Московский государственный технический университет им. Н.Э. Баумана

Кафедра «Автоматизированные системы обработки информации и управления»

Сёмкин П.С., Сёмкин А.П.

Методические материалы к лабораторным работам
по дисциплине
«Операционные системы (Второе образование)»

Лабораторная работа № 3

«ОС Alt Linux. Управление дисковой подсистемой. Администрирование файловых систем»

Москва

2024 г.

ОГЛАВЛЕНИЕ

1 Цель работы

Целью работы является получение навыков планирования пространства дисковой подсистемы ОС Alt Linux, создания дисковых разделов, форматирования и монтирования файловых систем.

2 Теоретическая часть

2.1 Планирование пространства дисковой подсистемы

Основными устройствами хранения информации в компьютерной системе являются жёсткие диски, образующие дисковую подсистему операционной системы.

На жёстких дисках хранятся как файлы и данные самой операционной системы, так и файлы и данные пользователей.

Надёжность хранения данных и эффективность доступа к ним возрастает, если разделять хранение данные разных типов, различающихся по характеру использования.

С этой целью всё доступное пространство на жёстком диске (или дисках) разделяется на независимые области, каждая из которых предназначена для хранения данных определённого типа.

Для организации таких областей предназначена технология разделения диска на **разделы**.

Поскольку разделы не зависят друг от друга, изменение содержимого одного раздела никак не сказывается на других. Одна из выгод такого подхода: в случае физического сбоя или повреждения данных, ошибки будут локализованы внутри того раздела, где произошёл сбой, и не затронут других разделов.

Разбиение на разделы может быть использовано и для оптимизации скорости доступа: скорость чтения и записи для большинства дисков выше в середине и ниже к концу и началу диска. Для этого разделы с данными, для которых важна скорость доступа, целесообразно располагать в середине диска.

Разделение диска на разделы необходимо и в том случае, если на одном физическом устройстве должны быть установлены несколько операционных систем.

2.2 Таблицы разделов диска

При разделении диска на разделы создаются таблицы разделов. В зависимости от технологии, эти таблицы носят название **PT** (Partition Table) или **GPT**(GUID Partition Table.)

2.2.1 Таблица разделов PT (Partition Table) MBR

PT - это стандарт таблицы разделов диска, связанный с загрузкой операционных систем с использованием системы **BIOS** и таблицы **MBR**

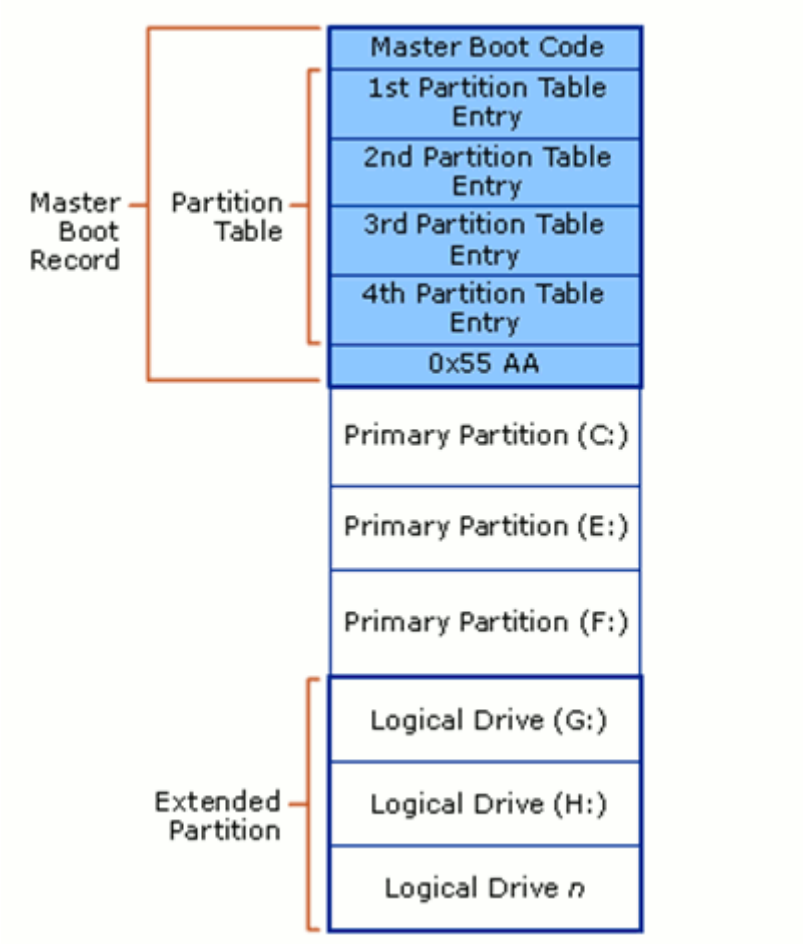
BIOS (Basic Input-Output System) - базовая система ввода-вывода. Это программы низкого уровня, хранящиеся на чипе материнской платы. Программы BIOS загружаются при включении компьютера, инициируют его аппаратные компоненты, ищут в таблице **MBR** (Master Boot Record) главную загрузочную запись (Master Boot Code), которая используется для запуска **программы-загрузчика** операционной системы. Для Linux там находится код инициализации **Grub.** Этот код используется только для инициализации основного загрузчика, расположенного в другой области на диске, а тот уже загружает ОС.

