **ext2** пытается распространить каталоги равномерно по всему диску.

В **двух битовых массивах** ведется учет **свободных блоков** и **свободных i-узлов**. Размер каждого битового массива равен одному блоку.

При размере блоков в 1 Кбайт такая схема ограничивает размер группы блоков 8192 блоками и 8192 i-узлами. (На практике ограничение числа i-узлов никогда не встречается, так как блоки заканчиваются раньше.)

Затем располагаются сами **i-узлы**. Размер каждого i-узла — **128 байт**, что в два раза больше размера стандартных i-узлов в UNIX.

Дополнительные байты в **i-узле** используются следующим образом:

- Вместо 10 прямых и 3 косвенных дисковых адресов файловая система Linux позволяет **12 прямых и 3 косвенных дисковых адреса**.
- Кроме того, длина адресов увеличена с **3** до **4 байт**, и это позволяет поддерживать дисковые разделы размером более 224 блоков (16 Гбайт), что уже стало проблемой для UNIX.

Работа файловой системы похожа на функционирование быстрой файловой системы Berkeley.

Однако в отличие от BSD, в системе Linux используются дисковые блоки только одного размера, **1 Кбайт**.

Быстрая файловая система Berkeley использует 8-килобайтные блоки, которые затем разбиваются при необходимости на килобайтные фрагменты.

Файловая система **ext2** делает примерно то же самое, но более простым способом. Как и система Berkeley, когда файл увеличивается в размерах, файловая система **ext2** пытается поместить новый блок файла в ту же группу блоков, что и остальные блоки, желательно сразу после предыдущих блоков. Кроме того, при создании нового файла в каталоге файловая система **ext2** старается выделить ему блоки в той же группе блоков, в которой располагается каталог. Новые каталоги, наоборот, равномерно распределяются по всему диску.

Другой файловой системой Linux является файловая система **/proc** (process).

Идея этой файловой системы изначально была реализована в 8-й редакции операционной системы UNIX, созданной лабораторией Bell Labs, а позднее скопированной в 4.4BSD и System V.

Однако в операционной системе Linux данная идея получила дальнейшее развитие.

Основная концепция этой файловой системы заключается в том, что **для каждого процесса системы создается подкаталог в каталоге /proc.**

Имя подкаталога формируется из PID процесса в десятичном формате. Например, /proc619 — это каталог, соответствующий процессу с PID 619. В этом каталоге располагаются файлы, которые хранят информацию о процессе — его командную строку, строки окружения и маски сигналов. В действительности этих файлов на диске нет. Когда они считываются, система получает информацию от фактического процесса и возвращает ее в стандартном формате.

Многие расширения, реализованные в операционной системе Linux, относятся к файлам и каталогам, расположенным в каталоге /proc. Они содержат информацию о центральном процессоре, дисковых разделах, векторах прерывания, счетчиках ядра, файловых системах, подгружаемых модулях и о

многом другом. Непривилегированные программы пользователя могут читать большую часть этой информации, что позволяет им узнать о поведении системы безопасным способом. Некоторые из этих файлов могут записываться в каталог **/proc**, чтобы изменить параметры системы.

2. Файловые системы ext

После своего появления в 1993 году файловая система **ext2** (second extended file system, **ext2fs**) стала самой распространенной файловой системой для **ОС Linux.**

Целью создания файловой системы **ext2** являлось обеспечение высокой производительности и устойчивости файловой системы, а также поддержка дополнительных возможностей.

Как и в любой файловой системе UNIX, в **ext2** можно выделить следующие элементы:

- **Блоки;**
- **Группы блоков;**
- **Индексные дескрипторы;**
- **Суперблок;**

2.1 Блоки

Всё пространство раздела диска разбивается на блоки фиксированного размера, кратные размеру сектора: **1024, 2048, 4096 или 8192 байт**.

Размер блока указывается при создании файловой системы в разделе диска. Все блоки имеют порядковые номера.

