Семинар #6: Диски и файловые системы

Единицы измерения информации

единица	единица (англ.)	сокращение	сокращение (англ.)	величина
байт	byte	Б	B	1 байт
килобайт	kilobyte	КБ	KB	$1000 = 10^3$ байт
мегабайт	megabyte	МБ	MB	$1000^2 = 10^6$ байт
гигабайт	gigabyte	ГБ	GB	$1000^3 = 10^9$ байт
терабайт	terabyte	ТБ	TB	$1000^4 = 10^{12}$ байт
кибибайт	kibibyte	КиБ	KiB	$1024 = 2^{10}$ байт
мебибайт	mebibyte	МиБ	MiB	$1024^2 = 2^{20} = 1,048,576$ байт
гибибайт	gibibyte	ГиБ	GiB	$1024^3 = 2^{30} = 1,073,741,824$ байт
тебибайт	tebibyte	ТиБ	TiB	$1024^4 = 2^{40} = 1,099,511,627,776$ байт

Файлы устройств

- **Директория `/dev`**
  Директория `/dev` содержит специальные файлы, представляющие все физические и виртуальные устройства, подключенные к системе. Каждый файл в `/dev` — это не обычный файл с данными, а способ общения с определенным устройством (диском, клавиатурой, принтером и т. д.) через его драйвер в ядре.

- **Файлы блочных устройств в `/dev`**
  Блочные устройства – это устройства которые передают данные фиксированными блоками (например, 512 байт или 4 КиБ) и поддерживают произвольный доступ (то есть можно читать начиная с произвольного места в устройстве). К блочным устройствам относятся жёсткие диски (HDD), твердотельные накопители (SSD), CD/DVD (только для чтения), USB-флешки, а также разделы дисков и тома LVM.

  Если просмотреть директорию `/dev` с помощью `ls -l`, то можно увидеть файлы типа `b`:

  ```
  brw-rw----  1 root    disk      8,   0 Oct 23 13:10 sda
  brw-rw----  1 root    disk      8,   1 Oct 23 13:10 sda1
  brw-rw----  1 root    disk      8,   2 Oct 23 13:10 sda2
  brw-rw----  1 root    disk      8,  16 Oct 23 13:10 sdb
  brw-rw----  1 root    disk      8,  17 Oct 23 13:10 sdb1
  ```

  `sda`, `sdb` – это файлы дисков, а `sda1`, `sda2`, `sdb1` - файлы разделов. Эти файлы не предназначены для хранения информации, а служат интерфейсом для доступа и управления устройствами. Размер этих файлов равен нулю, вместо размера для таких файлов показывается два числа:

  - Major Number – идентифицирует класс устройств и связанный с этим классом драйвер. Старший номер 8 всегда обозначает драйвер для устройств SCSI/SATA/USB дисков.
  - Minor Number – идентифицирует конкретный экземпляр устройства или его раздел.

- **Файлы символьных устройств в `/dev`**
  Символьные устройства – это устройства, которые передают и принимают данные единым потоком байтов. Данные передаются и принимаются последовательно, по одному байту, без произвольного доступа. К символьным устройствам относятся клавиатуры, мыши, звуковые карты. Помимо физических символьных устройств, часто используются виртуальные символьные устройства, которые не привязаны к физическому оборудованию. Например, файл `/dev/tty` ссылается на терминал текущего процесса.

  ```
  crw-rw-rw-  1 root    tty       5,   0 Oct 23 14:00 tty
  ```

- **Специальные файлы виртуальных символьных устройств**
  Эти файлы не привязаны к физическому оборудованию, а реализованы как специальные интерфейсы для выполнения типовых задач ввода/вывода.
  - Файл `/dev/null` – при записи в этот файл данные игнорируются. Операция всегда проходит успешно, как будто данные были сохранены. При попытке чтения возвращается конец файла.
  - Файл `/dev/zero` – при чтении получаем неограниченную последовательность нулевых байт. При попытке записи – данные игнорируются (как и в `/dev/null`).
  - Файл `/dev/random` – неограниченная последовательность случайных байт.

