Глава 6:

 Файлови системи, дялове и дискови квоти

Обща информация – дискове, дялове, файлови системи;  
• Управление на дялове с fdisk и parted;  
• Видове файлови системи;  
• Създаване на файлови системи (mkfs);  
• Настройки на файловите системи (tune2fs, dumpe2fs, fsck);  
• Монтиране и демонтиране на файлови системи (mount, umount);  
• NFS и SMB;  
• Настройка на /etc/fstab;  
• Проверка на използвано дисково пространство (du и df);  
• Конфигуриране на дискови квоти.

# Видове файлови системи

Файловете на медиите (твърди дискове,флаш устройства, оптични носители и др.) се организират, съхраняват и назовават по определен ред, който се нарича файлова система. Различните носители са различни **типове файлови системи**, Кои от тях може да срещне обикновеният потребител?

**Файловата система организира файлове, така че операционната система да може по-лесно да работи с тях**: драйверите на файловата система предават информацията за операционната система заимената на файловете, техния размер, атрибути, местоположения. Файловата система определя максималната възможна дължина на името на файла, неговия максимален размер и други параметри.

За различните носители съществуват различни видове файлови системи. Между другото, средата не трябва да бъде физическа: има например виртуални и мрежови файлови системи. **Какви са видовете файлови системи в зависимост от целта им, т.е. средата?**

На първо място, потребителят е изправен пред **файлови системи, създадени за носители с произволен достъп**, Такива носители включват например твърди дискове. Ако използвате операционната система Windows, най-вероятно имате работа с файлова система **NTFS**, По-старите версии на операционната система използват файлова система **FAT32**, който все още се използва на флаш дискове.

При много дистрибуции на операционни системи, базирани на ядрото на Linux, файловата система по подразбиране обикновено се използва **вътр** (Разширена файлова система - разширена файлова система). Има няколко версии на тази файлова система - **ext2, ext3, ext4**, В последните версии на дистрибуциите, базирани на ядрото на Linux (включително Google Android), файловата система е ext4.

Файловите системи са **оптични носители** - CD и DVD дискове. Стандартът се счита за стандартен **ISO 9660**, такива дискове се четат от компютри с всяка операционна система - Windows, Mac OS X, Unix. Съществува и формат на файловата система **СДС**, което е по-подходящо за големи дискове (DVD, Blu-ray). Има и други файлови системи за оптични дискове, по-рядко срещани.

При твърди дискове, флаш памети и оптични дискове се срещаме по-често, отколкото при други оператори, така че техните файлови системи ни интересуват най-много.**Но все пак си струва да знаете какви други видове файлови системи съществуват**:

* виртуални файлови системи;
* мрежови файлови системи;
* файлови системи за последователни медии за достъп (те включват, например, магнитни ленти);
* файлови системи за флаш памет;
* специализирани файлови системи.

Нека поговорим повече за видовете файлови системи за медиите с произволен достъп, например, твърди дискове и флаш памети. **Типът на конкретна файлова система засяга настройките на файла, например размера на името на файла**, В системата FAT32 максималната дължина на името на файла- 255 знака. В NTFS, според спецификацията, има 32 768 знака, но някои операционни системи налагат ограничения, така че в действителност максималната дължина ще бъде една и съща 255 символа Unicode. В ext2 / ext3, дължината на името е ограничена до 255 байта.

също **от файловата система зависи от възможните атрибути на файла**, Например, FAT32 и NTFS системи ви позволяват да присвоитефайловете атрибутите "само за четене", "система", "скрит", "архив". Системата ext2 предлага атрибути като "задаване на потребителски идентификатор", "задаване на идентификатор на група" и така наречения "лепкав бит".

**Съществуват и разлики между файловите системи FAT32 и NTFS**, И двете файлови системи се използват от операционната системаWindows, системата NTFS замени FAT32 и се използва в най-новите версии на операционната система. В системата FAT32 размерът на диска е ограничен до около 8 терабайта, в NTFS може да бъде 264 байта. Максималният размер на файла в FAT32 е 4 GB, в NTFS е 264 байта минус 1 килобайт (теоретично), а всъщност - 244 байта минус 64 килобайта. Освен това NTFS има повече максимален брой файлове, има и други разлики.

Но в този случай **Системата FAT32 все още се използва на USB флаш устройства (флаш памети)**, защото осигурява по-висока скоростзаписване, четене и копиране на данни. Поради това повечето флаш устройства са форматирани в FAT32, а не в NTFS. Форматирането на USB флаш устройство в NTFS има смисъл само ако трябва да напишете файл с размер по-голям от 4 GB.

Сега знаете, **какви са основните типове файлови системи** и в кои случаи може да срещнете определена файлова система.

Сега да разгледаме нашата линукс файлова система

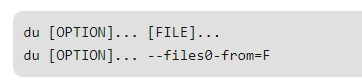
За целта ще използваме командите du и dh

# **Du**

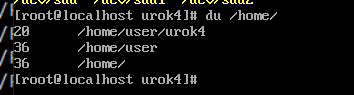
Командата du, съкратено за използването на диска, се използва за оценка на използването на файловото пространство.

Командата du може да се използва за проследяване на файлове и директории, които консумират прекомерно много място на твърдия диск.

Синтаксиса е :



Например



**Опциите са :**

**-0, –null: приключва всяка изходна линия с NULL**

**-a, –всички: брой записи за всички файлове, а не само за директории**

**-Намаляващ размер: отпечатва се видими размери, а не използване на диск.**

**-B, -block-size = SIZE: мащабирайте размерите до SIZE преди печат на конзолата**

**-c, - общо: произвежда общия сбор**

**-d, –max-depth = N: отпечатва общо за директорията само ако е N или по-малко нива под аргумента на командния ред**

**-h, - четене от човека: размери за печат в четлив за човека формат**

**-S, -разделни-директории: за директории, не включват размер на поддиректории**

**-s, –summarize: показва само общо за всяка директория**

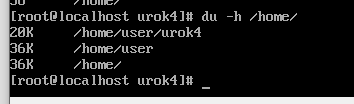
**–Time: показва времето на последната модификация на всеки файл или директория.**

**-Exclude = PATTERN: изключва файлове, които съответстват на PATTERN**

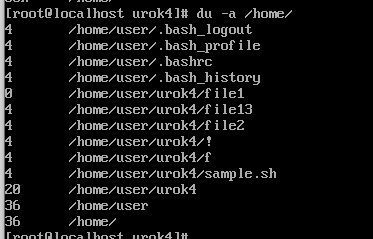
37/5000

Примери за използване на командите с опции:

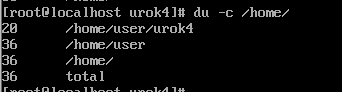
**Ако искаме да отпечатаме размери в четлив за човека формат (K, M, G), използвайте опцията –h**



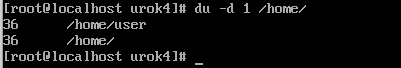
**Използвайте -a опция за отпечатване на всички файлове, включително директории.**



**Използвайте опцията -c, за да отпечатате общия размер**



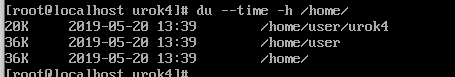
**За да отпечатате размери до определено ниво, използвайте опцията -d с ниво no.**



**Получете обобщение на опцията за използване на файлова система**



**Вземете времевия отпечатък от последната модификация, използвайки опцията –time**



**За разлика от Du df ни дава друг вид статистики**

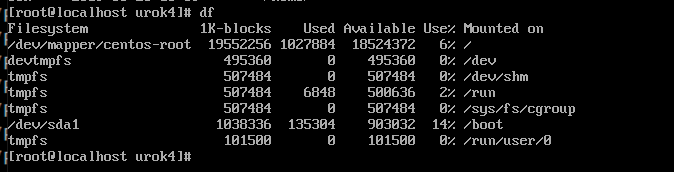
**Командата df (съкратено от диск свободно), се използва за показване на информация, свързана с файловите системи за общото пространство и наличното пространство.**

**Синтаксиса е :**



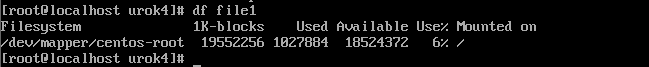
**Ако не е зададено име на файл, то показва наличното пространство за всички текущо монтирани файлови системи.**

**Например :**



**Сега, ако посочите конкретен файл, той ще покаже информация за монтирането на този конкретен файл.**

**Например:**



**Опции за командата df:**

**-a, - всички: включва псевдо, дублирани и недостъпни файлови системи.**

**-B, -block-size = SIZE: мащабира размера по SIZE преди да ги отпечата.**

**-h, - четене от човека: размери на печата в сила от 1024**

**-H, –si: размери за печат с мощност 1000**

**-i, –inodes: списък на информация за inode вместо използването на блока**

**-l, -local: ограничаване на списъка към локални файлови системи**

**-P, - преносимост: използвайте POSIX изходния формат**

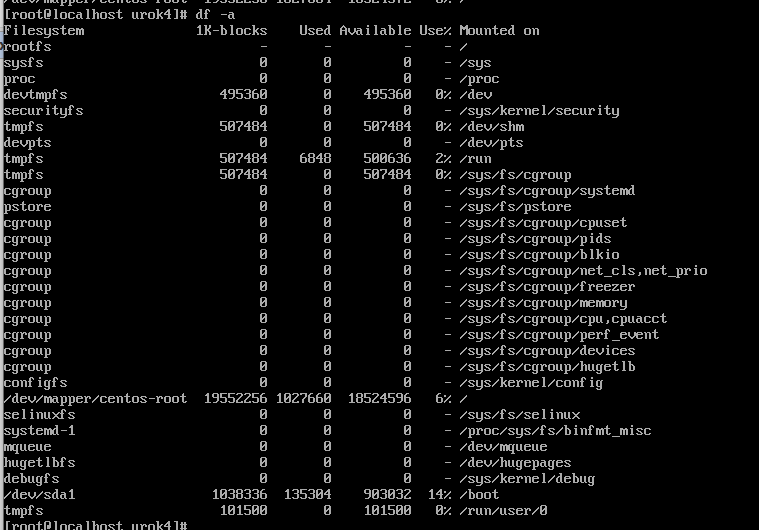
**–Sync: извиква синхронизирането, преди да получи информация за използването**

**- общо: премахнете всички записи, които са незначителни за наличното пространство, и генерирайте общ сбор**

**-t, -type = TYPE: ограничаване на списъка към файлови системи от тип TYPE**

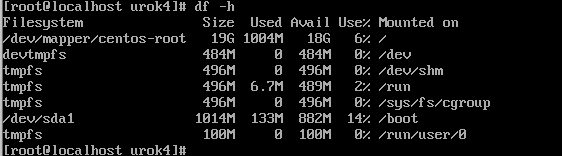
**-T, - print-type: тип на печатната файлова система**

Ако искате да покажете цялата файлова система, използвайте опцията –a



Посоченото по-горе не е пълно изход, но можете да видите, че показаната информация е разширена до информация, предоставена от командата df.

