

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ им. Н.Э. БАУМАНА

Факультет «Информатика и системы управления»

Кафедра «Автоматизированные системы обработки информации и управления»



Сёмкин П.С., Сёмкин А.П.

Методические материалы к лабораторным работам

по дисциплине

«Операционные системы»

(кафедра СГНЗ)

Лабораторная работа № 6

«ОС Ubuntu. Администрирование файловых систем ext»

Москва

2023 г.

ОГЛАВЛЕНИЕ

1 ЦЕЛЬ РАБОТЫ	4
2 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ.....	4
2.1 Файловые системы ext.....	4
2.1.1 Блоки	4
2.1.2 Блочные группы	5
2.1.3 Индексные дескрипторы	7
2.1.4 Суперблок	8
2.1.5 Файловая система ext3.....	8
2.1.6 Файловая система ext4.....	10
2.1.7 Экстенты	10
2.1.8 Производительность	11
2.2 Создание разделов и файловых систем на жёстких дисках	13
2.2.1 Разбиение жёстких дисков на разделы	13
2.2.2 Создание разделов и файлов подкачки.....	14
2.2.3 Создание файловой системы в разделе диска.....	15
2.2.4 Просмотр и изменение атрибутов файловых систем	15
2.2.5 Создание виртуальной файловой системы.....	15
2.2.6 Монтирование файловых систем	16
2.2.7 Монтирование файловых систем из файла fstab	16
2.2.8 Монтирование файловых систем с помощью команды mount.....	17
2.2.9 Демонтирование файловых систем.....	18
2.2.10 Проверка файловой системы	18
3 ВЫПОЛНЕНИЕ РАБОТЫ	19
3.1 Задание	19
3.2 Порядок выполнения работы	19
3.2.1 Создание жёстких дисков и разделов диска.....	19
3.2.2 Создание раздела подкачки.....	20
3.2.3 Создание и монтирование файловой системы в разделе sdb1	20
3.2.4 Создание и монтирование файловой системы в разделе sdb5	20
3.2.5 Создание виртуальной файловой системы.....	21
3.2.6 Создание файла подкачки	21
3.2.7 Редактирование файле /etc/fstab.	22
4 КОНТРОЛЬНЫЕ ВОПРОСЫ	22
5 ЛИТЕРАТУРА.....	22
6 ПРИЛОЖЕНИЕ	23
6.1 Работа с дисками и разделами	23
6.1.1 Утилита fdisk	23

Операционные системы Лаб. работа №6(ОС Ubuntu. Администрирование файловых систем ext)	3
6.1.2 Утилита parted	23
6.1.3 Метки разделов	23
6.2 Создание раздела и файла подкачки	23
6.2.1 Раздел подкачки	23
6.2.2 Файл подкачки.....	23
6.2.3 Использование разделов и файлов подкачки	24
6.3 Создание файловой системы	24
6.3.1 Создание файловой системы в разделе диска.....	24
6.3.2 Создание виртуальной файловой системы.....	24
6.3.3 Просмотр и изменение атрибутов файловой системы.....	25
6.3.4 Монтирование файловой системы	25
6.3.5 Демонтирование файловой системы.....	26
6.3.6 Проверка файловой системы	26

1 Цель работы

Целью работы является знакомство с физической организацией файловых систем Linux и практическое освоение команд создания дисковых разделов, форматирования и монтирования файловых систем ext ОС Ubuntu.

2 Теоретическая часть

2.1 Файловые системы ext

После своего появления в 1993 году, файловая система **ext** стала самой распространенной файловой системой ОС Linux.

Целью создания файловой системы (название первой версии - **ext2**, second extended file system, **ext2fs**) являлось обеспечение высокой производительности и устойчивости файловой системы, а также поддержка дополнительных возможностей.

В файловой системе **ext** можно выделить следующие элементы:

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

2.1.1 Блоки

Всё пространство раздела диска при форматировании файловой системы разбивается на блоки фиксированного размера, кратные размеру сектора: **1024, 2048, 4096** или **8192** байт. Размер блока указывается при создании файловой системы или принимается равным по умолчанию.

Все блоки имеют порядковые номера. По умолчанию, во время форматирования, 5% блоков резервируются для привилегированного пользователя **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 числа перемещений головок жёсткого диска, при чтении больших массивов данных блоки объединяются в **блочные группы**.

2.1.2 Блочные группы

Все блоки раздела **ext** 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 дескрипторов 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 дескрипторов 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 дескрипторов (inode table)**, которая содержит записи для каждого inode в блочной группе.