По умолчанию, во время форматирования, **5% блоков** резервируются исключительно для нужд привилегированных пользователей **root**. Такой подход используется в целях безопасности, чтобы процессы, запущенные с привилегиями пользователя root, могли выполняться даже в случае, если вредоносный пользовательский процесс либо процесс с ошибками займет все остальные блоки в системе. Оставшиеся 95% блоков могут использоваться другими пользователями для хранения данных.

С целью уменьшения фрагментации и количества перемещений головок жёсткого диска при чтении больших массивов данных блоки объединяются в **блочные группы.**

2.2 Блочные группы

Все блоки раздела ext2 разбиваются на группы блоков, называемые **блочными группами** (block group).

Для каждой группы создаётся отдельная запись в глобальной дескрипторной таблице, в которой хранятся основные параметры:

- **номер блока в битовой карте блоков,**
- **номер блока в битовой карте inode,**
- **номер блока в таблице inode,**
- **число свободных блоков в группе,**
- **число индексных дескрипторов, содержащих каталоги**.

Битовая карта блоков - это структура, каждый бит которой показывает, отведён ли соответствующий ему блок какому-либо файлу. Если бит равен 1, то блок занят.

Аналогичную функцию выполняет битовая карта индексных дескрипторов, которая показывает, какие именно индексные дескрипторы заняты, а какие нет.

Ядро Linux, используя число индексных дескрипторов, содержащих каталоги, пытается равномерно распределить inode каталогов по группам, а inode файлов старается по возможности переместить в группу с родительским каталогом. Все оставшееся место, обозначенное в таблице как данные, отводится для хранения файлов.

Все родственные данные файловая система пытается сохранить в одной блочной группе. Это позволяет уменьшить время поиска больших групп родственных данных (например, индексных узлов каталогов, файлов и самих данных), поскольку блоки внутри блочной группы хранятся в смежных областях диска.

Первым блоком в структуре блочной группы является **суперблок**.

Суперблок содержит важную информацию не только об отдельной блочной группе, но и обо всей файловой системе. Сюда относятся сведения об общем количестве блоков и индексных узлов **inode** файловой системы, размере данной блочной группы, времени монтирования файловой системы и другая дополнительная информация.

Поскольку содержимое суперблока играет немаловажную роль для целостности файловой системы, в некоторых блочных группах хранится дополнительная копия суперблока. Благодаря этому в случае порчи одной из копий файловая система может быть восстановлена при помощи резервной копии.

Блочная группа содержит несколько структур данных, обеспечивающих файловые операции над этой группой. В качестве одной из таких структур выступает **таблица индексных узлов** (inode table), которая содержит записи для каждого inode в блочной группе.

При форматировании файловой системы каждой блочной группе в ext2 назначается фиксированное количество индексных узлов. Общее число inode в системе зависит от соотношения количества байтов на один индексный узел, задаваемого во время форматирования раздела.

Поскольку размер таблицы индексных узлов является фиксированным, единственный способ увеличения количества индексных узлов в файловой системе ext2 заключается в увеличении размера файловой системы. Объекты inode в таблице индексных узлов каждой группы обычно ссылаются на файл и данные каталога, хранимые внутри этой группы, что позволяет уменьшить время, необходимое для загрузки файлов с диска, благодаря смежности их расположения.

В блочных группах также имеются блоки, содержащие битовые **карты распределения индексных узлов** (inode allocation bitmap), которые позволяют отслеживать их использование внутри блочной группы. Каждый бит битовой карты соответствует определенной записи в таблице индексных узлов группы.

При выделении дискового пространства под файл, для его представления выбирается доступный inode из таблицы индексных узлов. В битовой карте

распределения устанавливается бит, соответствующий индексу inode в таблице индексных узлов, который свидетельствует о занятости данного inode.

Например, если запись 45 в таблице индексных узлов связана с файлом, в битовой карте распределения inode 45-й бит будет установлен в 1.