Използвайте опцията -h за показване на размера



И така да се върнеме обратно на това какво е файлова система и смисълът от нея .

**Файлови системи Видове файлови системи. Файлови операции. Каталози. Операции с указатели.**

*досие*  - Това е имена на външна памет, в която можете да пишете и от които можете да четете данни.

Основното използване на файла.

*Дългосрочно и надеждно съхранение на информация*, Дълготрайността се постига чрез използването на устройства за съхранение, които не са зависими от мощността, а високата надеждност се определя чрез защита на достъпа до файловете и общата организация на софтуерния код на операционната система, в който отказите на оборудването най-често не унищожават информацията, съхранявана във файлове.

*Споделяне на информация*, Файловете осигуряват естествен и лесен начин за споделяне на информация между приложения и потребители, като имат четливо за човека символично име и постоянство на съхранената информация и местоположение на файла. Потребителят трябва да има удобни средства за работа с файлове, включително директории-директории, комбиниране на файлове в групи, инструменти за търсене на файлове по функции, набор от команди за създаване, модифициране и изтриване на файлове. Файлът може да бъде създаден от един потребител и след това да се използва от напълно различен потребител, докато създателят на файла или администраторът може да определи правата за достъп на други потребители до него. Тези цели се изпълняват в файловата система на ОС.

*Файлова система (FS)*  - е част от операционната система, включително:

събиране на всички файлове на диска;

набори от структури от данни, използвани за управление на файлове, като например файлови директории, файлови дескриптори, таблици за разпределяне на свободно и използвано дисково пространство;

набор от системни софтуерни инструменти, които реализират различни файлови операции, като например създаване, унищожаване, четене, писане, именуване и търсене на файлове.

По този начин файловата система играе ролята на междинен слой, защитавайки всички сложности на физическата организация на дългосрочното съхранение на данни, и създавайки по-опростен логически модел за това съхранение за програми, както и предоставяйки им набор от лесни за използване команди за манипулиране на файлове.

*Следните файлови системи са широко известни:*

*файлова система операционна система MS - DOS въз основа на таблица за разпределение на файлове - FAT ( досие разпределяне маса ).*

Таблицата съдържа информация за местоположението на всички файлове (всеки файл е разделен на *клъстери*  в съответствие с наличието на свободно дисково пространство, клъстерите на един файл не са непременно разположени един до друг). Файловата система MS-DOS има значителни ограничения и недостатъци, например под *името*  файлът се разпределя на 12 байта, работата с голям твърд диск води до значителна фрагментация на файла;

*Основните функции в такава файлова система са насочени към решаване на следните задачи:*

именуване на файлове;

софтуерен интерфейс за приложения;

картографиране на логическия модел на файловата система към физическата организация на хранилището за данни;

устойчивост на файловата система към прекъсвания на захранването, хардуерни и софтуерни грешки.

*OS /2 наречен HPFS ( високо - производителност досие система    - високоскоростна файлова система).*

Осигурява възможност за име на файл до 254 знака. Файловете, записани на диск, имат минимална фрагментация. Може да работи с файлове, записани в MS DOS;

*Към горните задачи се добавя нова задача.*  споделяне на файл от множество процеси. Файлът в този случай е споделен ресурс, което означава, че файловата система трябва да реши целия комплекс от проблеми, свързани с такива ресурси. По-специално, във файловата система трябва да бъдат осигурени средствата за блокиране на файл и неговите части, предотвратяване на състезания, премахване на задънена улица, съвпадение на копия и т.н.

В многопотребителските системи се появява още едно предизвикателство: защита на файловете на един потребител от неоторизиран достъп от друг потребител.

*файловата система на операционната система Windows 95*

Той има слоеста структура, която ви позволява едновременно да поддържате множество файлови системи. Старата MS-DOS файлова система се поддържа директно, а файловите системи не са разработени от компанията. *Microsoft*подкрепени от специални *модули*, Можете да използвате дълги (до 254 знака) имена на файлове.

*файлови системи на операционната система Unix*

Те осигуряват унифициран начин за достъп до I / O файлови системи.

Правата за достъп до файловете практически определят правата за достъп до системата (собственикът на файла е потребителят, който го е създал).

**Типове файлове**

Файловите системи поддържат няколко функционално различни типа файлове, които обикновено включват обикновени файлове, файлови директории, специални файлове, имена на тръбопроводи, картографирани файлове с памет и други.

*Редовни файлове*или просто файлове, съдържащи информация от произволен характер, която потребителят въвежда в тях или който се формира в резултат на работата на системни и потребителски програми. Повечето съвременни операционни системи (например UNIX, Windows, OS / 2) не ограничават или контролират съдържанието и структурата на обикновен файл. Съдържанието на обикновен файл се определя от приложението, което работи с него. Например текстов редактор създава текстови файлове, състоящи се от низове от символи, представени в някой код. Това могат да бъдат документи, изходни кодове на програми и др. Текстовите файлове могат да се четат на екрана и да се отпечатват от принтер. Двоичните файлове не използват символни кодове, те често имат сложна вътрешна структура, като изпълним програмен код или архивен файл. Всички операционни системи трябва да могат да разпознават поне един тип файл - свой собствен изпълним файл.

*Каталози*  - това е специален тип файлове, които съдържат системна справочна информация за набор от файлове, групирани по потребители според някаква неформална характеристика (например файлове, съдържащи документи от един договор или файлове, съдържащи един софтуерен пакет, се обединяват в една група). В много операционни системи директорията може да включва файлове от всякакъв тип, включително други директории, поради което се създава дървовидна структура, която е удобна за търсене. Директориите установяват съответствие между имената на файловете и техните характеристики, използвани от файловата система за управление на файлове. Такива характеристики включват по-специално информация (или указател към друга структура, съдържаща тези данни) за типа на файла и неговото местоположение на диска, права за достъп до файла и датите на неговото създаване и модификация. Във всички други аспекти, директориите се третират като обикновени файлове от файловата система.

*Специални файлове*Това са фиктивни файлове, свързани с I / O устройства, които се използват за унифициране на механизма за достъп до файлове и външни устройства. Специалните файлове позволяват на потребителя да изпълнява I / O операции, като използва обичайните команди за записване или четене от файл. Тези команди се обработват първо от програмите на файловата система и след това на някакъв етап от изпълнението на заявката операционната система се превръща в контролни команди за съответното устройство.

Модерните файлови системи поддържат други типове файлове, като например символни връзки, имена на тръбопроводи, картографирани файлове с памет.

**Йерархична структура на файловата система**

Потребителите имат достъп до файлове със символични имена. Възможностите за човешка памет обаче ограничават броя на имената на обекти, които потребителят може да получи по име. Йерархичната организация на пространството от имена ви позволява значително да разширите тези граници. Ето защо повечето файлови системи имат йерархична структура, в която нивата се създават поради факта, че директория с по-ниско ниво може да бъде включена в директория с по-високо ниво (фиг. 7.3).

Графиката, описваща йерархията на директориите, може да бъде дърво или мрежа. Директориите образуват дърво, ако на файла е позволено да влезе само в една директория (фиг. 7.3, б), а мрежата - ако файлът може да бъде включен в няколко директории едновременно (фиг. 7.3, в). Например в MS-DOS и Windows директориите образуват дървовидна структура, а в UNIX - мрежовата структура. В дървовидна структура всеки файл е лист. Директорията от най-високо ниво се извиква *основна директория или корен ( корен ).*

При такава организация потребителят се освобождава от запаметяването на имената на всички файлове, достатъчно е да има груба представа за коя група даден файл може да бъде назначен, за да го намери чрез последователно преглеждане на директории. Йерархичната структура е удобна за многопотребителска работа: всеки потребител с неговите файлове е локализиран в собствената си директория или поддерево на директории, и в същото време всички файлове в системата са логически свързани.

Специален случай на йерархичната структура е организация на едно ниво, когато всички файлове са включени в една директория (фиг. 7.3, а).

**Имената на файловете**

Всички типове файлове имат символични имена. В йерархично организирани файлови системи обикновено се използват три типа имена на файлове: прости, сложни и относителни.

Просто или кратко символично име идентифицира файл в една директория. Простите имена се присвояват на файлове от потребители и програмисти, като в същото време те трябва да вземат под внимание ограниченията на операционната система както върху диапазона от символи, така и върху дължината на името. До относително скоро тези граници бяха много тесни. Така, в популярната файлова система FAT, дължината на имената е ограничена до 8,3 (8 символа - самото име, 3 символа - разширение на името), а в файловата система s5, поддържана от много версии на UNIX OS, просто символично име не може да съдържа повече от 14 символа. Въпреки това е много по-удобно за потребителя да работи с дълги имена, тъй като те позволяват да се дават файлове лесни за запомняне имена, които ясно показват какво се съдържа в този файл. Следователно, съвременните файлови системи, както и подобрените версии на съществуващите файлови системи, са склонни да поддържат дълги, прости, символични имена на файлове. Например във файловите системи NTFS и FAT32, включени в операционната система Windows NT, името на файла може да съдържа до 255 символа.

В йерархичните файлови системи на различни файлове е позволено да имат едни и същи символни имена, при условие че принадлежат към различни директории. Това означава, че тук работи схемата "много файлове - едно просто име". За уникална идентификация на файл в такива системи се използва т.нар. Пълно име.

Пълното име е верига от прости символни имена на всички директории, през които преминава пътя от корена до дадения файл. По този начин пълното име е композитно, в което простите имена се отделят един от друг от сепаратора, приет в операционната система. Често напред или обратна наклонена черта се използва като разделител и е обичайно да не се указва името на главната директория. На фиг. 7.3, b два файла имат просто име main.exe, но техните имена /depart/main.exe и /user/anna/main.exe са различни.