MBR находится в самом начале диска и занимает первые 512 байт.

MBR содержит таблицу разделов **PT** (Partition Table), которая содержит информацию, о том, какие разделы есть на этом устройстве.

Ограничение **MBR** в том, что диск может иметь только **четыре раздела**. Это связанно с ограниченным количеством памяти, выделенным под таблицу разделов.

Разделы могут быть первичными - **primary**(до четырёх разделов) , и также расширенный - **extended**(только один). Расширенный раздел может содержать несколько логических- **logical,** что позволяет создать необходимое количество разделов диска.



MBR использует 32-битную адресацию, что позволяет работать только с дисками размером до двух терабайт.

В MBR используется адресация, зависимая от геометрии диска. Адрес состоит с трех значений: номер цилиндра (Cylinder, **C)**, номер рабочей поверхности (Head, **H**), номер сектора (Sector, **S**) - (**CHS**) (например, **0,0,0**).

2.2.2 Таблица разделов GPT (GUID Partition Table)

GPT это технология управления разделами на жестком диске, являющаяся частью стандарта **EFI** (Extensible Firmware Interface), разработанного в Intel для замены **BIOS**.

Спецификация **UEFI** (Unified Extensible Firmware Interface) - унифицированный интерфейс расширяемой прошивки –промышленный стандарт, зависящий не только от Intel.

UEFI не поддерживает загрузку, которая поддерживается только **BIOS**. **UEFI** работает как с таблицами разделов, так и с файловыми системами.

Стандартно используемые версии **UEFI** имеют поддержку таблиц разделов **MBR** и **GPT**.

UEFI не выполняет никакой код из **MBR** даже если он есть. Вместо этого используется **специальный раздел** на жестком диске называемый "**EFI SYSTEM PARTITION**", на которой и располагаются файлы, которые необходимо запустить для загрузки.

