ОГЛАВЛЕНИЕ

1 Файловая система ext2

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

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

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

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

1.1 *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 для привилегированных пользователей **root**. Такой подход используется в целях безопасности, чтобы процессы, запущенные с привилеги-

ями 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 числа перемещений головок жёсткого диска, при чтении больших массивов данных блоки объединяются в **блочные группы.**

1.2 *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 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 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.

1.3 *Индексные дескрипторы*

Индексный дескриптор, или **inode** (information node) - это специальная структура данных, 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.

1.4 *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 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 утилиты **mount**, или 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**

3 Файловая система ext4

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

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

Глубина подкаталогов в **ext4** была увеличена с **32 КБ** до фактически неограниченной. Это может показаться избыточным, но 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.

3.1 *Экстенты*

Одним из главных недостатков системы **ext3** был ее метод выделения места на дисках. Дисковые ресурсы для файлов выделялись с помощью битовых карт - 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 фрагментированных файлов. Также узлы имеют механизмы самопроверки для защиты от повреждений файловой системы.

3.2 *Производительность*

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

- **Предварительное выделение на файловом уровне**

Некоторые приложения, например, базы данных, предполагают, 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 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 состоит в выделении физических блоков памяти до того, пока они действительно будут записаны. Это похоже на предварительное выделение, но в этом случае задачу система выполняет автоматически. Но если размер файла известен заранее, лучше применять предварительное выделение.

- **Выделение блоков памяти группами**

В **ext3** каждый блок выделяется по отдельности. Поэтому для последовательных данных выделенные блоки могут располагаться не последовательно. В **ext4** эта проблема решена за счет того, что выделение группы блоков происходит за один раз, поэтому фрагментирование данных меньше. 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 ликвидации этого явления 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). 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. При этом, копируются 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, благодаря архитектуре ext4 это можно будет сделать быстро и надежно