В дървовидната файлова система има едно-към-едно съответствие между файла и неговото пълно име “един файл - едно пълно име”. В файловите системи, които имат мрежова структура, файлът може да бъде включен в няколко директории, което означава, че може да има няколко пълни имена; тук кореспонденцията “един файл - много пълни имена” е справедлива. И в двата случая файлът е уникално идентифициран с пълното име.

Файлът може също да бъде идентифициран с относително име. Относителното име на файла се определя чрез понятието "текуща директория". За всеки потребител по всяко време един от директориите на файловата система е актуален и тази директория се избира от потребителя чрез командата OS. Файловата система улавя името на текущата директория, за да я използва като допълнение към относителните имена, за да формира пълното име на файла. Когато се използват относителни имена, потребителят идентифицира файла с верига от имена на директории, чрез които маршрутът преминава от текущата директория към този файл. Например, ако текущата директория е директорията / user, тогава относителното име на файла /user/anna/main.exe изглежда така: anna / main.exe.

На някои операционни системи е позволено да присвоите няколко прости имена на един и същи файл, които могат да се интерпретират като псевдоними. В този случай, както и в система с мрежова структура, кореспонденцията “един файл - много пълни имена” е установена, тъй като поне едно пълно име съответства на всяко просто име на файл.

Въпреки че пълното име еднозначно идентифицира файла, операционната система е по-лесно да работи с файла, ако между файловете и техните имена има съответствие между тях. За тази цел той присвоява уникално име на файла, така че съотношението „един файл - едно уникално име“ е валидно. Съществува уникално име заедно с едно или повече символични имена, присвоени на файл от потребители или приложения. Уникалното име е цифров идентификатор и е предназначено само за операционната система. Пример за такова уникално име на файл е номерът на inode в UNIX.

**Файлови атрибути**

Понятието „файл“ включва не само данните и името, което съхранява, но и атрибутите. *атрибути*  - Това е информация, описваща свойствата на файла. Примери за възможни атрибути на файлове:

тип на файла (обикновен файл, директория, специален файл и т.н.);

собственика на файла;

създател на файлове;

парола за достъп до файла;

информация за разрешени операции за достъп до файлове;

създаване, последно време за достъп и последна промяна;

текущ размер на файла;

максимален размер на файла;

флаг „само за четене“;

функция "скрит файл";

знакът "системен файл";

знакът "архивен файл";

знакът "двоичен / символен";

знакът "временно" (изтриване след приключване на процеса);

знак за блокиране;

дължината на записа във файла;

показалец към ключовото поле в записа;

дължина на ключа

Наборът от файлови атрибути се определя от спецификата на файловата система: в файловите системи от различен тип могат да се използват различни атрибутни набори за характеризиране на файлове. Например във файловите системи, които поддържат неструктурирани файлове, няма нужда да се използват последните три атрибута в горния списък, свързани с структурирането на файлове. В една потребителска операционна система, атрибутите, свързани с потребителите и защитата, няма да присъстват в набора атрибути, като например собственика на файла, създателя на файла, паролата за достъп до файла, информацията за разрешения достъп до файла.

Потребителят може да получи достъп до атрибутите, като използва инструментите, предоставени за тази цел от файловата система. Обикновено е позволено да се четат стойностите на всякакви атрибути и промяната - само някои. Например, потребителят може да променя разрешенията за файлове (при условие че има необходимите правомощия за това), но не му е позволено да променя датата на създаване или текущия размер на файла.

Стойностите на файловите атрибути могат да се съдържат директно в директории, както се прави в MS-DOS файловата система (фиг. 7.6, а). Фигурата показва структурата на запис в директорията, съдържаща просто символично име и файлови атрибути. Тук буквите показват знаците на файла: R е само за четене, А се архивира, Н е скрито, S е системно.



**Фиг. 7.6.**  Структура на директориите: a - структура на записите на директория на MS-DOS (32 байта), b - структура на влизане в директорията на UNIX OS

Друга възможност е атрибутите да се поставят в специални таблици, когато директориите съдържат само препратки към тези таблици. Този подход е реализиран например в файловата система ufs на UNIX OS. Структурата на директориите е много проста в тази файлова система. Записът за всеки файл съдържа кратко символично име на файл и указател към inode на файла, т.нар. Ufs таблица, в която са концентрирани стойностите на файловите атрибути (фиг. 7.6, б).

И в двата случая директориите осигуряват връзка между имената на файловете и самите файлове. Въпреки това, подходът, когато името на файла е отделено от неговите атрибути, прави системата по-гъвкава. Например, файлът може лесно да бъде включен в няколко директории едновременно. Вписванията за този файл в различни директории могат да съдържат различни прости имена, но същият номер на inode ще бъде посочен в референтното поле.

Файлови операции

Повечето съвременни операционни системи разглеждат файла като неструктурирана последователност от байтове с променлива дължина. Стандартно *POSIX*следните операции са дефинирани във файла:

*инт отворен ( овъглявам \* fname , инт знамена , вид \_ т вид )*

Тази операция `` отваря файла, установяващ връзка между програмата и файла. Програмата получава **дескриптор на файл**  е цяло число, идентифициращо това съединение. Всъщност, това е индексът в системната таблица на отворените файлове за тази задача. Всички други операции използват този индекс за справка с файла.

Параметърът char \* fname определя името на флаговете file.int е маска, която определя режима на отваряне на файла.Файлът може да бъде отворен само за четене, само за писане и четене и писане; Освен това можете да отворите съществуващ файл и да опитате да създадете нов файл с нулева дължина.По желание режимът на трети параметър се използва само при създаването на файла и задава атрибутите на този файл.

*от \_ т lseek ( инт дръжка , от \_ т офсет , инт отдето )*

Тази операция премества показалеца за четене / запис във файла, а параметърът offset определя броя на байтовете, към които трябва да се премине стрелката, и параметъра wherece откъде да се компенсира отместването. текущото местоположение на показалеца (SEEK\_CUR). Операцията връща позицията на показалеца, броена от началото на файла. По този начин, извикването на lseek (манипулатор, 0, SEEK\_CUR) ще върне текущата позиция на показалеца, без да го премества.

*int четене (int дръжка, char \* where, size\_t how\_much)*

Четене на операция от файл. Когато указателят определя буфера къде да се поставят прочетените данни; Третият параметър показва колко данни да се четат: системата чете необходимия брой байтове от файла, започвайки от показателя четене / запис в този файл, и премества показалеца до края на прочетената последователност. Ако файлът е изчерпан преди, толкова данни се четат, както е останало до края му. Операцията връща броя прочетени байтове. Ако файлът е отворен само за писане, извикването за четене ще върне грешка.

*int write (int дръжка, char \* what, size\_t how\_much)*

Запис на операция във файл. Какъв указател определя началото на буфера за данни, третият параметър определя колко данни да се запишат, системата записва необходимия брой байтове в файла, като се започне с показалеца за четене / запис в този файл, като се заменят данните, съхранявани на това място, и се премества показалецът към края на записания блок. Ако файлът изтече по-рано, неговата дължина се увеличава. Операцията връща броя на записаните байтове.

Ако файлът е отворен само за четене, извикването на запис ще върне грешка.

*int ioctl (int дръжка, int cmd, ...) ; инт fcntl ( инт дръжка , инт пМС , ...)*

Допълнителни операции с файлове. Първоначално се предполагаше, че ioctl е операция върху самия файл, а fcntl са операции на дескриптора на отворения файл, но след това историческото развитие донякъде смесва функциите на тези системни повиквания. стандарт *POSIX*  дефинира някои операции като суперскриптор, например дублиране (в резултат на тази операция получаваме два дескриптора, свързани с един и същи файл), а в самия файл, например операцията за отрязване - изрязва файла до определена дължина. *Unix*  операцията за отрязване може да се използва и за отрязване на данни от средата на даден файл. При четене на данни от такава изрязана област се четат нули и тази област не заема никакво физическо дисково пространство.

Важна операция е да заключите части от файла *POSIX*  оферти за тази библиотечна функция, но в семейни системи *Unix*  Тази функция се изпълнява чрез извикване на fcntl.

Повечето приложения на стандарта *POSIX*  оферти и незадължителни операции. Така че *Unix SVR4*  тези операции могат да задават синхронно или забавено записване и т.н.

caddr\_t mmap (caddr\_t addr, size\_t len, int prot, int знамена, int дръжка, off\_t офсет)

Показване на част от файл във виртуалното адресно пространство на процеса.Проектният параметър задава права за достъп до показаната част: за четене, писане и изпълнение. Съпоставянето може да се осъществи на посочения виртуален адрес или системата може да избере самия адрес на картографиране.

Още две операции се извършват не във файла, а по неговото име: това са операции за преименуване и изтриване на файл. В някои системи, например в системите на семейството *Unix*, файлът може да има няколко имена и има само системно повикване, за да изтриете името й. Файлът се изтрива, когато се изтрива последното име.

Вижда се, че наборът от операции на файл в този стандарт е много подобен на набора от операции на външно устройство. И двете се третират като неструктурирани байтови потоци. За пълнота, трябва да се каже, че основното средство за комуникация между процесите в семейните системи *Unix* (**тръба**) също представлява неструктуриран поток от данни. Идеята, че повечето действия на предаване могат да бъдат сведени до поток от байтове, е доста стара, но *Unix*  беше една от първите системи, в които тази идея беше доведена до логично заключение.

Приблизително един и същ модел за обработка на файлове е приет *CP/ М*и набор от повиквания за файлова система *MS DOS*  действително копирани от повикванията *Unixv7*, На свой ред *OS/2*и *Windows NT* наследи принципите на работа с файла директно от *MS DOS*.

За разлика от системите без *Unix*  в родословието може да се използва малко по-различно тълкуване на понятието за файл.Най-често файлът се третира като набор от записи. Обикновено системата поддържа записи както на постоянна дължина, така и на променлива. Например, текстов файл се интерпретира като файл с записи с променлива дължина, а един ред съответства на всеки ред текст. Това е моделът за работа с файлове в *VMS*и в линията OS *OS/360*-*MVS*  IBM фирми.

Една от компонентите на операционната система е файловата система - основното хранилище на системна и потребителска информация. Всички модерни операционни системи работят с една или няколко файлови системи, например FAT (таблица за разпределение на файлове), NTFS (NT файлова система), HPFS (високопроизводителна файлова система), NFS (мрежова файлова система), AFS (файлова система Андрю), интернет Файлова система.