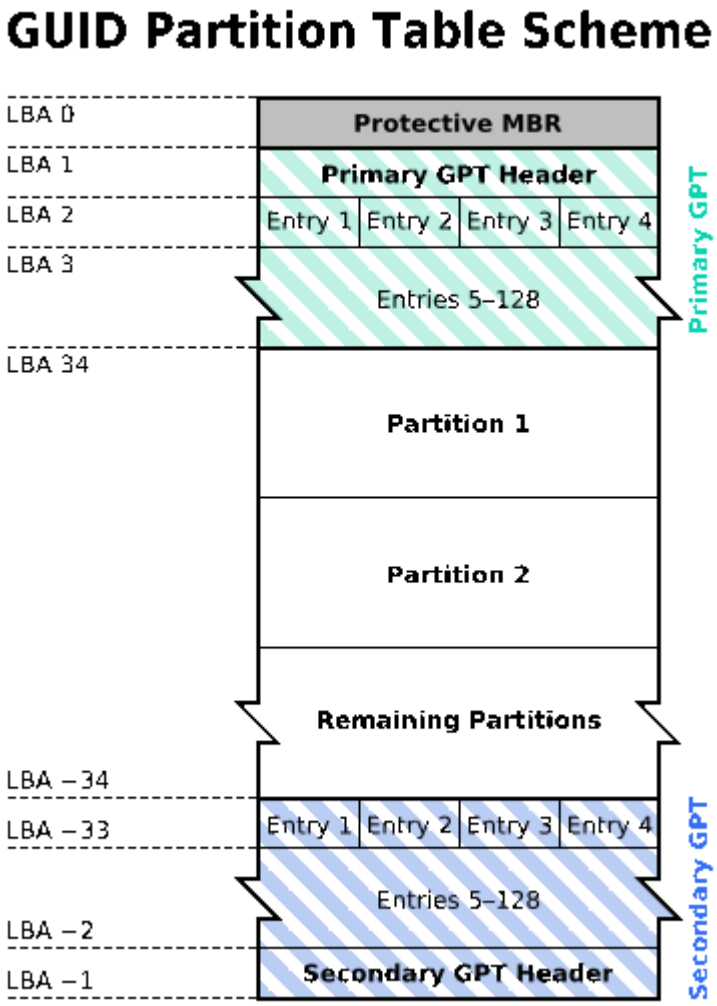
Первое отличие **GPT** - это использование другой адресации диска. В GPT используется адресация **LBA**. **LBA** (**Logical block addressing**) - стандартизованный механизм адресации и доступа к блоку данных на блочном устройстве (жёстком или оптическом диске, твердотельном накопителе), при котором нет необходимости учитывать специфику накопителя (например, геометрию жёсткого диска

Это блочная адресация, каждый блок имеет свой номер, например: **LBA1, LBA2, LBA3**, и так далее, причем адреса MBR автоматически транслируются в **LBA**, например **LBA1** будет иметь адрес **0,0,1** и так далее.

GPT не содержит кода загрузчика (загрузкой занимается **EFI**), здесь размещена только таблица разделов. В блоке **LBA0** находится **MBR**, это сделано для защиты от затирания **GPT** старыми утилитами работы с дисками, а уже с блока (LBA1) начинается сама GPT.

Под таблицу разделов резервируется **16 384** байт памяти, по **512** на блок (32 блока).

Первые разделы диска начнутся с блока **LBA34** (32+1MBR+1GPT).



Количество разделов не ограничено.

Ограничение на количество разделов устанавливается только операционной системой. Ядро **Linux** поддерживает до **256 разделов**.

Благодаря адресации **LBA**, **GPT** в отличии от **MBR** может создавать разделы до **9,4 ЭБ**.

Кроме того, служебная информация **GPT** дублирована, она размещается не только в начале диска, но и в конце. Таким образом во многих случаях при повреждении GPT может сработать автоматическое восстановление.

GPT поддерживает **юникод** поэтому можно задавать имена и атрибуты разделам. Имена могут быть заданы на любом поддерживаемом языке и можно обращаться к дискам по этим именам.

Для дисков используются глобальные уникальные идентификаторы **GUID (Globally Unique IDentifier)**, это одна из вариаций **UUID**, которые также можно также использоваться для идентификации дисков вместо имен.

UUID (Universally Unique identifier) — это 128 битный уникальный номер, стандартизированный Open Software Foundation. **UUID помогает идентифицировать разделы в системах Linux**. Он генерируется с помощью библиотеки libuuid (используемой **e2fsprogs)**, которая является частью **util-linux**. Доступна сразу после установки **Linux** начиная с версии ядра 2.15.1.

2.2.3 Поддержка таблиц разделов операционными системами

Ядро **Linux** включает поддержку как **PT** так и **GPT**

При установки операционной системы на дисках с **GPT** используется загрузчик **Grub2**.

Старая операционная система на диск с **GPT** может не установиться.

2.2.4 Сравнение PT и GPT

- **PT** поддерживает диски до **2 Тб**, **GPT** - до **9 Эб**
- **GPT** поддерживает **более четырех разделов**
- **GPT** использует **GUID** для идентификации дисков
- **GPT** использует систему адресации **LBA**, вместо **CHS**
- Служебная информация **GPT** дублируется в начале и конце диска
- **GPT** проверяет **контрольные суммы**, что позволяет **обнаружить модификацию таблицы разделов**
- GPT поддерживает Unicode, а, следовательно, имена на кириллице имена.