При форматировании файловой системы, каждой блочной группе в **ext** назначается фиксированное количество индексных дескрипторов. Общее число inode в системе зависит от количества байтов на один индексный дескриптор, задаваемого во время форматирования раздела.

Поскольку размер таблицы индексных дескрипторов является фиксированным, единственный способ увеличения количества индексных дескрипторов в файловой системе **ext** заключается в увеличении размера файловой системы.

Объекты inode в таблице индексных дескрипторов 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 дескрипторов (inode allocation bitmap)**, которые позволяют отслеживать их использование внутри блочной группы. Каждый бит битовой карты соответствует определенной записи в таблице индексных дескрипторов группы.

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

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

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

Когда необходимость в индексном дескрипторе 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 дескрипторов 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.

2.1.3 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 **ext (ext 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.

2.1.4 Суперблок

Суперблок - основной элемент файловой системы **ext**. 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.

2.1.5 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 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

2.1.6 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 В-дереву. Поэтому, несмотря на гораздо больший размер, поиск в **ext4** работает очень быстро.

2.1.7 Экстенты

Одним из главных недостатков системы **ext3** - ее метод выделения места на дисках. Дисковые ресурсы для файлов выделялись с помощью битовых карт - способ, 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 экстендах 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 экстенда (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 дескриптор (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.

2.1.8 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 фрагментация 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 (утилита 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 проверяет файловую систему после внутреннего повреждения, благодаря архитектуре ext4 это выполняется быстро и надежно

2.2 Создание разделов и файловых систем на жёстких дисках

2.2.1 Разбиение жёстких дисков на разделы

Для создания файловых систем пространство жёсткого диска должно быть разделено на независимые области, называемые **разделами диска**

Разбиение диска на разделы производится при установке операционной системы и при подключении нового диска.

Чтобы создать на вновь подключённом жёстком диске файловую систему, необходимо создать на новом диске разделы (по крайней мере один) и затем в каждом из разделов создать файловую систему.

Создать разделы диска можно:

1. С использованием таблицы разделов **PT** (partition table) и **MBR**(Master Boot Record).

Максимальное число разделов –**четыре**.

Разделы: **первичные** (primary partition) – до четырех разделов на диске, **расширенный** (extended partition) – один раздел на диске.

Разделы содержат **логические диски** (logical disk), на которых создаются файловые системы.

Каждый **первичный раздел** содержит **один логический диск**.

Расширенный раздел может содержать **несколько логических дисков**.

2. С использованием таблицы разделов **GPT** (GUID Partition Table).

Допускает неограниченное количество разделов. Лимит устанавливает операционная система (ядро Linux поддерживает до 256 разделов).

Создание разделов и логических дисков выполняется с помощью команд:

1. **fdisk** (при использовании таблицы разделов **PT** и ограничении размеров раздела величиной 2ТБ)

2. **parted** (которая может работать с таблицами разделов **PT** и **GPT**).

После подключения нового диска нет необходимости перенастраивать ядро ОС, если диски такого типа в системе уже есть. Диск будет обнаружен как доступное устройство **sd**, например, **/dev/sdb** , **/dev/sdc**.

2.2.2 Создание разделов и файлов подкачки

Разделы подкачки предназначены для хранения страниц процессов, вытесненных из оперативной памяти.

Раздел подкачки может быть создан при установке Linux, либо его можно создать позднее с помощью команды **mkswap**.

Раздел подкачки можно создать либо в обычном **дисковом разделе**, либо в **файле**, отформатированном как раздел подкачки.

После создания раздела подкачки или файла подкачки, необходимо, используя команды **swapon** подключить к системе данную область подкачки.

Команду **swapon** можно использовать для просмотра списка файлов и разделов подкачки:

Чтобы прекратить использование области подкачки, необходимо выполнить команду **swapoff**:

Области подкачки имеют разные приоритеты. ОС будет в первую очередь ис-

пользовать области подкачки с высоким приоритетом, а затем те, которые имеют более низкий приоритет. Области с одинаковым приоритетом используются попеременно.

2.2.3 Создание файловой системы в разделе диска

Программные пакеты для работы с файловыми системами Ubuntu:

util-linux (включает в себя команду **mkfs** и другие приложения общего назначения)

e2fsprogs (включает в себя специальные приложения файловых систем **ext2/ext3**).

Основные приложения устанавливаются вместе с Ubuntu.