Файловата система е част от операционната система, чиято цел е да предостави на потребителя удобен интерфейс при работа с данни, съхранявани във външна памет, и да осигури споделяне на файлове от няколко потребители и процеси.

В по-широк смисъл понятието "файлова система" включва:

Събиране на всички файлове на диска;

Комплекти структури от данни, използвани за управление на файлове, като например файлови директории, файлови дескриптори, таблици за разпределяне на свободно и използвано дисково пространство;

Набор от системни софтуерни инструменти, които прилагат управление на файлове, по-специално: създаване, унищожаване, четене, писане, именуване, търсене и други файлови операции.

Файловата система се използва обикновено както когато операционната система се стартира след включване на компютъра, така и в процеса. Файловата система изпълнява следните основни функции:

Определя възможните начини за организиране на файлове и файлови структури в медиите;

Реализира методи за достъп до съдържанието на файловете и осигурява инструменти за работа с файлове и файлова структура. В същото време достъпът до данни може да бъде организиран от файловата система както по име, така и по адрес (номер на сектор, повърхности и медийни песни);

Проследява свободното пространство върху медиите.

Когато дадена приложна програма достигне до файл, тя няма представа как информацията се намира в конкретен файл, както и какъв физически тип носител (CD, твърд диск или блок от флаш памет) е записан. Всичко, което програмата знае, е името на файла, неговия размер и атрибути. Той получава тези данни от драйвера на файловата система. Това е файловата система, която установява къде и как ще бъде записан файлът на физически носител (например твърд диск).

От гледна точка на операционната система целият диск е набор от клъстери (секции с памет) с размер от 512 байта или повече. Драйверите на файловата система организират клъстери във файлове и директории (които всъщност са файлове, съдържащи списък с файлове в тази директория). Същите драйвери следят кои клъстери се използват, кои са свободни и кои са маркирани като неуспешни. За да се разбере ясно как се съхраняват данните на дисковете и как операционната система осигурява достъп до тях, е необходимо да се представят, поне в общи линии, логическата структура на диска.

**3.1.5 Логическа структура на диска**

За да може компютърът да съхранява, чете и пише информация, твърдият диск трябва да бъде предварително маркиран. На него се създават дялове с помощта на подходящи програми - това се нарича "разделен твърд диск". Без това маркиране на твърдия диск няма да е възможно да се инсталира операционната система (въпреки че Windows XP и 2000 могат да бъдат инсталирани на непрекъснат диск, но самите те извършват такава маркировка по време на инсталационния процес).

Твърдият диск може да бъде разделен на няколко секции, всеки от които ще се използва самостоятелно. За какво е? Един диск може да съдържа няколко различни операционни системи, разположени на различни дялове. Вътрешната структура на дял, разпределен за операционната система, се определя изцяло от тази операционна система.

Освен това има и други причини за разделяне на диск, например:

Възможност за използване на MS DOS дискове с капацитет по-голям от   
  32 MB;

В случай на повреда на диска изчезва само информацията, която е на този диск;

Реорганизирането и разтоварването на малък диск е по-лесно и по-бързо от голямо;

Всеки потребител може да разпредели логическото ви устройство.

Извиква се операцията по подготовка на диск за работа *форматиране*или *инициализация*, Цялото налично дисково пространство е разделено на страни, песни и сектори, с песни и страници, номерирани от нула, и сектори от една. Колекцията от песни, разположени на същото разстояние от оста на диска или диска, се нарича цилиндър. По този начин, адресът на физическия сектор се определя от следните координати: номер на коловоза (цилиндър - С), номер на диска (глава - Н), номер на сектор - R, т.е. CHR.

Първият сектор на твърдия диск (C = 0, H = 0, R = 1) съдържа главния зареждащ запис **–***Главен зареждащ запис*, Този запис не заема целия сектор, а само неговата първоначална част. Master Boot Record е програма за зареждане извън системата.

В края на първия сектор на твърдия диск има таблица с дялове - *Разделителна таблица*, Тази таблица съдържа четири реда, описващи максимум четири секции. Всеки ред в таблицата описва един раздел:

1) активен дял или не;

2) номера на сектора, съответстващ на началото на раздела;

3) номера на сектора, съответстващ на края на раздела;

4) размер на разпределението в секторите;

5) код на операционната система, т.е. каква операционна система принадлежи към този раздел.

Разделът се нарича активен, ако съдържа програма за зареждане на операционна система. Първият байт в елемента на секцията е флагът на активността на раздела (0 - неактивен, 128 (80H) - активен). Той се използва, за да се определи дали дялът е системен (boot), и да се наложи да стартира операционната система от нея, когато компютърът се стартира. Само един дял може да бъде активен. Малките програми, наречени начални устройства за стартиране (Boot Manager), могат да бъдат разположени в първите сектори на диска. Те интерактивно изискват от потребителя, от коя секция да зареди и коригира флаговете за активност на секциите. Тъй като в таблицата за разделяне има четири реда, на диска може да има до четири различни операционни системи, следователно дискът може да съдържа няколко първични дяла, принадлежащи към различни операционни системи.

Пример за логическа структура на твърдия диск, състоящ се от три дяла, два от които принадлежат на DOS, а един принадлежи на UNIX, е показан на фигура 3.2а.

Всеки активен дял има свой собствен зареждащ запис - програма, която зарежда тази операционна система.

На практика дискът е разделен най-често на две части. Размерите на дялове, независимо дали са обявени за активни или не, се задават от потребителя в процеса на подготовка на твърдия диск за работа. Това се прави с помощта на специални програми. В DOS, тази програма се нарича FDISK, във версии на Windows-XX - Diskadministrator.

В DOS основният дял е **Първичен дял**Това е разделът, който съдържа товарача на операционната система и самата операционна система. Така основният дял е активният дял, използван като логическо устройство, наречено C:.

Операционната система WINDOWS 2000 (а именно WINDOWS 2000) промени терминологията: активният дял се нарича системен дял, а дяла за зареждане е логическото устройство, което съдържа системните файлове на WINDOWS. Логическият диск за зареждане може да бъде същият като системния дял, но може да бъде разположен на друг дял на същия твърд диск или на друг твърд диск.

Разширен раздел **Разширен дял**  може да се раздели на няколко логически устройства с имена от D: до Z:.

Фигура 3.2б показва логическата структура на твърдия диск, в който има само две дяла и четири логически диска.

Файловата система е начин за организиране на съхранението на данни на носители. Освен това файловата система определя дължината на името на файла, максималния размер на файла и дяла, файловите атрибути. В тази статия ще говорим за това какви са файловите системи.

Задачи, които файловата система трябва да реши:

* именуване на файлове.
* програмен интерфейс за потребителски програми.
* защита на данните от прекъсвания на захранването и хардуерни и софтуерни грешки.
* съхраняване на параметрите на файла.

Съвременните файлови системи могат да се разделят на няколко групи според предназначението им:

* Файлови системи за носители с произволен достъп (за флаш памети): FAT32, HPFS, ext2 и много други.
* Файлови системи за медии с последователен достъп (магнитни ленти): QIC и др.
* Оптични дискови файлови системи: ISO9660, HFS, UDF и др.
* Виртуални файлови системи: AEFS и др.
* Мрежови файлови системи: NFS, SSHFS, CIFS, GmailFS и др.
* Файлови системи, предназначени изключително за: YAFFS, exFAT, ExtremeFFS.

**Популярни файлови системи:**

FAT - Файловата система, разработена от Бил Гейтс и Марк Макдоналд през 70-те години на миналия век. Поради своята простота, тя все още се използва във флаш памети. Има три версии на файловата система FAT: FAT12, FAT16 и FAT32. Тези версии на файловата система FAT се отличават с битовата дълбочина на записите (броят на битовете, разпределени за съхраняване на номера на клъстера). Това означава, че колкото е по-голям битът, толкова по-голямо е количеството диск, с който може да работи файловата система FAT. Така за FAT32 максималният размер на диска е 127 гигабайта.

NTFS  - файловата система на новото поколение от Microsoft. Тази файлова система се използва за всички операционни системи на Microsoft Windows NT. За първи път NTFS е издаден през 1993 г. заедно с операционната система Windows NT 3.1. В сравнение с FAT, файловата система NTFS е получила голям брой подобрения. По този начин, границата на максималния размер на файла и диска е почти изчезнала. В допълнение, има подкрепа за твърди връзки, криптиране и компресиране.

вътр  - файлова система, създадена специално за операционни системи на ядрото на Linux. Развитието е въведено за първи път през 1992 година. Сега има няколко версии на тази файлова система: ext, ext2, ext3, ext3cow и ext4. Файловата система ext4 в момента е най-новата и актуална версия на ext, тази версия се използва от повечето модерни Linux дистрибуции.

Файловете на компютъра се създават и поставят на основата на системни принципи. Благодарение на тяхната реализация, потребителят има възможност удобно да достигне до необходимата информация, без да мисли за сложните алгоритми за достъп до нея. Как се организира файловата система? Кои са най-популярните днес? Какви са разликите между файловите системи, адаптирани за персонални компютри? А тези, които се използват в мобилни устройства - смартфони или таблети?

**Определение на файловите системи**

Според една обща дефиниция, файловата система е комбинация от алгоритми и стандарти, които се използват за организиране на ефективен достъп на потребител от компютър до данни, съхранявани на компютър. Някои експерти го смятат за част от него.Други ИТ специалисти, признавайки факта, че тя е пряко свързана с операционната система, смятат, че файловата система е независим компонент от управлението на компютърни данни.

Как се използват компютрите преди изобретяването на файловата система? Информатиката - като научна дисциплина - отчита факта, че за дълго време управлението на данните се осъществява чрез структуриране в рамките на алгоритми, вградени в конкретни програми. Следователно, един от критериите за файловата система е съществуването на стандарти, които са еднакви за повечето програми, използващи достъп до данни.

**Принципи на файловите системи**

На първо място, файловата система е механизъм, който включва използването на ресурси за компютърен хардуер. Като правило, ние говорим за магнитни или лазерни медии - твърди дискове, CD, DVD, флаш памети, които все още не са успели да остареят дискети. За да разберем как работи съответната система, нека дефинираме какъв е самият файл.