2.3 Разделы системного диска ОС ALT Linux

2.3.1 Необходимые разделы

Минимальное количество разделов, которые необходимы **ALT Linux** для работы- **два.**

Первый – для **области подкачки**, второй – для **корневой файловой системы**.

Область подкачки (swap space) - это пространство на диске, используемое системой для организации **виртуальной памяти**.

В Linux принято размещать область подкачки в отдельном разделе, что позволяет увеличить скорость доступа к данным и уменьшить риск повреждения данных на основных разделах.

Для обеспечения максимальной скорости доступа к данным области подкачки ее раздел рекомендуется размещать в начале либо в середине диска. Данные, находящиеся в **swap-разделе**, являются временными и не представляют ценности после перезагрузки компьютера. Поэтому размещение swap-раздела в начале диска предпочтительнее: снижается риск потери важных данных.

Корневая файловая система является основой всего дерева каталогов и файлов Linux.

Корневая файловая система создаётся при установке ОС **Alt Linux** и расположена в разделах системного диска операционной системы

Точкой монтирования для корневой файловой системы служит “/” - корневой каталог.

Все данные ОС (включая пользовательские файлы) могут быть помещены в один раздел- в этом случае для ALT Linux потребуется всего два раздела.

Для повышения эффективности и надёжности некоторые каталоги операционной системы могут быть вынесены в дополнительные разделы (если это позволяет процедура установки операционной системы).

Пользовательские каталоги и файлы могут быть размещены в разделах дополнительно установленных жёстких дисках.

2.3.2 Дополнительные разделы для каталогов ОС

/home

Домашние каталоги пользователей. Здесь хранятся персональные каталоги всех пользователей машины. Размер каталога зависит от количества работающих пользователей и от их потребностей. Рекомендуемые файловые системы - Ext3

или XFS. Параметры монтирования - **noexec** (в случае невозможности применения - nosuid).

/usr

Статические данные: большая часть пакетов устанавливает свои исполняемые файлы и данные в каталог **/usr**. Преимуществом размещения этого каталога в отдельном разделе является то, что при нормальной работе (кроме установки/удаления пакетов) не требуется в него записывать никаких данных, поэтому этот раздел можно монтировать в режиме «**только чтение**», в том числе по локальной сети. В этом случае несколько машин могут пользоваться одним сетевым разделом /usr. Размер этого раздела зависит от количества пакетов, которые будут установлены, он колеблется в пределах от 500 Мб для маленькой установки до нескольких гигабайт для полной установки. Вариант на **2–3 Гб** (в зависимости от размера диска) скорее всего подойдёт.

/var

Переменные данные, которые создаются системой в процессе работы. Запись в этот каталог осуществляется весьма часто, а количество данных в нём имеет тенденцию расти (здесь расположены все **системные журналы**). Требования к объёму очень сильно зависят от профиля машины. На пользовательских домашних станциях может быть достаточно и нескольких сотен мегабайт, однако лучше выделять не меньше свободного места, чем для раздел **/usr**. На серверах объем раздела с переменными данными, как правило, больше. К тому же, для повышения производительности и надежности хранения информации переменные данные разных типов рекомендуется располагать в разных разделах. Файловая система этого раздела должна поддерживать журналирование (**Ext3**). При монтировании желательно задать параметры noexec и nosuid.

/var/log

При установке как на сервер, так и на рабочие станции, лучшим решением будет вынести системные журналы в отдельный раздел. При сбоях или внешних атаках размер журналов может резко увеличиваться, заполняя все доступное на разделе пространство, что, в случае их хранения вместе с другими переменными

системными данными, чревато сбоем в работе системы. Также, если сервер используется для выполнения узкого круга задач (например, как web-сервер), рекомендуется выносить в отдельный раздел журналы основной системной службы (например, /var/log/apache). Оптимальная файловая система.- Ext3, параметры монтирования- **noatime, noexec, nosuid**.