Когда необходимость в индексном узле отпадает, в битовой карте распределения индексных узлов очищается соответствующий бит, делая inode доступным для повторного использования. Та же самая стратегия применяется и в **битовых картах распределения блоков** (block allocation bitmap), с помощью которых отслеживается использование блоков в группах.

Еще один элемент метаданных, присутствующий в каждой блочной группе, называется **дескриптором группы** (group descriptor). Дескриптор группы содержит номера блоков, соответствующие местоположению битовой карты распределения блоков и таблицы индексных узлов. Кроме того, дескриптор группы может хранить информацию об учетных записях группы, такую как, например, количество свободных блоков и индексных узлов в группе. Каждая блочная группа содержит избыточную копию дескриптора группы, предназначенную для восстановления.

Оставшиеся блоки каждой блочной группы используются для хранения файловых данных и каталогов.

2.3 Индексные дескрипторы

Индексный дескриптор, или **inode** (*information node*) - это специальная структура, которая содержит информацию об атрибутах и физическом расположении данных файла. Индексные дескрипторы объединены в таблицу, которая содержится в начале каждой группы блоков.

Каждый индексный дескриптор в ext2 (ext2 inode) хранит информацию об одном файле либо каталоге.

Каждый индексный дескриптор содержит 15 указателей на блоки данных (каждый величиной в 32 бита). Первые двенадцать указателей ссылаются непосредственно на первые 12 блоков данных. Указатели с 13-го по 15-тый предназначены для косвенной адресации блоков данных.

- **13-й косвенный указатель** (indirect pointer) определяет местоположение блока, содержащего указатели на блоки данных.
- 14-й указатель представляет собой **указатель двойной косвенной адресации** (doubly indirect pointer). Указатель двойной косвенной адресации ссылается на блок простых косвенных указателей.
- 15-й указатель представляет собой **указатель тройной косвенной адресации** (triply indirect pointer) - он указывает местоположение блока указателей двойной косвенной адресации.

2.4 Суперблок

Суперблок - основной элемент файловой системы ext2. Он содержит общую информацию о файловой системе:

- **общее число блоков и индексных дескрипторов в файловой системе,**
- **число свободных блоков и индексных дескрипторов в файловой системе,**
- **размер блока файловой системы,**
- **количество блоков и индексных дескрипторов в группе блоков,**
- **размер индексного дескриптора,**
- **идентификатор файловой системы.**

Суперблок находится в 1024 байтах от начала раздела. От целостности суперблока напрямую зависит работоспособность файловой системы. Операционная система создаёт несколько резервных копий суперблока на случай повреждения раздела.

В следующем блоке после суперблока располагается глобальная дескрипторная таблица - описание групп блоков, представляющее собой массив, содержащий общую информацию обо всех группах блоков раздела.

2.5 Файловая система ext3

Является файловой системой по умолчанию во многих дистрибутивах. Основана на ФС ext2.

Основное отличие от ext2 состоит в том, что ext3 – **журналируемая** файловая система

Стандартом предусмотрено три режима журналирования:

- ***writeback*:** в журнал записываются только **метаданные файловой системы**, то есть информация о её изменении. Не может гарантировать целостности данных, но уже заметно сокращает время проверки по сравнению с ext2;
- ***ordered***: то же, что и writeback, но **запись данных в файл производится гарантированно *до* записи информации о изменении этого файла**. Немного снижает производительность, также не может гарантировать целостности данных (хотя и увеличивает вероятность их сохранности при дописывании в конец существующего файла);
- ***journal***: полное журналирование как метаданных ФС, так и пользовательских данных. Самый медленный, но и самый безопасный режим; может гарантировать целостность данных при хранении журнала на отдельном разделе (а лучше - на отдельном жёстком диске).

Режим журналирования указывается в строке параметров для программы **mount,** например:

mount /dev/hda6 /mnt/disc -t ext3 -o data=<режим>

либо в файле /etc/fstab.