*Примеры использования команды **mkfs** для создания файловых систем приведены в приложении*

2.2.4 Просмотр и изменение атрибутов файловых систем

Используя команду **tune2fs** или **dumpe2fs**, можно просматривать атрибуты файловых систем **ext2**, **ext3** и **ext4**.

Команду **tune2fs** также можно применять для изменения атрибутов файловых систем.

2.2.5 Создание виртуальной файловой системы

Виртуальная файловая система –это файловая система, располагающаяся в файле существующей файловой системы.

Пример создания файла образа диска размером 1 Гбайт и его форматирование как файловой системы с последующим монтированием для обеспечения доступа к данным в файловой системе приведён в приложении

Команда **dd** создает пустой файл образа диска объемом 2 048 000 блоков (примерно 1 Гбайт).

Команда **mkfs** позволяет создать файловую систему любого типа на выбор. Файл не является файлом специального блочного устройства, как это имеет место при форматировании дисковых разделов, поэтому **mkfs** предупредит, прежде чем приступит к созданию файловой системы.

Выполнив монтирование виртуальной файловой системы, можно осуществлять к ней доступ как к любой файловой системе.

Закончив работать с виртуальной файловой системой, можно её демонтировать.

Демонтировав виртуальную файловую систему, её можно перенести в другую систему.

Если файловая система больше не нужна, можно просто удалить соответствующий файл.

2.2.6 Монтирование файловых систем

Файловая система ОС Ubuntu организована в виде единого дерева с одной исходной вершиной, которая называется корнем (записывается: "/");

Монтирование представляет собой операцию присоединения файловой системы(ветви) к единому дереву файловой системы операционной системы и выполняется автоматически в момент загрузки (используется информация файла **/etc/fstab**) или по команде **mount**.

Точка монтирования - это каталог, в котором появляется содержимое файловой системы логического или физического раздела после его монтирования в данную точку.

Для того, чтобы ветвь будет присоединена к дереву, должны существовать и файловая система, и точка монтирования.

Если присоединяется новый носитель к существующей файловой системе, то на этом носителе уже должны быть определены разделы и содержаться файловые система известного операционной системе типа.

После монтирования все каталоги нового раздела будут доступны в качестве подкаталогов точки монтирования.

2.2.7 Монтирование файловых систем из файла fstab

Файл **/etc/fstab** содержит информацию о смонтированных файловых системах.

Описание полей в записи о монтируемой файловой системе в файле **/etc/fstab**.

1- имя устройства, которое представляет файловую систему. Первоначально это поле содержало название устройства раздела для монтирования (например, /dev/sda1). Теперь оно также может содержать LABEL, универсальный уникальный идентификатор (UUID) или удаленную файловую систему (NFS либо CIFS) вместо имени устройства.

2 - точка монтирования в файловой системе.

3 - тип файловой системы.

4 - параметры команды mount. К примерам параметров команды mount относятся **noauto** (для предотвращения монтирования файловой системы во время загрузки) и **ro** (для монтирования файловой системы только для чтения). Чтобы дать любому пользователю возможность монтировать файловую систему, можно добавить параметр user или owner в это поле. Параметры должны быть разделены запятыми (Остальные параметры можно посмотреть по команде **mount -o**

5 - использовать ли dump в отношении файловой системы. Это поле существенно, если выполняется резервное копирование с помощью dump. **Значение 1 говорит о необходимости резервного копирования файловой системы, а значение 0 подразумевает обратное.**

6 - необходима ли проверка файловой системы. Значение в этом поле является индикатором того, нужна ли проверка файловой системы с помощью **fsck**. Значение 0 свидетельствует о необходимости проверки файловой системы. Значение 1 будет индикатором, что файловую систему нужно проверить в первую очередь (такой подход применяется для корневой файловой системы). Значение 2 говорит о том, что файловая система может быть проверена в любой момент после проверки корневой файловой системы.

Можно создавать записи в файле **/etc/fstab** для разделов любого жесткого диска или съемного носителя. В удаленных файловых системах (NFS, Samba и др.) также могут содержаться записи в файле для автоматического монтирования этих файловых систем во время загрузки .

2.2.8 Монтирование файловых систем с помощью команды mount

Команда **mount** используется для просмотра смонтированных файловых

систем, а также непосредственного монтирования любых локальных (на жестком диске, USB-диске, CD, DVD и т. д.) или удаленных (NFS, Samba и т. п.) файловых систем.

Можно указать параметры команды **mount**, добавив **-o** и список параметров, разделенных запятыми. Эти параметры аналогичны тем, которые можно добавить в поле 4 файла **/etc/fstab**.