Диски и разделы

Просмотр информации о дисках и разделах

- `lsblk` (*list block devices*)
  Показывает список всех блочных устройств, доступных системе, и их структуру в древовидном формате.

NAME	Системное имя устройства.
MAJ:MIN	Идентификатор, который Linux использует для управления устройством.
RM	Removable. 1, если это съемный носитель (например, флешка); 0 иначе.
SIZE	Размер.
RO	Read-Only. 1, если устройство смонтировано только для чтения; 0 иначе.
TYPE	Тип устройства: disk (весь диск), part (раздел), lvm (логический том LVM).
MOUNTPOINT	Директория, в которую смонтирована файловая система.

- `blkid` (*block device identifier*)
  Требует `sudo`. Выводит информацию о блочных устройствах, включая их статические идентификаторы, такие как UUID. Имена устройств вида `sda`, `sdb` непостоянны и могут меняться при перезагрузках системы. Например, диск `sdb` может поменять название на `sda`, а `sda` на `sdb`. Чтобы точно идентифицировать файловую систему на диске, используется UUID.

UUID	Уникальный 128-битный идентификатор, который генерируется при создании файловой системы на разделе. UUID не меняется при перезагрузках.
TYPE	Тип файловой системы.
PARTUUID	Уникальный идентификатор самого раздела.
PTUUID	Уникальный идентификатор таблицы разделов.

Таблицы разделов

Таблица разделов – это структура данных, обычно расположенная в начале диска, которая содержат информацию об разбитии этого диска на разделы. Для каждого раздела таблица хранит начальную и конечную позицию раздела, размер, тип и загрузочный флаг. Существуют два основных формата таблиц разделов:

1. **MBR (Master Boot Record)**
   - Старый, но до сих пор используемый формат. Часто называется как `msdos` или `dos` по имени операционной системы, с которой этот формат исторически начал применяться.
   - Таблица расположена в первых 512 байтах диска.
   - Максимальный размер диска – 2 ТиБ. Максимальное количество основных разделов – 4.
   - Для того, чтобы обойти ограничение на количество разделов вводят расширенный раздел. Один из четырех основных (primary) разделов становится расширенным (extended). Этот раздел сам по себе не содержит файловой системы или данных, но он служит контейнером для других разделов, называемых логическими (logical). На одном диске можно создать только один расширенный раздел, но внутри него может содержаться неограниченное количество логических разделов.
   - Перед каждым логическим разделом хранится специальная таблица EBR (Extended Boot Record), описывающая соответствующий логический раздел, размером 512 байт.

2. **GPT (GUID Partition Table)**
    - Современный стандарт.
    - Поддерживает диски размером до 9.4 миллиардов терабайт и до 128 разделов.
    - Таблица хранится в двух местах: в начале диска и в резервной копии в конце диска.
    - Использует 17408 байт в начале диска и 16896 байт в конце диска.

Выравнивание начала и конца разделов

Для более эффективной работы информация записывается на HDD/SSD-носители не по одному байту, а кусками некоторого фиксированного размера. Старые HDD традиционно использовали куски, называемые секторами, размером 512 байт. То есть при записи на диск одного байта сначала считываются 512 байт, изменяется 1 байт в этом куске, а затем записываются эти 512 байт. Современные HDD используют секторы большего размера — например, 4 КиБ, — но эмулируют работу с 512-байтными секторами для совместимости со старыми программами. Современные SSD могут читать и записывать куски, называемые страницами, размером от 4 КиБ до 16 КиБ, но стирать данные они могут только кусками размером от 512 КиБ до 1 МиБ, называемыми блоками.

Файловые системы тоже делят всю память на куски, также называемые блоками. Это делается для удобства организации файловой системы. Типичный размер блока у современных файловых систем равен 4 КиБ, но может достигать размера 1 МиБ для некоторых систем.