Според общоприетото определение на ИТ специалисти, това е поле за данни с фиксирана стойност, изразено в основни единици информация - байтове. Файлът се намира на диск, като правило, под формата на няколко взаимосвързани блока, които имат специфичен "адрес" на достъп. Файловата система определя същите тези координати и ги "отчита", от своя страна, операционната система. Което ясно превежда съответните данни на потребителя. Данните се обръщат към тях, за да ги прочетете, модифицирате, създадете нови. Специфичният алгоритъм за работа с "координатите" на файловете може да е различен. Тя зависи от вида на компютъра, операционната система, спецификата на съхранените данни и други условия. Следователно съществуват различни видове файлови системи. Всеки от тях е оптимизиран за използване в определена операционна система или за работа с определени типове данни.

Адаптирането на дисковия носител за използване чрез алгоритми, специфични за файловата система, се нарича форматиране. Съответните хардуерни елементи на диска - клъстерите - са подготвени за последващо писане на файлове към тях, както и тяхното отчитане в съответствие със стандартите, определени в една или друга система за управление на данни. Как да смените файловата система? В повечето случаи това може да стане само чрез преформатиране на носителя за съхранение. По правило файловете се изтриват. Съществува обаче вариант, при който, като се използват специални програми, все още е възможно, въпреки че това обикновено изисква много време, да се промени системата за управление на данните, като последната остава непокътната.

Файловите системи не работят без грешки. Може да има някои неуспехи в организацията на работата с блокове данни. Но в повечето случаи те не са критични. Като правило, няма проблеми с това как да се поправи файловата система, да се поправят грешки. В Windows за това, в частност, има вградени софтуерни решения, които са достъпни за всеки потребител. Например, например, програмата "Check Disk".

**вид**

Какви типове файлови системи могат да бъдат наречени най-често срещани? Вероятно, на първо място, тези, които се използват от най-популярната операционна система за компютри в света - Windows. Основните файлови системи на Windows са FAT, FAT32, NTFS и техните различни модификации. Наред с компютрите, смартфоните и таблетите са придобили популярност. Повечето от тях, ако говорим за световния пазар и не вземат предвид различията в технологичните платформи, се контролират от Android OS и iOS. Тези операционни системи използват свои собствени алгоритми за обработка на данни, различни от тези, които характеризират файловите системи на Windows.

**Стандартите са отворени за всички**

Отбележете, че наскоро се наблюдава известна унификация на стандартите на световния пазар на електроника по отношение на операционни системи с различни типове данни. Това може да бъде проследено в два аспекта. Първо, на различни устройства, работещи с два различни типа ОС, често се използва една и съща файлова система, еднакво съвместима с всяка ОС. Второ, модерните версии на операционната система, като правило, са способни да разпознават не само файловите системи, които са типични за тях, но и тези, които традиционно се използват в други операционни системи, като се използват вградени алгоритми и се използва софтуер на трети лица. Например, модерните версии на Linux, като правило, лесно разпознават маркираните файлови системи за Windows.

**Структура на файловата система**

Въпреки, че видовете файлови системи са представени в доста голям брой, те работят като цяло на много сходни принципи (очертахме общата схема по-горе) и в рамките на подобни структурни елементи или обекти. Помислете за тях. Какви са обектите на основната файлова система?

Един от ключовете - Това е изолирана област от данни, в която могат да се намират файлове. Структурата на директориите е йерархична. Какво означава това? Една или повече директории могат да бъдат поставени в друга. Което, от своя страна, е част от "висшия". Най-"основната" е главната директория. Ако говорим за принципите, на които работи файловата система на Windows - 7, 8, XP или друга версия - основната директория е логическо устройство, обозначено с буква - обикновено C, D, E (но можете да персонализирате всеки, който е в Английска азбука). Що се отнася до, например, OS Linux, тогава там е главната директория е магнитна медия като цяло. В тази операционна система и други операционни системи, базирани на нейните принципи - това е Android - логическите дискове не се използват. Мога ли да съхранявам файлове без директории? Да. Но това не е много удобно. Всъщност комфортът при използването на персонален компютър е една от причините за въвеждане на принципа на разпространение на данни в директориите във файловите системи. Между другото, те могат да бъдат наречени по различен начин. В Windows директории се наричат ​​папки, а в Linux - едни и същи. Но традиционните, използвани в продължение на много години името на директориите в тази операционна система - "директория". Както и в предишните Windows и Linux OS - DOS, Unix.



В средата на ИТ специалистите няма категорично становище дали даден файл трябва да се разглежда като структурен елемент на съответната система. Тези, които смятат, че това не е напълно правилно, твърдят, че системата може да съществува без файлове. Нека това от практическа гледна точка и безполезно явление. Дори ако на диска не са записани файлове, съответната система все още може да присъства. Като правило, магнитните медии, продавани в магазините, не съдържат никакви файлове. Но те вече имат съответната система. Според друга гледна точка, файловете трябва да се разглеждат като неразделна част от системите, чрез които те се управляват. Защо? И тъй като, според експерти, алгоритмите за тяхното използване са адаптирани основно за работа конкретно с файлове в рамките на определени стандарти. Въпросните системи не са предназначени за нищо друго.

Друг елемент, присъстващ в повечето файлови системи - Това е област от данни, съдържаща информация за местоположението на даден файл в определено място. Това означава, че можете да поставите етикет на едно място на диска, но е възможно да се осигури достъп до желаната област от данни, която се намира в друга част на средата. Можем да предположим, че преките пътища са пълноценни обекти на файловата система, ако е договорено, че файловете също са файлове.

Така или иначе, няма да бъде грешка да се каже, че и трите вида данни - файлове, преки пътища и директории - са елементи на съответните системи. Поне тази теза ще съответства на една от широко разпространените гледни точки. Най-важният аспект, характеризиращ как работи файловата система, са принципите за именуване на файлове и директории.

**Имена на файлове и директории в различни системи**

Ако се съгласим, че файловете са все още съставни елементи на съответните им системи, тогава си струва да се разгледа тяхната основна структура. Какво може да се отбележи първо? За удобство при организирането на достъпа до тях, повечето модерни системи за управление на данни осигуряват двустепенна структура за именуване на файлове. Първото ниво е името. Вторият е разширение. Вземете например музикалния файл Dance.mp3. Танцът е името. Mp3 - разширение. Първият е предназначен да разкрие на потребителя същността на съдържанието на файла (и за да бъде програмата ръководство за бърз достъп). Вторият показва типа на файла. Ако той е Mp3, тогава не е трудно да се досетите, че говорим за музика. Файловете с разширението Doc обикновено са документи, Jpg - снимки, Html - уеб страници.



Директории, от своя страна, имат структура на едно ниво. Те имат само име, без разширение. Ако говорим за разликите между различните типове системи за управление на данни, първото нещо, на което трябва да обърнете внимание, са принципите за именуване на файлове и директории, които се прилагат в тях. По отношение на операционната система Windows, следното. В най-популярната операционна система в света файловете могат да бъдат наименувани на всеки език. Максималната дължина обаче е ограничена. Специфичният му интервал зависи от използваната система за управление на данните. Обикновено тези стойности варират от 200 до 260 символа.

Общото правило за всички операционни системи и съответните им системи за управление на данни е, че файловете със същото име не могат да бъдат в една и съща директория. В Линукс има някаква "либерализация" на това правило. В същата директория могат да бъдат файлове със същите букви, но в различен случай. Например, Dance.mp3 и DANCE.mp3. В Windows това не е възможно. Същите правила са установени и в аспекта на поставянето на директории в други.

**Адресиране на файлове и директории**

Адресирането на файлове и директории е съществен елемент от съответната система. В Windows неговият персонализиран формат може да изглежда така: C: / Documents / Music / - това е достъп до музикалната директория. Ако се интересуваме от конкретен файл, адресът може да изглежда така: C: /Documents/Music/Dance.mp3. Защо "обичай"? Факт е, че на ниво софтуерно и хардуерно взаимодействие между компютърните компоненти, структурата на достъпа до файловете е много по-сложна. Файловата система локализира файловите блокове и взаимодейства с операционната система в по-голямата си част в рамките на операции, скрити от потребителя. Въпреки това е изключително рядко за потребител на персонален компютър да използва други "адресни" формати. Почти винаги достъпът до файловете се извършва в посочения стандарт.

**Сравнение на файловата система за Windows**

Изследвахме общите принципи на функциониране на файловите системи. Обмислете сега характеристиките на най-често срещаните типове. Windows най-често използва файлови системи като FAT, FAT32, NTFS и exFAT. Първият в тази серия се счита за остарял. В същото време, за дълго време, това беше един вид флагман на индустрията, но тъй като PC стана по-технологичен, неговите възможности вече не отговаряха на изискванията на потребителите и изискванията за ресурси от софтуера.



Проектиран да замени FAT файловата система - това е FAT32. Според много ИТ специалисти, сега тя е най-популярната, ако говорим за PC пазара с Windows. Най-често се използва при съхраняване на файлове на твърди дискове и флаш памети. Също така може да се отбележи, че тази система за управление на данните се използва често в модулите на паметта на различни цифрови устройства - телефони, камери. Основното предимство на FAT32, които разграничават IT-експертите, така че е гъвкавост. Независимо от факта, че тази файлова система е създадена от Microsoft, повечето модерни операционни системи, включително тези, инсталирани на определени видове цифрови технологии, могат да работят с данните в рамките на вложените в нея алгоритми.

В системата FAT32 има редица недостатъци. На първо място, можем да отбележим границата на размера на един файл - не може да бъде повече от 4 GB. Също така, в системата FAT32 е невъзможно да се използват вградените инструменти на Windows, за да се определи логическо устройство, чийто размер ще бъде повече от 32 GB. Но това може да стане чрез инсталиране на допълнителен специализиран софтуер.

Друга популярна система за управление на файлове, разработена от Microsoft, е NTFS. Според някои ИТ специалисти той надминава FAT32 в повечето параметри. Но тази теза е вярна, ако говорим за работата на компютър с Windows. NTFS не е толкова гъвкав, колкото FAT32. Характеристиките на неговото функциониране правят използването на тази файлова система не винаги удобно, по-специално в мобилните устройства. Една от ключовите предимства на NFTS е надеждността. Например, в случаите, когато захранването на твърдия диск е внезапно изключено, вероятността от повреда на файловете е сведена до минимум, благодарение на алгоритмите за дублиране на достъп до данни, предоставени от NTFS.