/var/tmp

Может быть полезным создать отдельную файловую систему для временных данных, которые нежелательно потерять в случае программного или аппаратного сбоя. Этот раздел должен обеспечивать высокую надежность хранения данных, поэтому оптимально создать в нем файловую систему с поддержкой журналирования (Ext3), указав параметры монтирования noexec и nodev.

/var/spool/mail

Если на сервере хранится почта пользователей, каталог с ней необходимо выделить в отдельный раздел. Также обязательно устанавливать ограничения на использование дискового пространства для отдельных пользователей, чтобы избежать неожиданного переполнения раздела и проблем с работоспособностью сервера.

/var/www

Раздел для сайтов пользователей.

/tmp

Этот каталог предназначен для **временных файлов**: в таких файлах программы хранят промежуточные данные, необходимые для работы. После завершения работы программы временные файлы теряют смысл и должны быть удалены. Обычно каталог /tmp очищается при каждой загрузке системы. Поскольку запись в этот каталог осуществляется очень часто, а требования к надёжности очень низкие, то есть большой смысл выделить /tmp в отдельный раздел. В противном случае он окажется частью раздела “/”, требования к которому по записи и надёжности прямо противоположные. Размер раздела /tmp в обычном случае

должен быть примерно равен размеру swap. В последнее время раздел /tmp зачастую размещают в виртуальной файловой системе tmpfs непосредственно в оперативной памяти.

/

Корневой раздел - это самый важный раздел. Он содержит наиболее важные данные и программы системы и служит точкой монтирования для других разделов. Если **/usr, /var и /home** вынесены в отдельные разделы, то потребность в объёме корневого раздела небольшая, обычно достаточно 500 Мб. Корневой раздел должен быть доступен в процессе загрузки, в процессе работы доступ на запись в этот раздел требуется нечасто, но очень важна надёжность.

/boot

Небольшой раздел (5–10 Мб), на котором хранятся исполняемые и failsafe ядра и данные используемого загрузчика. Обычно располагается в самом начале жесткого диска и всегда является первичным разделом (в отличие от логических томов в случае использования LVM). Оптимальная файловая система - Ext2, поскольку запись в раздел производится редко, а его объем мал.

Выделение дополнительных разделов направлено прежде всего на повышение эффективности работы сервера. Для рабочих станций чаще всего вполне достаточно, помимо необходимых разделов, выделить всго один - для хранения пользовательских данных (/home).

Увеличения гибкости управления разделами, особенно при большом их количестве, можно добиться использованием **технологии LVM, которая позволяет создавать, удалять и изменять размер логических устройств без риска потери данных.**

2.4 Создание разделов и файловых систем на жёстких дисках

2.4.1 Создание разделов жёсткого диска

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

При подключении нового диска необходимо:

- разделить его на разделы
- отформатировать каждый раздел для создания файловой системы
- смонтировать диск для возможности записи и чтения данных
- настроить автоматическое монтирование файловых систем при загрузке операционной системы.

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.**

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.

Разделы диска создаются:

1. С использованием таблицы разделов **PT** (partition table) и MBR(Master Boot Record). Разделы: первичные (**primary**), расширенные (**extended**), логические (**logica**l). Максимальное число разделов –**четыре**. Расширенный раздел –**один**. В расширенном разделе – **логические разделы**.

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 на диске выполняется с помощью команд **fdisk, parted. cfdisk, sfdisk, gparted.**

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 **разделе диска**, 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 использовать области подкачки с высоким приоритетом, а затем те, которые имеют более низкий приоритет. Области с одинаковым приоритетом используются попеременно.

2.4.3 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**).

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

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.

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 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.

2.4.6 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.

Смонтированной файловой системе важно указать правильные **параметры монтирования**. Задавая разделам с разными типами данных походящие

параметры, можно добиться значительного повышения производительности и безопасности.