Крайне важно, чтобы разделы на диске были выровнены относительно значений, с которыми оперируют физические носители. Другими словами, необходимо, чтобы значения начала и конца разделов были кратны значениям, с которыми работают носители. Если это не так, то даже для записи небольшого объёма данных, который умещается в один блок файловой системы, потребуется дважды обращаться к физическому носителю, что снижает производительность и увеличивает износ. Таким образом, можно сформулировать следующее эмпирическое правило:

Начала и концы разделов на диске должны быть кратны 1 МиБ.

Если придерживаться этого правила, то никаких проблем с выравниванием на современных дисках и файловых системах не возникнет.

Рассмотрим выравнивание в контексте таблиц разделов. Хотя таблицы разделов занимают немного места, обязателен большой отступ перед началом первого раздела, чтобы он начинался со смещения в 1 МиБ. Кроме того, в формате MBR перед каждым логическим разделом также необходимо вводить отступ в 1 МиБ для размещения таблиц EBR.

Разметка диска с помощью `parted`

Программа `parted` используется для создания разделов на диске. Для запуска программы нужно ей передать файл блочного устройства. **Будьте осторожны!** Выбирайте диск, на котором нет важной информации и на котором не установлена операционная система.

```
$ sudo parted /dev/sdb
```

После выполнения команды, программа перейдёт в интерактивный режим, в котором можно выполнять следующие команды:

`print`	выводит информацию о разделах
`print free`	выводит информацию о разделах и свободном месте
`unit MiB`	устанавливает мебибайт как единицу измерения
`mklabel msdos`	создает новую таблицу разделов типа MBR на диске
`mklabel gpt`	создает новую таблицу разделов типа GPT на диске
`mkpart тип_раздела начало конец`	создает новый раздел типы разделов: primary, extended, logical
`rm номер_раздела`	удаляет раздел по номеру
`resizepart номер конец`	изменяет размер раздела (но не размер файловой системы)
`quit`	выход из программы

Замечания по работе с программой `parted`:

- В начале работы установите единицы измерения как MiB для удобства.

  ```
  (parted) unit MiB
  ```

  в ином случае программа будет печатать значения в мегабайтах, а не мебибайтах.

- Значения начала и конца разделов лучше писать с использованием единиц измерения, например так:

  ```
  (parted) mkpart primary 1MiB 500MiB
  ```

- Желательно, чтобы значения начала и конца разделов были кратны 1 МиБ. Если это будет не так, то программа автоматически округлит начало и конец разделов до оптимального значения.

- В качестве размеров диска можно использовать и проценты. В этом случае значения границ разделов будут округлены до ближайшего целого значения в мебибайтах.

  ```
  (parted) mkpart primary 500MiB 100%
  ```

- Программа `parted` только размечает диск, но не создаёт файловые системы. Для создания файловых систем используется другая команда – `mkfs`.

- При создании разделов в `parted` можно указать тип файловой системы:

  ```
  (parted) mkpart primary ext4 1MiB 500MiB
  ```

  Но эта операция не создаёт файловую систему на данном разделе, а просто делает небольшие пометки в таблице разделов. Впоследствии можно использовать этот раздел для создании файловой системы любого типа.

- Программу `parted` можно использовать и не в интерактивном режиме следующим образом:

  ```
  $ sudo parted /dev/sdb mkpart primary 1MiB 500MiB
  ```

- Программу `parted` можно использовать и в скриптах. Для этого есть специальная опция `-s`. Например:

  ```
  $ sudo parted /dev/sdb -s mkpart primary 1MiB 500MiB
  ```

  Опция `-s` подавляет вывод от большинства команд и не останавливает, если вы делает что-то не то.

- В отличие от некоторых других программ разметки, `parted` сразу применяет изменения на диск после каждой команды. Будьте осторожны, выполнение некоторых команд в `parted` может привести к необратимым последствиям.

