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 указания по выполнению лабораторных работ
по дисциплине
«Операционные системы»

Лабораторная работа № 11

«Администрирование файловых систем ОС Ubuntu»

Москва
2017 г.

ОГЛАВЛЕНИЕ

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.

Продолжительность работы – 2 часа.

2 Теоретическая часть

2.1 *Файловая система ext2*

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

2.1.1 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.**

2.1.2 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, 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.

2.1.3 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.

2.1.4 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.

2.2 *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 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**

2.3 *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.

2.3.1 Экстенты

Одним из главных недостатков системы ext3 был ее метод выделения места на дисках. Дисковые ресурсы для файлов выделялись с помощью битовых карт 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 после-

довательных 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.

2.3.2 Производительность

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

- **Предварительное выделение на файловом уровне**

Некоторые приложения, например, базы данных, 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.

2.4 *Создание разделов и файловых систем на жёстких дисках*

2.4.1 Разбиение жёстких дисков на разделы

Разбиение диска на разделы производится при установке 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 новую файловую систему.

Создание разделов на диске выполняется с помощью команд **fdisk**(при использовании 32-битной таблицы разделов **PT**(Partition Table) и ограничении размеров раздела величиной 2ТБ) и **parted(**которая может работать также и таблицами разделов GPT(GUID Partition Table – таблица разделов GUID).

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

Пример использования команды **fdisk** для просмотра информации о дисках и дисковых разделах и для создания разделов на подключаемых дисках:

$ sudo fdisk –l *показать сведения о разделах всех дисков*

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

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

Пример использования команды **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*

2.4.2 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.

Примеры:

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

Для проверки области подкачки на предмет поврежденных блоков используется команду mkswap с параметром **-c:**

$ sudo mkswap -c /dev/sdb1

Кроме дискового раздела, возможно создание области подкачки в файле:

$ sudo dd if=/dev/zero of=/myfs1/swapfile count=65536 - *создание файла размером 32 Мбайт*

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

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

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

$ sudo swapon –v /dev/sdal *использовать /dev/sda1 как раздел подкачки*

$ sudo swapon –v /mnt/swapfile *Использовать /mnt/swapfile как файл подкачки*

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

$ swapon -s *показать все используемые файлы подкачки и разделы подкачки.*

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

$ sudo swapoff -v /mnt/swapfile

Области подкачки имеют разные приоритеты. ОС будет в первую очередь использовать области подкачки с высоким приоритетом, а затем те, которые имеют более низкий приоритет. Области с одинаковым приоритетом используются попеременно.

Можно указать приоритет для области подкачки, воспользовавшись параметром -p:

$ sudo swapon -v -p 1 /dev/sdal *Присвоить sddl высший приоритет*

2.4.3 Создание файловой системы в разделе диска

Программные пакеты для работы с файловыми системами Ubuntu:

util-linux (включает в себя команду mkf s 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** (сначала обязательно указывать параметр -t):

$ sudo mkfs -t ext4 /dev/sdbl *Создать файловую систему ext4 в разделе sbal*

$ **sudo mkfs -t ext4 -v -c /dev/sdbl** *Сгенерировать подробный вывод/сканировать на предмет поврежденных блоков*

$ **sudo mkfs.ext4 -c /dev/sdbl** *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/sdbl** *Добавить метку mypartition*

2.4.4 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.

$ sudo tune2fs -l /dev/sda1 | less *Показать настраиваемые атрибуты файловой системы*

$ **sudo dumpe2fs -h /dev/sdal** *Генерирует тот же вывод, что* и ***tune2fs***

Чтобы изменить настройки существующей файловой системы **ext2, ext3 или**

ext4, можно воспользоваться командой **tune2fs**.

$ sudo tune2fs -c 31 /dev/sdal *Задает количество монтирований, по достижении которого будет запущена принудительная проверка*

Чтобы обеспечить запуск принудительной проверки файловой системы через определенный интервал времени, а не по достижении определенного количества монтирований, необходимо деактивизировать соответствующую проверку, задав отрицательное значение в виде единицы с минусом (-1):

$ sudo tune2fs -c -1 /dev/sdal

Используя параметр -1, можно активизировать проверку, зависящую от времени:

$ sudo tune2fs -1 10 /dev/sdal *Запустить проверку через 10 дней*

Всегда должна быть активизирована либо проверка, запускаемая по достижении определенного количества монтирований, либо проверка, зависящая от времени.

Используя параметр -j, можно преобразовать файловую систему ext2 в ext3 (задействовав при этом журналирование):

$ sudo tune2fs -j /dev/sdal *Обеспечить журналирование преобразования ext2 в ext3*

2.4.5 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:

$ dd if=/dev/zero of=/mnt/myfs1/vfile1 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*

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

Команда **mkfs** позволяет создать файловую систему любого типа на ваш выбор (в данном случае создается ext4). 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.

2.4.6 Монтирование файловых систем

Общим понятием для всех файловых систем, соответствующих POSIX, является **точка монтирования**. Это каталог, в котором появляется содержимое логического или физического раздела файловой системы после его монтирования.