Една от най-новите файлови системи на Microsoft е exFAT. Той е най-подходящ за флаш памети. Основните принципи на работа в него са същите като в FAT32, но има и значителен ъпгрейд в някои аспекти: например, няма ограничения за размера на един файл. Въпреки това, системата exFAT, както е отбелязано от много ИТ специалисти, е сред тези, които имат ниска гъвкавост. На компютри, работещи с операционна система, различна от Windows, работата с файлове с използване на exFAT може да бъде трудна. Освен това, дори и в някои версии на самата Windows, като например XP, данните на дисковете, форматирани с използване на exFAT алгоритми, може да не са четливи. Инсталирането на допълнителния драйвер ще се изисква.

Имайте предвид, че поради използването на доста широка гама от файлови системи в операционната система Windows, потребителят може да изпита периодични затруднения по отношение на съвместимостта на различни устройства с компютъра. В някои случаи, например, трябва да инсталирате драйвера за файловата система WPD (Windows Portable Devices - технологията, използвана при работа с преносими устройства). Понякога може да не е достъпно за потребителя, в резултат на което външната операционна система може да не бъде разпозната. Файловата система WPD може да изисква допълнителен софтуер, за да се адаптира към работната среда на конкретен компютър. В някои случаи потребителят ще бъде принуден да се свърже с ИТ специалистите, за да реши проблема.

Как да определите коя файлова система - exFAT или NTFS, и може би FAT32 - е оптимална за използване в специфични случаи? Препоръките на IT-специалистите като цяло са следните. Можете да използвате два основни подхода. Според първия, трябва да се прави разлика между типичните файлови системи на твърдия диск, както и тези, които са по-добре адаптирани към флаш памети. FAT и FAT32, както смятат много експерти, са по-подходящи за "флаш дискове", NTFS - за твърди дискове (поради технологичните характеристики на работа с данни).

При втория подход размерът на носителя е важен. Ако говорим за използването на сравнително малко количество диск или флаш устройство, можете да ги форматирате в системата FAT32. Ако дискът е по-голям, можете да опитате exFAT. Но само ако не е предназначена за използване на медиите на други компютри, особено тези, които не разполагат с най-новите версии на Windows. Ако говорим за големи твърди дискове, включително външни, тогава те трябва да бъдат форматирани в NTFS. Това са приблизително критериите, по които може да се избере оптималната файлова система - exFAT или NTFS, FAT32. Това означава, че всеки от тях трябва да се използва, като се има предвид големината на носителя, неговия тип, както и версията на операционната система, на която основно се използва устройството.

**Файлови системи Mac**

Друга популярна хардуерна и софтуерна платформа на световния компютърен пазар е Macintosh на Apple. Компютрите от тази линия работят с Mac OS. Какви са възможностите за организиране на работата с файлове на компютри Mac? Най-напредналите компютри на Apple използват файловата система Mac OS Extended. По-рано Macs обработваха данни в съответствие със стандартите на HFS.



Основното нещо, което може да се отбележи в аспекта на неговите характеристики: на диска, който се управлява от файловата система Mac OS Extended, могат да се поставят много големи файлове - можем да говорим за няколко милиона терабайта.

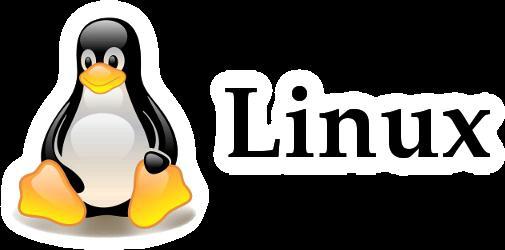
**Файлова система в устройства с Android**

Най-популярната операционна система за мобилни устройства - форма на електронна технология, която не е по-ниска по популярност от компютър - е Android. Как се управляват файловете на устройства от подходящ тип? Отбелязваме на първо място, че тази операционна система всъщност е „мобилна“ адаптация на операционната система Linux, която благодарение на своя отворен код може да бъде модифицирана с перспективата да бъде използвана на най-широк кръг устройства. Следователно, управлението на файлове в мобилни устройства, работещи с Android, се извършва като цяло според същите принципи, както в Linux. Някои от тях отбелязахме по-горе. По-специално, управлението на файловете в Linux се извършва без разделянето на медиите на логически дискове, както се случва в Windows. Какво друго е интересно за файловата система Android?



Основната директория в Android, като правило, е областта за данни, наречена / mnt. Съответно, адресът на желания файл може да изглежда по следния начин: /mnt/sd/photo.jpg. Освен това има и друга характеристика на системата за управление на данни, която се прилага в тази мобилна операционна система. Факт е, че флаш паметта на устройството обикновено се класифицира в няколко секции, като например System или Data. В същото време първоначално определеният размер на всеки от тях не може да бъде променян. Приблизителна аналогия по отношение на този технологичен аспект може да се намери, като се помни, че е невъзможно (ако не използвате специален софтуер) да промените размера на логическите дискове в Windows. Тя трябва да бъде фиксирана.

Друга интересна особеност на организацията на работа с файлове в Android - съответната операционна система, като правило, записва нови данни в определена област на диска - Data. Работата, например, със секцията Система не се извършва. Следователно, когато потребителят активира функцията за нулиране на софтуерните настройки на смартфона или таблета на ниво "фабрика", на практика това означава, че тези файлове, които са записани в областта за данни, просто се изтриват. Разделът System, като правило, остава непроменен. Освен това, потребителят не може да извършва каквито и да е корекции на съдържанието в Системата, без да притежава специализиран софтуер. Процедурата за актуализиране на системната медийна област в Android устройството се нарича мигаща. Това не е форматиране, въпреки че и двете операции често се извършват едновременно. Като правило мигането се използва за инсталиране на по-нова версия на Android OS на мобилно устройство.



По този начин ключовите принципи, на базата на които работи Android файловата система, са липсата на логически устройства, както и твърда разлика между достъпа до системни и потребителски данни. Това не означава, че този подход е коренно различен от това, което се прилага в Windows, обаче, както смятат много ИТ специалисти, операционната система на Microsoft има малко повече свобода за потребителите да работят с файлове. Въпреки това, както смятат някои експерти, това не може да се счита за определено предимство на Windows. “Либералният” режим в аспекта на управлението на файловете включва, разбира се, не само потребителите, но и компютърните вируси, на които Windows е много податлив (за разлика от Linux и неговата “мобилна” реализация под формата на Android). Това, според експертите, е една от причините, че има толкова малко вируси за Android устройства - чисто от технологична гледна точка, те не могат да функционират напълно в операционна среда, базирана на принципите на строг контрол на достъпа до файлове.

Запишете информация за файловете, създадени в специална област на диска. Като се има предвид зависимостта на компютърния хардуер и възможностите на инсталираната операционна система, се използват различни файлови системи за организиране на работата.

**Файлова система**(*FAT (таблица за разпределение на файлове) - таблица за разпределение на файлове*) - обща структура, която определя името, записа и местоположението на файловете в операционната система. От файловата система зависи от правилата за именуване на файлове, начините за достъп до файловете и начина на работа с тях.

Всички съвременни дискови операционни системи осигуряват създаването на файлова система за съхранение на данни на дискове и осигуряване на достъп до тях. Принципът на организация на файловата система зависи от операционната система. Най-често срещаният тип е табличен.

Дискът е представен като набор от повърхности. Дискети имат всички, има два от тях (горни и долни), но твърдите дискове всъщност са "e-stackers", състоящи се от няколко плочи, затова имат повече повърхности.

Всяка повърхност на диска е разделена на пръстеновидни и всяка песен в сектори. Размерите на секторите са фиксирани и са равни на 512 байта.

**сектор**  - това е най-малката единица за съхранение на данни, но за адресиране тя се използва далеч от всички файлови системи. За това тя е твърде малка. Операционни системи като MS-DOS, Windows, OS / 2 използват по-голяма единица за съхранение, наречена ***струпване****.*Клъстерът е група от sos сектори. Размерът на клъстера зависи от размера на твърдия диск. Колкото по-голям е дискът, толкова по-голям е размерът на клъстера. Типични стойности са 8, 16, 32 или 64 сектора.

**струпване**  - минималното количество дисково пространство, което трябва да бъде разпределено за поставяне на файла. Всички файлови системи, използвани от Windows за работа с твърди дискове, се основават на клъстери, които се състоят от един или повече съседни сектори. Колкото по-малък е размерът на клъстера, толкова по-ефективно се използва дисковото пространство. Когато форматирате диск, размерът на клъстера не е изрично посочен, Windows избира една от стандартните стойности въз основа на размера на тома. Стандартните стойности се избират по такъв начин, че да се намали загубата на дисково пространство и степента на възможна фрагментация на обема. Размерът на клъстера също се нарича единица за разпределение на паметта.

В файловата система **FAT**данните за кой дисков клъстер започва даден файл се съхранява в специалната област на системния диск *таблици за разпределение на файлове*(*FAT*- таблици). От нарушението *FAT*-Таблиците правят невъзможно използването на записаните на диска данни, на които се налагат специални изисквания за надеждност и то съществува в две копия, чиято идентичност се контролира редовно от средствата на операционната система.

**FAT16 файлова система.**  Тази система е съществувала преди MS-DOS. Максималният размер на поддържаното дисково пространство не надвишава 4096 MB, FAT16 не поддържа големи обеми. FAT16 използва 16-битово адресиране и съответно може да се използват до 2 адреса. Обемът на паметта, форматиран от FAT 16, е разделен на клъстери. Размерът на клъстера зависи от размера на обема и варира от 512 байта до 64 Kbytes, като се серия от дефинирани стойности. Организирана от файловата система FAT16, тя включва зареждащ диск, FAT16 оригинал, FAT16 копие, основната директория, директории и файлове. Разликата между главната директория и всички други поддиректории е фиксиран брой записи (обикновено 512). Този номер е равен на общия брой поддиректории и файлове, създадени в главната директория.

**FAT32 файлова система.** Започвайки с Windows 95, се появи FAT32, който може да обслужва томове до 2 TB с размер на клъстера до 32 KB. Като цяло, размерите на клъстерите в FAT32 са по-малки от съответните размери в FAT16. Това води до по-ефективно използване на дисковото пространство. В същото време максималният брой записи в основната директория е увеличен до 65535. FAT32 използва 32-битово адресиране, но първите четири бита на таблицата с местоположението на файла FAT32 са необходими за техните собствени нужди.