Просмотр информации о таблице разделов

Просмотреть информацию о таблице разделов и о самих разделах можно с помощью команды `parted -l`:

```
$ sudo parted -l
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sda: 21.5GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system     Flags
 1      1049kB  20.3GB  20.3GB  primary   ext4            boot
 2      20.3GB  21.5GB  1154MB  extended                  lba
 5      20.3GB  21.5GB  1154MB  logical   linux-swap(v1)  swap
```

Запись `Partition Table: msdos` означает, что используется таблица разделов MBR. Также видно, что раздел 1 помечен как `boot` – этот раздел содержит загрузочные файлы (например, GRUB) и является разделом, с которого система должна загружаться. Раздел 2 – расширенный, который содержит внутри себя раздел 5. Раздел 5 помечен как `swap`. Это специальный тип раздела, который Linux использует как область подкачки. Это виртуальная память, которую система использует, когда оперативной памяти недостаточно.

Файловые системы

- **ext**
  Первая расширенная файловая система для Linux, разработанная для преодоления ограничений Minix FS. Устарела и на практике больше не используется из-за отсутствия журналирования и других ограничений.

- **ext2**
  Вторая расширенная файловая система Linux. Не использует журналирование, что делает её быстрой, но уязвимой для потери данных при внезапном отключении питания. До сих пор используется для некоторых съемных носителей или в загрузочных разделах.

- **ext3**
  Третья расширенная файловая система Linux, по сути, ext2 с добавлением журналирования. Журналирование повышает надежность, сокращая время восстановления после сбоя, поскольку системе не нужно проверять весь диск.

- **ext4**
  Четвертая расширенная файловая система Linux, текущий стандарт де-факто для большинства дистрибутивов Linux. Включает экстенты (улучшенное управление пространством), отложенное выделение, большее ограничение на размер файла/тома и более высокую производительность по сравнению с ext3.

- **XFS**
  Высокопроизводительная журналируемая файловая система. Хорошо подходит для больших файлов, больших томов и высокой пропускной способности, часто используется на серверах.

- **Btrfs**
  Файловая система с поддержкой "copy-on-write"(CoW) для Linux, разработанная с упором на отказоустойчивость, самовосстановление, снимки (снапшоты) и встроенное управление томами. Считается потенциальной заменой `ext4` в будущем.

- **APFS**
  Apple File System, стандартная файловая система для современных операционных систем Apple. Оптимизирована для флэш-памяти/SSD, поддерживает клонирование файлов, снимки, шифрование и более эффективное управление пространством.

- **FAT32**
  Старая файловая система, широко совместимая практически со всеми операционными системами и устройствами. Ее главный недостаток — ограничение на размер файла в 4 ГБ и небольшой максимальный размер тома, что ограничивает ее использование.

- **exFAT**
  Extended File Allocation Table, разработанная Microsoft для замены FAT32. Преодолевает ограничение FAT32 в 4 ГБ, обеспечивая широкую совместимость, что делает ее идеальной для флэш-накопителей большой емкости и внешних дисков.

- **NTFS**
  New Technology File System, стандартная файловая система для ОС Windows. Поддерживает журналирование, списки контроля доступа, шифрование, сжатие и большие размеры файлов/томов.

Создание файловых систем

Создать файловую систему на разделе можно с помощью команды `mkfs`:

```
$ sudo mkfs.ext4 /dev/sdb1
$ sudo mkfs.xfs /dev/sdb1
$ sudo mkfs.vfat -F 32 /dev/sdb1
$ sudo mkfs.ntfs /dev/sdb1
```

Монтирование файловой системы

Для монтирования/размонтирования файловой системы используются команды `mount` и `umount`. Можно монтировать в любую директорию, но обычно используются поддиректории директории `/mnt`.

```
$ mkdir /mnt/new
$ mount /dev/sdb1 /mnt/new
```

После этого можно содержимое раздела `/dev/sdb1` появится в `/mnt/new`.
Для размонтирования: `umount /dev/sdb1` или `umount /mnt/new`.