Наиболее часто используемые общие параметры монтирования:

noatime

При каждом доступе к файлу, в том числе при чтении, обновляется время последнего доступа к нему. При использовании этого параметра это обновление производиться не будет, что может быть полезно для ускорения работы (особенно актуально для серверов).

nodev

Этот параметр не позволяет создавать в файловой системе файлы-устройства. Если точно известно, что на данной файловой системе файлы-устройства не нужны, можно использовать этот параметр для повышения безопасности.

nosuid

Параметр запрещает исполнение SUID-программ (SUID-программы перекрывают нормальные права доступа и всегда выполняются с правами владельца программы.).

noexec

Запрещает запуск исполняемых программ из файлов на данной файловой системе.

ro

Обеспечивает доступ к файловой системе только для чтения.

2.4.7 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 (напри-

мер, /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 .

2.4.8 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):**

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 **umount.**

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).

Если устройство окажется занятым, то демонтирование потерпит неудачу.

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

2.4.10 Проверка файловой системы

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.

По умолчанию **badblocks** производит безопасное тестирование блоков, при котором осуществляется только чтение. Можно выполнять безопасное тестирова-

ние 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
3. Создать раздел подкачки и подключить его к операционной системе
4. Создать в разделах диска файловые системы и выполнить монтирование файловых систем.
5. Создать виртуальную файловую систему и монтировать её
6. Создать файл подкачки и подключить его к системе
7. Добавить в файл **fstab** информацию для постоянного монтирования файловых систем

3.2 Порядок выполнения работы

1. Войти в систему под учётной записью **stud_XX** (XX –индекс группы).
2. Запустить программу виртуализации **Oracle VM VirtualBox**.

3.2.1 Добавление жёсткого диска и создание разделов диска

1. Для виртуальной машины **Alt-10** к контроллеру **SATA** добавить динамический виртуальных жёстких диска (тип файла виртуализации **VDI**, динамический виртуальный жёсткий диск):

- диск с именем **my_disk1** размером **27 Gb**

2. Запустить виртуальную машину **Alt-10**

3. Войти в систему под учётной записью **root/adminroot.**

4. Получить информацию обо всех подключенных дисках и разделах жестких дисков

5. Используя утилиты командной строки, создать на новом диске разделы с использованием таблицы разделов **GPT** (GUID Partition Table):

раздел 1 размером **2 Gb**

раздел 2 размером **10 Gb**

раздел 3 размером **5 Gb**

раздел 4 размером **10 Gb**

6. Просмотреть информацию обо всех дисках и разделах жестких дисков

3.2.2 Создание раздела подкачки в разделе sdb1

1. Форматировать раздел **sdb1** 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 sdb2

1. Создать в разделе **sdb2** файловую систему **ext4**
2. Создать точку монтирования файловой системы в домашнем каталоге пользователя **admin_kaf.** Имя точки монтирования **КАФЕДРА**
3. Смонтировать файловую систему раздела **sdb2** в точку монтирования **КАФЕДРА**

3.2.4 Создание и монтирование файловой системы в разделе sdb3

1. Создать в разделе **sdb3** файловую систему **ext4**
2. Создать точку монтирования файловой системы в домашнем каталоге пользователя **admin_kaf.** Имя точки монтирования **ОБЩИЕ_ДОКУМЕНТЫ**
3. Смонтировать файловую систему раздела **sdb3** в точку монтирования **ОБЩИЕ_ДОКУМЕНТЫ.**

3.2.5 Создание и монтирование файловой системы в разделе sdb4

1. Создать в разделе **sdb4** файловую систему **ext4**
2. Создать точку монтирования файловой системы в домашнем каталоге пользователя **admin_stud.** Имя точки монтирования - **СТУДЕНТЫ**
3. Смонтировать файловую систему раздела **sdb2** в точку монтирования **СТУДЕНТЫ**

3.2.6 Создание виртуальной файловой системы

1. Создать в домашней папке пользователя **admin_kaf** файл **fs_virt** размером **1 Gb**
2. Создать в файле **fs_virt** виртуальную файловую систему размером **1 Gb**
3. Создать в домашнем каталоге пользователя **admin_kaf** точку монтирования виртуальной файловой системы
4. Монтировать виртуальную файловую систему в точку монтирования

3.2.7 Создание файла подкачки

1. Создать в каталоге **/tmp** файл подкачки **my_swap** размером 1 Гбайт
2. Форматировать файл **my_swap** как файла подкачки.
3. Подключить файл **my_swap** как файл подкачки
4. Присвоить разделу подкачки **sdb1** высший приоритет
5. Просмотреть информацию обо всех используемых разделах и файлах подкачки

3.2.8 Редактирование файле /etc/fstab.