Файловая система ext3 может поддерживать файлы размером до 1 терабайта.

С Linux-ядром 2.4 объём файловой системы ограничен максимальным размером блочного устройства, что составляет 2 терабайта.

В Linux 2.6 (для 32-разрядных процессоров) максимальный размер блочных устройств составляет 16 терабайт, однако ext3 поддерживает только до 4 терабайт.

Табл. Ограничения размеров файлов и каталогов ext3

Размер блока	Макс. размер файла	Макс. размер файловой системы
1 Кб	**16 GB**	**до 2 TB**
2 Кб	**256 GB**	**до 4 TB**
4 Кб	**2 TB**	**до 8 TB**
8 Кб	**2 TB**	**до 16 TB**

2.6 Файловая система ext4

В ext4 появилось несколько новых улучшений производительности и надежности. ext4 поддерживает файловые системы **до одного экзабайта** (1000 петабайт).

Хотя это и большая цифра, потребление места на устройствах хранения увеличивается, так что ext4 была разработана с расчетом на будущее.

Файлы в ext4 могут достигать размера 16 ТБ (при блоках размером 4 КБ), что в восемь раз больше, чем в ext3.

Глубина поддиректорий в ext4 была увеличена с 32 КБ до фактически бесконечной. Это может показаться избыточным, но тут надо принимать во внимание возможную иерархию файловой системы размером в экзабайт.

Было оптимизировано индексирование директорий, которое теперь использует хэширующую структуру, подобную B-дереву. Поэтому, несмотря на гораздо больший размер, поиск в ext4 работает очень быстро.

2.6.1 Экстенты

Одним из главных недостатков системы ext3 был ее метод выделения места на дисках. Дисковые ресурсы для файлов выделялись с помощью битовых карт свободного места - способа, не выделяющегося ни скоростью, ни масштабируемостью. Формат, применяемый в ext3, очень эффективен для маленьких файлов, но очень неэффективен для больших.

Поэтому для улучшения выделения ресурсов и поддержки более эффективной структуры хранения данных в ext4 вместо битовых карт применяются экстенты.

Экстент - это способ представления непрерывной последовательности блоков памяти. При использовании экстентов сокращается количество метаданных, так как вместо информации о том, где находится каждый блок памяти, экстенты содержат информацию о том, где находится большой список непрерывных блоков памяти.

В ext4 для эффективного представления маленьких файлов в экстентах применяется уровневый подход, а для эффективного представления больших файлов применяются деревья экстентов.

Например, один индексный дескриптор в ext4 имеет достаточно места, чтобы ссылаться на четыре экстента (каждый из которых представляет множество последовательных блоков). Для больших (в том числе фрагментированных) файлов дескриптор может содержать ссылки на другие индексные дескрипторы, каждый из которых может указывать на концевой узел (указывающий на экстенты). Такое дерево экстентов постоянной глубины предоставляет мощный механизм представления больших, потенциально фрагментированных файлов. Также узлы имеют механизмы самопроверки для защиты от повреждений файловой системы.

2.6.2 Производительность

ext4, вместе с повышением масштабируемости и надежности, имеет ряд улучшений, связанных с производительностью.

- **Предварительное выделение на файловом уровне**

Некоторые приложения, например, базы данных, рассчитывают, что их файлы будут храниться в непрерывных блоках (чтобы использовать оптимизацию при последовательном чтении данных с дисков, а также минимизировать количество команд Read в расчете на блок данных). Хотя сегменты непрерывных блоков можно получить с помощью экстентов, есть и другой, более грубый метод: предварительно выделять очень большие сегменты непрерывных блоков желаемого размера. ext4 делает это с помощью нового системного вызова, который осуществляет предварительное выделение и инициализацию файла заданного размера. Далее можно записывать необходимые данные и читать их посредством операций Read .