Опции монтирования

`defaults`	Использует набор стандартных опций: `rw`, `suid`, `dev`, `exec`, `auto`, `nouser`, `async` `relatime`
`rw`	монтировать файловую систему для чтения и записи
`ro`	монтировать файловую систему только для чтения
`nosuid`	игнорировать биты SUID и SGID (безопасность)
`suid`	не игнорировать биты SUID и SGID
`dev`	файлы блочных и символьных устройств будут интерпретироваться как реальные устройства
`nodev`	файлы блочных и символьных устройств не будут интерпретироваться как реальные устройства
`exec`	разрешает выполнение бинарных файлов с этой ФС
`noexec`	запрещает выполнение бинарных файлов с этой ФС
`auto`	будет монтироваться автоматически во время загрузки системы
`user`	разрешает обычному пользователю (не `root`) монтировать и размонтировать файловую систему
`users`	похожа на `user`, но любой пользователь может монтировать и размонтировать устройство
`owner`	разрешает размонтировать файловую систему только владельцу устройства
`nouser`	только `root` может монтировать
`sync`	все операции ввода/вывода на файловой системе будут выполняться синхронно
	снижает производительность, но повышает надежность данных
`async`	разрешает асинхронные операции записи
`atime`	обновляет время последнего доступа к файлу каждый раз, когда к нему обращаются
`noatime`	не обновляет время последнего доступа к файлу. Значительно повышает производительность
`relatime`	обновляет atime только в том случае, если старое значение atime старше времени изменения файла

Указать опции можно так `mount -o ro,noexec ...` Если не указать опции, то выберется `defaults`.

Файл `/etc/fstab`

Команды для просмотра информации о смонтированных файловых систем и занятом на них месте

- `findmnt`
  Отображает список смонтированных (подключённых) файловых систем.

TARGET	Точка монтирования – каталог, куда подключено устройство.
SOURCE	Устройство, содержащее файловую систему.
FSTYPE	Тип файловой системы.
OPTIONS	Параметры монтирования.

  Помимо обычных файловых систем, привязанных к дискам, команда `findmnt` также отображает виртуальные файловые системы. Они используются для взаимодействия с различными ресурсами системы.

  - `/sys` – содержит `sysfs`, виртуальную ФС, которая предоставляет иерархическое представление конфигурации системы и подключенного оборудования.
  - `/proc` – содержит `procfs`, виртуальную ФС, которая является интерфейсом к информации о запущенных процессах и состоянию ядра.
  - `/dev` – содержит `devtmpfs`, виртуальную ФС, которая динамически создает файлы устройств.

  Эти виртуальные файловые системы хранят данные не на физическом диске, а в оперативной памяти. Содержимое этих директорий полностью пересоздается после каждой перезагрузки системы.

- `df` (*disk free*)
  Используется для отображения информации об использовании дискового пространства смонтированными файловыми системами.

`df`	размеры занятых данных будет показываться в кибибайтах
`df -h`	размеры занятых данных будет показываться human-readable формате
`df -hT`	дополнительно покажет тип файловой системы
`df -i`	показывает количество занятых inode

- `du` (*disk usage*)
  Используется для вывода информации об использовании дискового пространства файлами.

`du файл`	покажет размер файла, который он занимает на диске в кибибайтах. учтите, что все файлы занимают кратное число блоков (размер блока обычно 4КиБ) даже маленький файл займёт как минимум 1 блок
`du директория`	общий размер директории и всех поддиректорий рекурсивно
`du -s директория`	общий размер только для заданной директории
`du -h`	размеры будет показываться human-readable формате
`du -sh директория`	общий размер данной директории
`du -sh директория/*`	размеры всех файлов в данной директории