1. С помощью текстового редактора отредактировать файл **/etc/fstab** для постоянного монтирования файловых систем

- КАФЕДРА

- ОБЩИЕ_ДОКУМЕНТЫ

- СТУДЕНТЫ

2. Перезагрузить систему и просмотреть информацию о смонтированных файловых системах

4 Контрольные вопросы

1. С какой целью создаются разделы жёсткого диска?
2. Чем отличаются таблицы разделов PT и GPT?
3. Что такое разделы и файлы подкачки?
4. Для чего предназначено форматирование файловой системы?
5. Каково назначение операции монтирования 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. Семкин П.С.. Семкин А.П., Горячкин Б.С. Лабораторный практикум по дисциплине «Операционные системы». Часть 1. ОС Alt Linux. Управление пользователями. Администрирование дисковой подсистемы: Учебно-методическое пособие. –М.: Издательство «Спутник+», 2023. -78 с.
4. Документация для ОС «Альт Рабочая станция». Режим доступа: https://www.basealt.ru/alt-workstation/docs

6 Приложение

6.1 Утилиты для работы с дисками и разделами

6.1.1 Утилита fdisk

 # fdisk –l *показать сведения о разделах всех дисков*

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

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

6.1.2 Утилита parted

 # parted /dev/sdb print *– отображение информации о разделах диска*

 # parted /dev/sdc *- интерактивный сеанс*

6.1.3 Утилита cfdisk

 # cfdisk /dev/sdb *- интерактивный сеанс*

6.1.4 Метки разделов

 # e2label /dev/sdb1 *– просмотр метки раздела*

 # e2label /dev/sdb1 mypart1*– задание метки раздела*

 # findfs LABEL=mypart1 *– нахождение раздела по метке*

6.2 Утилиты для создания раздела и файла подкачки

6.2.1 Раздел подкачки

 # mkswap /dev/sdb1 форматировать **sdb1** как раздел подкачки

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

6.2.2 Файл подкачки

 # dd if=/dev/zero of=/myfs1/swapfile count=2048000 - создание файла размером 1 Гбайт

 # chmod 600 /myfs1/swapfile блокирование прав доступа к данному файлу

 # mkswap /myfs1/swapfile - форматирование файла **/mnt/swapfiIe** как файла подкачки.

6.2.3 Использование разделов и файлов подкачки

 # swapon –v /dev/sdb1 использовать **/dev/sdb1** как раздел подкачки

 # swapon –v /myfs1/swapfile использовать **/myfs1/swapfile** как файл подкачки

 # swapon -s показать все используемые файлы подкачки и разделы подкачки

 # swapoff -v /myfs1/swapfile прекратить использование области подкачки

 # swapon -v -p 1 /dev/sdb1 присвоить разделу подкачки **sdb1** высший приоритет

6.3 Утилиты администрирования файловых систем

6.3.1 Создание файловой системы в разделе диска

 # mkfs -t ext4 /dev/sdb1 **с**оздать файловую систему **ext4** в разделе **sdb1**

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

 # 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:

 # 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

 # mkdir mnt/myvfs1 создать точку монтирования

 # mount mnt/myfs1/vfile1 /mnt/myvfs1 монтировать виртуальную файловую систему

6.3.3 Просмотр и изменение атрибутов файловой системы

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

 # dumpe2fs -h /dev/sda1 генерирует тот же вывод, что и tune2fs

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

 # tune2fs -c -1 /dev/sda1 деактивизировать проверку

 # tune2fs -1 10 /dev/sda1 запустить проверку, зависящую от времени (через 10 дней)

 # tune2fs -j /dev/sda1 обеспечить журналирование преобразования **ext2** в **ext3**

6.3.4 Монтирование файловой системы

 # mkdir /mnt/myfs1 создание точки монтирования

 # mount показать смонтированные фс

 # mount –t ext4 показать смонтированные фс определённого типа

 # mount –t ext4 –l показать метки

 # mount /dev/sdb2 /mnt/myfs1 монтировать фс

 # mount –v /dev/sdb2 /mnt/myfs1 показывать подробную информацию

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

 # mount –vl -t dev/sdb1 /mnt/myfs1 – монтировать и показать метку

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