По умолчанию разделы монтируются с доступом с правом чтения/записи. Можно явно указать, следует ли **монтировать файловую систему с правом чтения/записи (rw) или только для чтения (ro)**:

Посмотреть, какие разделы в какие точки монтирования присоединены, можно командой **df -h**.

2.2.9 Демонтирование файловых систем

Чтобы демонтировать файловую систему, необходимо использовать команду **umount**.

Можно демонтировать файловую систему, используя имя устройства или точку монтирования. Предпочтительнее демонтажное с использованием точки монтирования во избежание путаницы при связанном монтировании (одно устройство, несколько точек монтирования).

Если устройство окажется занятым, то демонтажное потерпит неудачу.

При отложенном демонтаже файловая система демонтируется из дерева сразу же, но с ожиданием, пока устройство не освободится, прежде чем все убирать.

2.2.10 Проверка файловой системы

В Linux можно использовать команду **badblocks** для проверки устройства на предмет поврежденных блоков на физическом уровне или команду **fsck** для проверки файловой системы на наличие ошибок на логическом уровне.

По умолчанию **badblocks** производит безопасное тестирование блоков, при

котором осуществляется только чтение. Можно выполнять безопасное тестирование с чтением/записью. Это самое продолжительное по времени тестирование, однако и лучшее, которое можно произвести, не уничтожив данные на устройстве.

Для тестирования диска, не содержащего данных, которые необходимо сохранить, можно использовать команду, производящую **более быстрое тестирование с чтением/записью, при котором данные уничтожаются:**

Файловую систему ext можно проверить с помощью команды **fsck**.

3 Выполнение работы

3.1 Задание

1. Присоединить к виртуальной машине динамический виртуальный жёсткий диск
2. Используя утилиты операционной системы, создать на новом диске разделы и логические диски.
3. Создать раздел подкачки и подключить его к операционной системе
4. Создать в разделах файловые системы и выполнить монтирование файловых систем.
5. Создать виртуальную файловую систему и монтировать её
6. Создать файл подкачки и подключить его к операционной системе
7. Добавить в файл fstab информацию для постоянного монтирования файловых систем

3.2 Порядок выполнения работы

1. Войти в систему под учётной записью **user2** (Пароль **Stud-101**).
2. Запустить программу виртуализации **Oracle VM VirtualBox**.

3.2.1 Создание жёстких дисков и разделов диска

1. Для виртуальной машины **Ubuntu_XX** к контроллеру **SATA** добавить динамический виртуальный жёсткий диск (тип файла виртуализации **VDI**, динамический виртуальный жёсткий диск) **my_disk** размером **8 Gb**
2. Запустить виртуальную машину **Ubuntu_XX**

3. Войти в систему под учётной записью **admin_kaf**.
4. С помощью утилиты **fdisk** получить информацию обо всех подключенных дисках и разделах жестких дисков
5. Используя интерактивный сеанс утилиты **fdisk**, создать на новом диске **my_disk(sdb)** разделы:
раздел 1 размером **3 Gb** –первичный раздел
раздел 2 размером **5 Gb** – расширенный раздел, содержащий логические диски **2 Gb** и **3Gb**
6. Просмотреть информацию о созданных разделах
7. С помощью утилиты **fdisk** просмотреть информацию обо всех дисках и разделах жестких дисков

3.2.2 Создание раздела подкачки

1. Форматировать раздел **sdb6** как раздел подкачки
2. Проверить раздел подкачки на предмет повреждённых блоков
3. Подключить раздел подкачки к операционной системе
4. Просмотреть информацию обо всех используемых разделах и файлах подкачки

3.2.3 Создание и монтирование файловой системы в разделе sdb1

1. Создать в разделе **sdb1** файловую систему **ext4**
2. Создать точку монтирования созданной файловой системы в каталоге **/mnt**.
Имя точки монтирования **fs_share**
3. Сменить владельца и группу каталога **fs_share** на **admin_kaf**
4. Установить права доступа к каталогу **fs_share** (полный доступ для всех)
5. Смонтировать файловую систему раздела **sdb1** в точку монтирования **/mnt/fs_share**
6. Просмотреть информацию о смонтированных файловых системах

3.2.4 Создание и монтирование файловой системы в разделе sdb5

1. Создать в разделе **sdb5** файловую систему **ext4**

2. Создать точку монтирования файловой системы в домашнем каталоге пользователя **admin_kaf**. Имя точки монтирования **fs_admin**.
3. Сменить владельца и группу каталога **fs_admin** на **admin_kaf**
4. Назначить права доступа к **fs_admin** (полный доступ только для **admin_kaf**)
5. Смонтировать файловую систему раздела **sdb5** в точку монтирования **fs_admin**.
6. Просмотреть информацию о смонтированных файловых системах