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

До того, как ветвь будет присоединена к файловой системе, должны существовать и файловая система, и точка монтирования.

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

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

2.4.7 Монтирование файловых систем из файла fstab

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

Описание полей в файле **/etc/fstab**.

1- имя устройства, которое представляет файловую систему. Первоначально это поле содержало название устройства раздела для монтирования (например, /dev/sdal). 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 /etc/fstab для автоматического монтирования этих файловых систем во время загрузки или позднее ***вручную.***

2.4.8 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):**

Примеры выполнения команды mount:

$ sudo mount – *показать смонтированные системы*

$ sudo mount ext4 – *показать смонтированные системы определённого типа*

$ sudo mount ext4 –l – *показать метки разделов*

$ sudo mount /dev/sdb1 /mnt/myfs – *монтировать файловую систему*

$ sudo mount –v /dev/sdb1 /mnt/myfs – *показать подробную информацию*

$ sudo mount –bind –v /dev/myfs /mnt/myfs1 – *файловая система доступна в двух точках монтирования*

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

2.4.9 Демонтирование файловых систем

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

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

$ sudo umount -v /dev/sdbl *Демонтировать с использованием имени устройства*

$ sudo umount -v /mnt/mymount/ *Демонтировать с использованием точки монтирования.*

Если устройство окажется занятым, то демонтирование потерпит неудачу.

Выполнение отложенного демонтирования**:**

$ sudo umount -vi /mnt/mymount/ *Выполнить отложенное демонтирование*

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

Демонтирование съемных носителей также можно выполнять с помощью eject.

$ sudo eject /dev/cdrom *Демонтировать и извлечь CD*

2.4.10 Проверка файловых систем

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

$ sudo badblocks /dev/sdbl *Физически проверить диск на предмет поврежденных блоков*.

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.

$ sudo badblocks -vsn /dev/sdbl *Выполнить безопасную проверку на предмет поврежденных блоков*

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

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

Файловую систему ext можно проверить с помощью команды fsck:

$ sudo fsck /dev/sdbl

3 Задание на выполнение работы

1. Войти в систему под учётной записью studXX(XX –индекс группы).
2. Запустить программу виртуализации Oracle VM VirtualBox.
3. Для виртуальной машины Ubuntu добавить 2 динамических виртуальных жёстких диска (тип файла виртуализации VDI, динамический виртуальный жёсткий диск) размером 8 Gb каждый. Имена дисков –Disk1, Disk2.
4. Запустить виртуальную машину Uduntu.
5. Используя интерактивный сеанс программы fdisk создать на новом диске Disk1 первичные разделы :
   раздел 1размером 3 Gb и раздел 2 до конца диска:
6. Создать в разделе **sdb1** раздел подкачки и подключить его к системе:
7. Создать в разделе **sdb2** файловую систему **ext4** и смонтировать файловую систему в точку монтирования **/mnt/myfs1**
   df –h – посмотреть информацию о смонтированных разделах
8. Создать в файловой системе **myfs1** файл **vfile1** размером 1 Gb, создать в данном файле виртуальную файловую систему и смонтировать её в точку монтирования **/mnt/myvfs1**
9. Создать в файловой системе **myfs1** файл подкачки **swapfile**
10. Используя интерактивный сеанс программы fdisk создать на новом диске Disk2 первичный раздел, включив в него всё пространство диска
11. Создать в разделе **sdc1** файловую систему **ext4** и смонтировать файловую систему в точку монтирования **/mnt/myfs2**

4 Контрольные вопросы

1. С какой целью создаются разделы жёсткого диска?

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. Черненького. –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. Чёрненького. –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 Основные команды администрирования файловых систем

6.1 *Работа с разделами*

sudo fdisk –l – просмотр разделов

sudo fdisk /dev/sdb начать интерактивный сеанс

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*

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

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

$ 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/swapfiIe как раздела подкачки.

$ sudo swapon –v /dev/sdbl 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/sdbl присвоить разделу подкачки sdbl высший приоритет

6.3 *Создание файловых систем в разделе диска*

sudo mkfs -t ext4 /dev/sdb1 создание файловой системы в разделе

sudo tune2fs dev/sdb1- показать настраиваемые атрибуты файловой системы

6.4 *Создание виртуальных файловых систем*

$ dd if=/dev/zero of=/mnt/myfs1/vfile1 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

6.5 *Монтирование файловых систем*

$ 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

6.6 *Демонтирование файловых систем*

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

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

$ sudo umount -vi /mnt/mymount/ выполнить отложенное демонтирование

$ sudo eject /dev/cdrom демонтировать и извлечь CD

$ sudo umount /mnt/myvfs1 демонтировать виртуальную файловую систему

6.7 *Просмотр и изменение атрибутов файловых систем*

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

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

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

$ sudo tune2fs -c -1 /dev/sdal деактивизировать проверку

$ sudo tune2fs -1 10 /dev/sdal *запустить проверку, зависящую от времени(через 10 дней)*

$ sudo tune2fs -j /dev/sdal обеспечить журналирование преобразования ext2 в ext3