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

 # mount –vl -t ext4 dev/sdb1 /mnt/myfs1 – монтировать и показать метку

 # mount mnt/myfs1/vfile1 /mnt/myvfs1 монтировать виртуальную файловую систему

6.3.5 Демонтирование файловой системы

 # umount -v /dev/sdbl демонтировать с использованием имени устройства

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

 # umount -vi /mnt/mymount/ выполнить отложенное демонтирование

 # eject /dev/cdrom демонтировать и извлечь CD

 # umount /mnt/myvfs1 демонтировать виртуальную файловую систему

6.3.6 Проверка файловой системы

 # badblocks /dev/sdbl физически проверить диск на предмет поврежденных блоков.

 # badblocks -vsn /dev/sdbl выполнить безопасную проверку на предмет поврежденных блоков

 # badblocks -vsw /dev/sdal выполнить проверку на предмет поврежденных блоков с уничтожением данных

 # fsck /dev/sdbl проверка файловой системы

6.4 Структура корневой файловой системы ОС ALT Linux

/ - корневой каталог

/bin - командные оболочки (shell), основные утилиты;

/boot - содержит ядро системы;

/dev - псевдофайлы устройств, позволяющие работать с устройствами напрямую. Файлы в **/dev** создаются сервисом **udev**

/etc - общесистемные конфигурационные файлы для большинства программ в системе;

/etc/rc?.d, /etc/init.d, /etc/rc.boot, /etc/rc.d - каталоги, где расположены командные файлы, выполняемые при запуске системы или при смене её режима работы;

/etc/passwd - база данных пользователей, в которой содержится информация об имени пользователя, его настоящем имени, личном каталоге, его зашифрованный пароль и другие данные;

/etc/shadow - теневая база данных пользователей. При этом информация из файла **/etc/passwd** перемещается в **/etc/shadow**, который недоступен для чтения всем, кроме пользователя **root**. В случае использования альтернативной схемы управления теневыми паролями (TCB), все теневые пароли для каждого пользователя располагаются в каталоге **/etc/tcb/*имя пользователя*/shadow**;

/home - домашние каталоги пользователей;

/lib - содержит файлы динамических библиотек, необходимых для работы большей части приложений, и подгружаемые модули ядра;

/lost+found - восстановленные файлы;

/media - подключаемые носители (каталоги для монтирования файловых систем сменных устройств);

/mnt - точки временного монтирования;

/opt - вспомогательные пакеты;

/proc - виртуальная файловая система, хранящаяся в памяти компьютера при загруженной ОС. В данном каталоге расположены самые свежие сведения обо всех процессах, запущенных на компьютере.

/root - домашний каталог администратора системы;

/run - файлы состояния приложений;

/sbin - набор программ для административной работы с системой (системные утилиты);

/selinux - виртуальная файловая система SELinux;

/srv - виртуальные данные сервисных служб;

/sys - файловая система, содержащая информацию о текущем состоянии системы;

/tmp - временные файлы.

/usr - пользовательские двоичные файлы и данные, используемые только для чтения (программы и библиотеки);

/var - файлы для хранения изменяющихся данных (рабочие файлы программ, очереди, журналы).

Каталог **/usr:**

/usr/bin - дополнительные программы для всех учетных записей;

/usr/sbin - команды, используемые при администрировании системы и не предназначенные для размещения в файловой системе root;

/usr/local - место, где рекомендуется размещать файлы, установленные без использования пакетных менеджеров, внутренняя организация каталогов практически такая же, как и корневого каталога;

/usr/man - каталог, где хранятся файлы справочного руководства **man**;

/usr/share - каталог для размещения общедоступных файлов большей части приложений.

Каталог **/var:**

/var/log - место, где хранятся файлы аудита работы системы и приложений;

/var/spool - каталог для хранения файлов, находящихся в очереди на обработку для того или иного процесса (очереди печати, непрочитанные или не отправленные письма).