3.2.5 Создание виртуальной файловой системы

1. Создать в файловой системе **fs_admin** файл **sd_virt**, заполненный нулями, размером **1 Gb**
2. Создать в файле **sd_virt** виртуальную файловую систему
3. Создать в домашнем каталоге пользователя **admin_kaf** точку монтирования виртуальной файловой системы
4. Монтировать виртуальную файловую систему в точку монтирования
5. Просмотреть информацию о смонтированных файловых системах

3.2.6 Создание файла подкачки

1. Создать в файловой системе **fs_share** файл **my_swap** размером 1 Гбайт
2. Блокировать права доступа к файлу **my_swap** (права **600**)
3. Форматировать файл **my_swap** как файла подкачки.
4. Подключить файл **my_swap** как файл подкачки
5. Присвоить разделу подкачки **sdb6** высший приоритет
6. Просмотреть информацию обо всех используемых разделах и файлах подкачки

3.2.7 Редактирование файле **/etc/fstab**.

1. С помощью текстового редактора отредактировать файл **/etc/fstab** для постоянного монтирования файловых систем **fs_share, fs_admin, fs_virt**
2. Перезагрузить систему и просмотреть информацию о смонтированных файловых системах

4 Контрольные вопросы

1. С какой целью создаются разделы жёсткого диска и логические диски?
2. Что такое блок и блочная группа?
3. С какой целью создаются разделы и файлы подкачки?
4. Для чего предназначено форматирование файловой системы?
5. Что такое монтирование файловой системы?
6. Как и с какой целью создаётся виртуальная файловая система?

5 ЛИТЕРАТУРА

1. Сёмкин П.С., Аксёнов А.Н. Файловые системы. Логическая организация и физическая реализация. Сборник учебно-методических работ кафедры «Системы обработки информации и управления» (бакалавры). Учебное пособие. Вып. 1./Под ред: В.М. Черненко. –М: «АртКом», 2013. – стр. 95-120
2. Сёмкин П.С., Семкин А.П. Файловые системы операционных систем Windows и Unix. Сборник учебно-методических работ кафедры «Системы обработки информации и управления» (бакалавры). Учебное пособие. Вып. 2./Под ред. В.М. Чёрненко. –М: «АртКом», 2014. – стр. 160-189
3. Негус К. Ubuntu и Debian Linux для продвинутых. 2-е изд. – СПб.: Питер,2014. -384 с.: ил.

6 Приложение

6.1 Работа с дисками и разделами

6.1.1 Утилита fdisk

\$ sudo fdisk -l *показать сведения о разделах всех дисков*

\$ sudo fdisk -l /dev/sdb *показать сведения о разделах диска **sdb***

\$ sudo fdisk /dev/sdb *интерактивный сеанс **fdisk** для работы с конкретным диском*

6.1.2 Утилита parted

\$ sudo parted /dev/sdb print – *отображение информации о разделах диска*

\$ sudo parted /dev/sdc *- интерактивный сеанс*

\$ sudo mkpart logical ext4 1MB 2 GB – *создание нового раздела*

\$ sudo mkpart – *ввод параметров в интерактивном режиме*

6.1.3 Метки разделов

\$ sudo e2label /dev/sdb1 – *просмотр метки раздела*

\$ sudo e2label /dev/sdb1 mypart1 – *задание метки раздела*

\$ sudo findfs LABEL=mypart1 – *нахождение раздела по метке*

6.2 Создание раздела и файла подкачки

6.2.1 Раздел подкачки

\$ sudo mkswap /dev/sdb1 *форматировать **sdb1** как раздел подкачки*

\$ sudo mkswap -c /dev/sdb1 *проверки области подкачки на предмет поврежденных блоков*

6.2.2 Файл подкачки

\$ sudo dd if=/dev/zero of=/myfs1/swapfile count=2048000 *- создание файла размером 1 Гбайт*

\$ sudo chmod 600 /myfs1/swapfile *блокирование прав доступа к данному файлу*

\$ sudo mkswap /myfs1/swapfile - форматирование файла **/mnt/swapfile** как файла подкачки.