**NTFS файлова система.**Windows 2000 включва поддръжка за новата версия на файловата система NTFS (Нова технологична файлова система). Ключовото предимство на NTFS е възможността за ограничаване на достъпа до файлове и папки. При оформянето на NTFS файловата система се създава MTF (Master File Table) файл, в който се съхраняват адресите на копията на данните. Пълно копие на сектора за зареждане се намира в края на тома. В MTF, освен това, има таблица с име на атрибут, главната директория и т.н.Ако файлът има прекалено голям набор от атрибути, информацията за нея се съхранява в няколко записа, като първият (основния) запис води до адресите на други записи.

**Сравнение на файловите системи FAT16, FAT32 и NTFS.**  Числата в имената на файловите системи FAT16 и FAT32 показват броя на битовете, необходими за съхраняване на информация за номерата на клъстерите, използвани от файла, т.е. ширината на адреса. Нека сравним тези файлови системи, като посочим техните предимства и недостатъци.

FAT16 има следното ***ползите***:

1) тази файлова система се поддържа от всички ОС и ОС, включени в софтуера на Windows и някои версии на UNIX OS;

2) са натрупани голям брой програми за коригиране на грешки в тази файлова система и възстановяване на данни;

3) системата трябва да бъде заредена от системната дискета;

4) тази файлова система е много ефективна за обеми на паметта, по-малки от 256 MB.

K ***недостатъци на FAT16***  включват:

1) системата не поддържа резервно копие на сектора за зареждане;

2) FAT 16 не поддържа вградена защита на файлове и компресиране на файлове.

**Предимства на FAT32**  те са:

1) за големи дискове разпределеното дисково пространство се използва по-ефективно;

2) основната директория в FAT32 заема верига от клъстери и може да бъде разположена навсякъде на диска, така че системата не налага никакви ограничения върху броя на елементите (събития) в главната директория;

3) поради по-малкия размер на клъстера, заеманото дисково пространство е с 10-15% по-малко от това на FAT 16;

4) FAT32 поради възможността за използване на резервно FAT е по-надеждна система от FAT 16.

основен ***недостатъци на FAT32***:

1) размерът на обема на паметта под Windows 2000 е ограничен до 32 GB;

2) обемите не са достъпни за други операционни системи с изключение на Windows 95 и Windows 98;

3) не се поддържа резервно копие на сектора за зареждане;

4) Вградената защита на файлове и компресирането на файлове не се поддържат.

NTFS файловата система има няколко функции, които не са реализирани на FAT16 и FAT32 системи. В сравнение с тези файлови системи няма явни недостатъци. Отбелязваме само неговите допълнителни функции:

а) възможността за възстановяване на информация. NTFS гарантира целостта на данните чрез поддържане на протокол и някои вградени алгоритми за възстановяване на данни;

б) компресия на данни. Когато четете, файлът се разопакова автоматично, когато затворите и запазите, файлът се пакетира отново;

в) защита на файлове и директории чрез задаване на атрибути за достъп;

г) поддръжка на резервно копие на сектора за зареждане (в края на обема на паметта);

д) подкрепа за криптиране на съдържанието на файла.

Тази файлова система работи най-ефективно с големи файлове, а недостатъкът на системата е, че не се поддържа от MS-DOS, Windows 95 и Windows 98.

Преди появата на операционната система Windows 95, обичайният метод за именуване на файлове на компютри на IBM PC беше ***съгласие* *8.3*.**  Според това споразумение, прието в *MS-DOS,*името на файла се състои от две части: действителната *името*и *разширения на имена.*Името на файла е присвоено 8 знака и 3 знака за неговото разширение. Името на разширението е разделено с точка. Името и разширението могат да включват само букви и цифри от латинската азбука.

*Споразумение 8.3*това не е стандарт, поради което в някои случаи отклоненията от правилната форма на запис се допускат както от операционната система, така и от нейните приложения. Така например, в повечето случаи системата „не възразява“ срещу използването на някои специални символи (удивителен знак, долна черта, тире, тилда и т.н.) и някои версии *MS-DOS*дори позволяват използването на руски и други азбуки в имената на файловете.

Имената на файловете днес са записани в съответствие с *споразумение 8.3,*счита за кратък.

Основният недостатък на кратките имена е ниското им съдържание. Далеч от всички ние можем да изразим характеристиката на файл с няколко знака, и в тази връзка с появата на операционната система Windows 95, беше въведена концепцията за дълго име. Това име може да съдържа до 256 знака. Това е достатъчно, за да създадете значими имена на файлове. Дългото име може да съдържа всякакви знаци, освен девет специални: \\ t "< >  Името може да използва интервали и множество точки. Разширението на името се счита за всички символи, които следват последната точка.

Файлова система - концепция и типове. Класификация и характеристики на категорията "Файлова система" 2017, 2018.

Да кажем че трябва да си създадем файлова система в линукс командата е mkfs

Първо ще се запознаем с още малко теория преди да пристъпим към практиката

Създаването на файлови системи върху дисковите дялове (или казано с термините на DOS/Windows, форматирането на дяловете) е вторият етап от подготовката на диска за инсталация на Linux. Само по себе си това действие не е от най-сложните, но осъзнатото му изпълнение изисква някаква подготовка.

**УВОД**

Терминът “файлова система” е един от най-многозначните в компютърната терминология въобще и в литературата, свързана с операционната система Unix и нейните клонове (a Linux се отнася именно към тях) в частност. Голямо значение има по-точно думата “система”. Нещата са се развили исторически по този начин и затова се налага да се определя значението на термина според контекста. На мен са ми известни поне пет значения, като не съм сигурен, че това са всички възможни значения.

Първо, файлова система или понякога подсистема наричат частта от ядрото, която управлява организацията на файловете и операциите над тях. В този аспект, заедно с файловата система роля играе и системата за управление на процесите и системата за вход/изход. Между другото, именно това значение ние засега няма да разглеждаме.

Второ, специфично за Unix и Unix-подобните операционни системи, файловата система е универсален интерфейс за достъп до всички ресурси както на локалната машина, така и на машини, обединени в мрежа по произволен начин – от модемна връзка до спътников канал. Именно използването на такъв интерфейс в някоя операционна система е една причините за причисляването и към семейството на Unix-подобните.

Трето, файловата система е метод за описание на някакво физическо устройство (обикновено дисков дял). Именно това значение се използва в термина “идентификатор на типа файлова система”, за който се говореше в статията за [**дисковите дялове**](http://www.linux-bg.org/cgi-bin/y/index.pl?page=article&id=advices&key=365142949). Обикновено той е специфичен за конкретната операционна система и затова тук са уместни термините “DOS-дял”, разширен (extended) дял на DOS, дял Linux native и така нататък. Макар че повечето истински операционни системи могат по един или друг начин да познаят “нестандартните” за тях идентификатори и да се обръщат към тях за данни. А в Linux като “вградени” (native) се разглеждат дисковите дялове, идентифициращи цял ред файлови системи в третото значение на термина – от наистина вградената Linux native до extended DOS, включително и логическите (LVM) дялове, за които ще стане в съответните статии.

Тук трябва да отбележим, че покрай физическите файлови системи (тези, които надстройват реалните дискови устройства, защото в този контекст нерядко фигурира израза disk based file system), съществуват и виртуални файлови системи от различен тип. Към тях спадат файловата система на устройствата devfs, временната файлова система в оперативната памет tmpfs и procfs – системата, представяща процесите във вид на файлова система (съжалявам за тавтологията). Понякога се използват и файлови системи от междинен тип, например виртуални дискове (RAM дискове). Подобно на tmpfs, те съществуват само в оперативната памет, но във всичко останало имат поведението на физическите (disk based) файлови системи.

Четвърто, под файлови системи се разбира вътрешната управляваща структура, позволяваща да съхраняваме, идентифицираме и намираме данни, както и да ги манипулираме. Такива структури, от една страна, са специфични за операционните системи, като FAT16 (с всичките нейни вариации VFAT или FAT32) за DOS, UFS за FreeBSD или ext2fs за Linux. От друга страна, структурите за управление на файловете в редица операционни системи се построяват по близки принципи, като най-ярки примери за това са файловите системи на Unix и Unix-подобните операционни системи. И поради което, те могат да бъдат обединени в едно семейство, противопоставящо се, например, на FAT семейството.

Освен това, Linux в момента може да работи с управляващи структури от различен тип – от ext3fs, представляваща надстройка на традиционната ext2fs, до XFS и JFS, първоначално разработени за Unix съответно от SGI и IBM, както и ReiserFS. Няма забрана и за разполагането на Linux на файлова система FAT (макар също така да няма и причини за това).

Ще добавя, че в списъка, изброен в горния параграф са включени само файлови системи, способни да носят основните компоненти на Linux, отговорни за нейното стартиране и минимална функционалност. Колкото до обмена на данни – той е възможен от Linux практически с всички известни файлови системи, макар че с някои от тях (примерно NTFS или HPFS) обменът на данни е възможен само в режим на четене.

И пето, последно, файловата система в Unix е и логическа структура на директории и файлове, която обединява и физически и виртуални файлови системи от най-различен тип (например дискови дялове с ext2fs и FAT16, виртуалните procfs, devfs и tmpfs), при това не само от локалната машина, но и на всяка отдалечена машина. Структурата е йерархична (дървовидна), започваща с коренна директория (/), която се явява родител на всичко останало, от която се разклоняват отделни файлове и дъщерни директории, които на свой ред могат да бъдат родителски по отношение на поддиректории от по-дълбоки нива на влагане.

Положението на нещата в момента е такова, че в Linux, структурата на файловата система обикновено е специфична за конкретната дистрибуция или група дистрибуции, свързани с общ произход. Затова често се срещат изрази като файловата система на Red Hat или Debian. Именно тези исторически създали се различия в йерархията на директориите се явяват единият от критериите, по които се обособяват различните линии дистрибуции на Linux, както и потенциална причина за тяхната несъвместимост. Но може да се надяваме, че усилията на стандартизиращите организации, като [**Linux Standard Base**](http://www.linuxbase.org/) и [**Filesystem Hierarchy Standard**](http://www.pathname.com/fhs), (руският превод[**[1]**](http://www.linux-bg.org/cgi-bin/y/index.pl?page=article&id=advices&key=365258709#sdfootnote1sym) на стандарта е на сайта на [**Виктор Костромин**](http://rus-linux.net/)), ще се увенчаят с успех и ще може да говорим за единна логическа файлова система на Linux, подобна на тази, която е в операционните системи от семейството на BSD.

В контекста на настоящата статия, нас ни интересува само четвъртия аспект на файловите системи – създаването на управляващи файловете структури, базирани на дискове (по-точно на техните дялове).