- **Отложенное выделение блоков памяти**

Суть оптимизации в том, что откладывание выделения физических блоков памяти до того, пока они действительно будут записаны, позволяет выделять больше последовательных блоков. Это похоже на предварительное выделение, за

исключением того, что эту задачу система выполняет автоматически. Но если размер файла известен заранее, лучше применять предварительное выделение.

- **Выделение блоков памяти группами**

И последняя оптимизация, также связанная с последовательными блоками, - это групповое выделение блоков в ext4. В ext3 каждый блок выделяется по отдельности. Поэтому иногда получалось, что для последовательных данных выделенные блоки располагались не последовательно. В ext4 эта проблема решена за счет того, что выделение группы блоков происходит за один раз, поэтому фрагментирование маловероятно. Связанные данные хранятся на диске вместе, что в свою очередь позволяет оптимизировать их чтение.

Другим аспектом группового выделения блоков является объем работы, необходимой для выделения блоков. В ext3 выделение осуществляется по одному блоку за раз. Выделение блоков группами требует гораздо меньшего количества вызовов, что ускоряет выделение блоков.

- **Надежность**

При увеличении размеров файловых систем до уровней, поддерживаемых ext4, неизбежно встает проблема повышения надежности. Для ее решения в ext4 предусмотрено множество механизмов самозащиты и самовосстановления.

- **Контрольная сумма журнала файловой системы**

Как и ext3, ext4 является журналируемой файловой системой. Журналирование позволяет производить изменения более надежным образом и гарантировать целостность данных даже в случае краха системы или сбоя питания во время выполнения операции. В результате снижается вероятность повреждения файловой системы.

Но даже с применением журналирования повреждения системы возможны, если в журнал попадут ошибочные записи. Для борьбы с этим в ext4 реализована проверка контрольных сумм записей журнала, чтобы гарантировать, что в нижележащую файловую систему будут внесены правильные изменения.

В зависимости от нужд пользователя ext4 может работать в разных режимах журналирования. Например, ext4 поддерживает режим обратной записи (в журнал

заносятся только метаданные), режим упорядочивания (метаданные заносятся в журнал, но только после записи самих данных), а также самый надежный - журнальный режим (в журнал заносятся как данные, так и метаданные). Заметьте, что хотя журнальный режим - это лучший способ гарантировать целостность файловой системы, в то же время это и самый медленный режим, так как в нем через журнал проходят все данные.

- **Дефрагментация "на лету"**

Хотя ext4 включает в себя возможности уменьшения фрагментации внутри файловой системы (экстенты для выделения последовательных блоков), все же при длительной жизни файловой системы некоторая фрагментация неизбежна. Поэтому для улучшения производительности существует инструмент, который на лету дефрагментирует как файловую систему, так и отдельные файлы. Дефрагментатор - это простой инструмент, который копирует (фрагментированные) файлы в новый дескриптор ext4, указывающий на непрерывные экстенты.

Другим результатом дефрагментации на лету является уменьшение времени проверки файловой системы. (fsck). Ext4 помечает неиспользуемые группы блоков в таблице индексных дескрипторов, что позволяет процессу (fsck) полностью их пропускать и ускоряет тем самым процедуру проверки. Поэтому, когда операционная система решит проверить файловую систему после внутреннего повреждения (которые неизбежно будут происходить по мере увеличения размера файловой системы и ее распределенности), благодаря архитектуре ext4 это можно будет сделать быстро и надежно

- В ext4 представлен механизм "пространственной" (extent) записи файлов. Новая информация добавляется на диск по определённому алгоритму, который специально уменьшает фрагментацию и повышает производительность. А также предполагается механизм дефрагментации, чего не было в предыдущих ext2 и ext3.
- Для более точной работы с временными атрибутами файлов - время создания, модификации - используются наносекундные временные отметки

(timestamps). Максимально возможное время увеличено до 25 апреля 2514 года, против 18 января 2038 года у Ext3.