6.2.3 Использование разделов и файлов подкачки

\$ sudo swapon -v /dev/sdb1 использовать **/dev/sdb1** как раздел подкачки

\$ sudo swapon -v /myfs1/swapfile использовать **/myfs1/swapfile** как файл подкачки

\$ swapon -s показать все используемые файлы подкачки и разделы подкачки

\$ sudo swapoff -v /myfs1/swapfile прекратить использование области подкачки

\$ sudo swapon -v -p 1 /dev/sdb1 присвоить разделу подкачки **sdb1** высший приоритет

6.3 Создание файловой системы

6.3.1 Создание файловой системы в разделе диска

\$ sudo mkfs -t ext4 /dev/sdb1 создать файловую систему **ext4** в разделе **sdb1**

\$ sudo mkfs -t ext4 -v -c /dev/sdb1 сгенерировать подробный вывод/сканировать на предмет поврежденных блоков

\$ sudo mkfs.ext4 -c /dev/sdb1 дает тот же результат, что и предыдущая команда

Если надо добавить метку для нового раздела при создании файловой системы, необходимо использовать параметр **-L**:

\$ sudo mkfs.ext4 -c -L mypartition /dev/sdb1 добавить метку **mypartition**

6.3.2 Создание виртуальной файловой системы

\$ dd if=/dev/zero of=/mnt/fs_share/my_vf1 count=2048000 создать заполненный нулями файл размером 1 Гбайт

\$ du -sh mnt/myfs1/vfile1 проверить размер виртуального файла

\$ mkfs -t ext4 mnt/myfs1/vfile1 создать файловую систему в vfile1

\$ sudo mkdir mnt/myvfs1 создать точку монтирования

\$ sudo mount mnt/myfs1/vfile1 /mnt/myvfs1 монтировать виртуальную файловую систему

6.3.3 Просмотр и изменение атрибутов файловой системы

\$ sudo tune2fs -l /dev/sda1 | less показать настраиваемые атрибуты файловой системы

\$ sudo dumpe2fs -h /dev/sda1 генерирует тот же вывод, что и **tune2fs**

\$ sudo tune2fs -c 31 /dev/sda1 задает количество монтирований, по достижении которого будет запущена принудительная проверка

\$ sudo tune2fs -c -1 /dev/sda1 деактивизировать проверку

\$ sudo tune2fs -l 10 /dev/sda1 запустить проверку, зависящую от времени(через 10 дней)

\$ sudo tune2fs -j /dev/sda1 обеспечить журналирование преобразования **ext2** в **ext3**

6.3.4 Монтирование файловой системы

\$ sudo mkdir /mnt/myfs1 создание точки монтирования

\$ mount показать смонтированные фс

\$ mount -t ext4 показать смонтированные фс определённого типа

\$ mount -t ext4 -l показать метки

\$ sudo mount /dev/sdb2 /mnt/myfs1 монтировать фс

\$ sudo mount -v /dev/sdb2 /mnt/myfs1 показывать подробную информацию

\$ sudo mount -v -t ext4 /dev/sdb1 /mnt/myfs1 – монтировать фс определённого типа

\$ sudo mount -vl -t dev/sdb1 /mnt/myfs1 – монтировать и показать метку

\$ sudo mount -v -t ext4 -o rw dev/sdb1 /mnt/myfs1 – монтировать с правом чтения /записи

\$ sudo mount -v -t ext4 -o ro dev/sdb1 /mnt/myfs1 – монтировать только для чтения

\$ sudo mount -vl -t ext4 dev/sdb1 /mnt/myfs1 – монтировать и показать метку

\$ sudo mount mnt/myfs1/vfile1 /mnt/myvfs1 монтировать виртуальную файловую систему

6.3.5 Демонтирование файловой системы

\$ sudo umount -v /dev/sdb1 демонтировать с использованием имени устройства

\$ sudo umount -v /mnt/mymount/ демонтировать с использованием точки монтирования.

\$ sudo umount -vi /mnt/mymount/ выполнить отложенное демонтажное

\$ sudo eject /dev/cdrom демонтировать и извлечь CD

\$ sudo umount /mnt/myvfs1 демонтировать виртуальную файловую систему

6.3.6 Проверка файловой системы

\$ sudo badblocks /dev/sdb1 физически проверить диск на предмет поврежденных блоков.

\$ sudo badblocks -vsn /dev/sdb1 выполнить безопасную проверку на предмет поврежденных блоков

\$ sudo badblocks -vsw /dev/sdal выполнить проверку на предмет поврежденных блоков с уничтожением данных

\$ sudo fsck /dev/sdb1 проверка файловой системы