**УСТРОЙСТВО НА ФАЙЛОВИТЕ СИСТЕМИ ОТ СЕМЕЙСТВОТО НА UNIX**

В този раздел ще говорим за неща, общи за всички Unix операционни системи. Всички файлове в Unix физически се състоят от две части, реално разположени в различни блокове на диска, но задължително намиращи се на един и същ дисков дял, бил той първичен или логически.

Първата част на файла, неговите така наречени метаданни, които съдържат файловия дескриптор (някакво уникално число), сведения за неговите атрибути (принадлежност, права за достъп, време на модификация и т. н.), а също и информация за това, в кои блокове на дисковия дял (които така и се наричат – блокови данни) физически се намира съдържанието на файла – последователностите от байтове, които образуват достъпен за потребителя ASCII текст или изпълним модул на програма.

Метаданните за всеки файл са записани в специална област на диска, наречена суперблок, където образуват така наречените inodes (от information nodes – информационни възли). На всеки съществуващ файл отговаря собствен inode и именно той еднозначно се идентифицира с файловия дескриптор. Сам по себе си, списъкът с тези информационни възли, отговарящи както на съществуващи файлове, така и на свободни блокове от дисковият дял, определя границите на файловата система, тоест колко файла могат да бъдат създадени в нея.

По този начин, същността на процеса на създаване на файлова система на един дисков дял (или казано с терминологията на DOS/Windows – неговото форматиране) е създаването на този суперблок (в някои файлови системи – на няколко негови копия), списъка с информационните възли (inodes) и отделянето на дисково пространство за блоковете с данни (а също и за блок за зареждане, за който ще бъде споменато по-долу), а устройството на тези дискови области се определя от различията между файловите системи от различен тип. В резултат на това, на новия дял се създава един единствен файл – коренната директория (за дадената файлова система) на дяла (в някои случаи се създава още една директория, /lost+found, предназначена за съхраняване на повредени или загубени файлове).

Възниква въпроса, защо такъв неотменим атрибут на файла, като неговото име, не се намира нито в неговите метаданни, нито даже в неговите данни? Отговорът е прост – в Unix името на файла се явява само по себе си не атрибут на файла, а на файловата система (в петото, логическо разбиране на термина). И за съхранение на имената на файловете са предназначени файлове от особен тип – директориите (в Unix има и други типове файлове, например споменатите преди файлове на устройства). Те представляват просто списък на файловите дескриптори на идентификаторите и отговарящите им имена на файлове. Затова, идващата от MacOS и активно използвана във Windows метафора директория като папка с документи, в Unix само замъглява същността на нещата – тук по-скоро това е каталожно чекмедже в библиотека.

Независимо от простото устройство, ролята на директориите във файловата система на Unix е трудно да се оцени – имената на файловете, чрез които те се включват във файловата система (и чрез които потребителя получава достъп до тяхното съдържание), фигурират само в състава на директорията, към която файла е описан и никъде повече в системата. По този начин отстраняването на името на файла (или поддиректорията) от списъка, представляващ данните на неговата родителска директория (която, разбира се, има собствен информационен възел inode и файлов дескриптор, свързани към директорията, разположена едно ниво нагоре в йерархията на файловата система и така нататък) е равносилно на това, метаданните на файла да станат недостъпни, а свързаните към неговия информационен възел блокове с данни да се маркират като свободни. Именно по този начин се осъществява отстраняването на файловете от командата rm или от файлов мениджър от типа на Midnight Commander[**[2]**](http://www.linux-bg.org/cgi-bin/y/index.pl?page=article&id=advices&key=365258709#sdfootnote2sym).

Само че нас сега ни интересува точно обратното – как да направим достъпна файловата система. От казаното досега става ясно, че за тази цел тя със цялото си съдържание (суперблока, списъка с информационните възли, блоковете данни) трябва да бъде включена в състава на някоя от съществуващите директории, наречена точка на монтиране. Именно това е същността на процеса на монтиране. Резултатът за монтируемите файлови системи е, че коренната директория, която досега е била без име получава името на директорията – точка на монтиране (mount point), съдържанието на която представлява списък от имената на нейните файлове и поддиректории.

**ФАЙЛОВИТЕ СИСТЕМИ НА LINUX**

**Ext2fs**

До неотдавна списъкът на поддържаните (native) файлови системи за Linux беше ограничен до една единствена – ext2fs (вярно е, че Linux може да се зареди и работи с FAT дял, но затова даже не ми се говори). Названието на тази файлова система се разшифрова като “втора разширена файлова система” – “разширена” в сравнение с файловата система на операционната система Minix, послужила за прототип на Linux, “втора” – защото ранните версии на Linux използваха extfs с по-ограничени възможности.

За файловата система ext2fs е писано немалко (гледайте допълнителните източници). Затова ще отбележа само, че по начина на организиране на съхранението на данните, тя e типичен представител на файловите системи от семейството на Unix. Нейна отличителна особеност е наличието на няколко копия на суперблока, което увеличава надеждността на съхранение на данните. Освен това, тя има характерен, много ефективен механизъм за кеширане на дисковите операции, което осигурява забележителното им бързодействие – тя е една от най-бързите между известните ми файлови системи. Обратната страна на това бързодействие е относително ниската устойчивост при аварийно завършване на работата (вследствие на блокиране на компютъра или спиране на захранването), тъй като отложеният запис на изменението на файловете увеличава вероятността за нарушаване на връзката между техните информационни възли и блоковете с данни.

Разбира се, времената, когато некоректното спиране на машини с Linux можеше да доведе до пълно разрушаване на файловата система, останаха в далечното минало. Но във всеки случай, спирането на системата без стандартно демонтиране на файловите системи води до това, че в тях не се установява така наречения “бит за чисто демонтиране”. А без него, програмите за обслужване на диска (такива, като програмата за проверка fsck) при презареждането на системата няма да ги възприемат като цели и ще започне проверка, която при съвременните обеми на дисковете може да отнеме доста време.

**За журналните файлови системи**

Проблемът с нарушаването цялостта на файловата система при некоректно завършване на работата в по-голяма или по-малка степен е характерен за всички операционни системи от семейството на Unix. И затова от много време в тях се разработват така наречените журнални файлови системи. Дневникът[**[3]**](http://www.linux-bg.org/cgi-bin/y/index.pl?page=article&id=advices&key=365258709#sdfootnote3sym) е нещо подобно на log-файл на дисковите операции, в който се отбелязват не извършените, а предстоящите действия с файловете, вследствие на което е възможно самовъзстановяването на цялостта на файловата система след срив.

За избягване на недоразумения, трябва да подчертая, че протоколирането е насочено към осигуряване цялостта на файловата система, но в никакъв случай не гарантира съхраняването на потребителските данни като такива. Така, че не трябва да очаквате, че протоколирането по вълшебен начин ще възстанови незапазените преди срива промени в документа, зареден в текстовия редактор.

Нещо повече, в повечето журнални системи се фиксират бъдещите операции само над метаданните на променяните файлове. Обикновено това е достатъчно за съхраняване на цялостта на файловата система и във всеки случай предотвратява дълготраещите проверки, но не предотвратява загубите на данни при аварийни ситуации. В някои от файловите системи е възможно протоколиране на действията и в областта за данни на файла. Но, както винаги, повишаването на надеждността е за сметка на намаляване на бързодействието.

Текущите версии на ядрото на Linux подържат четири вградени журнални файлови системи – ReiserFS и ext3fs, специфични за нея и XFS и JFS – като резултат от портването в Linux на файловите системи, първоначално разработени за работни станции под управлението съответно на операционните системи Irix (SGI) и AIX (IBM). Широко признание получиха само първите три системи, поради което засега няма да говоря за JFS.

**ReiserFS**

Файловата система ReiserFS е исторически първата за Linux от журналните – тя се поддържа от каноничните ядра ([**http://www.kernel.org/**](http://www.kernel.org/)) още от първите версии на клона 2.4.х (в момента вече съществуват кръпки, позволяващи използването на ReiserFS с ядра от клона 2.2.х). И беше единствената, разработена от нулата специално за тази операционна система от Ханс Райзер и неговата фирма [**Namesys**](http://www.namesys.com/). Както и в повечето разгледани файлови системи, и в ReiserFS се осъществява протоколиране само на операциите над метаданните на файловете. При определено намаляване на надеждността се осигурява висока производителност – според мен, при повечето потребителски задачи тя отстъпва незначително на ext2fs. А при такава обикновена операция, като копирането на голямо количество малки файлове, значително я изпреварва.

Освен това, ReiserFS притежава уникалната (и по подразбиране активна) възможност за оптимизиране на дисковото пространство, заемано от малки, по-малки от един блок, файлове (а трябва да помним, че във всяка Unix система, такива файлове има в изобилие) – те се съхраняват изцяло в своите информационни възли (inodes), без да се отделят дискови блокове в областта за данни – заедно с икономията на място, това води и до увеличаване на производителността, тъй като и метаданните, и данните (в света на термините на ReiserFS – stat-data) се съхраняват в непосредствена близост и може да се смятат за една входно-изходна операция.

Втората особеност на файловата система ReiserFS е, че така наречените опашки на файловете, тоест техните крайни части, които са по-малки по размер от един блок, могат да бъдат опаковани. Този режим (tailing) също се включва по подразбиране при създаването на файловата система и осигурява около 5% икономия на дисково пространство. Което донякъде намалява бързодействието на системата, поради което режима на опаковане може да бъде отменен при монтирането на файловата система. Но при прекомпилиране на ядрото, този режим се възстановява автоматично, което, както ще бъде споменато малко по-надолу, изисква внимателно отношение.

ReiserFS не е съвместима с ext2fs на ниво обслужващи програми на файловата система. Но съответният инструментариум за това, обединен в пакета reiserfsprogs, отдавна е част от стандартният комплект на съвременните дистрибуции (или в краен случай може да бъде изтеглен от сайта [**Namesys**](http://www.namesys.com/)).

По-сериозният проблем със съвместимостта е този, че разпространените програми за начално зареждане на Linux (и Lilo, и GRUB, макар и по различни причини) често не могат да заредят ядрото на Linux от дял с ReiserFS файлова система, оптимизирана с режим на опаковане на опашките на файловете. А поради това, че след като бъде изключен, този режим притежава свойството да се самовъзстановява, потребителя може да се сблъска с това, че след прекомпилация на ядрото системата просто да откаже да зареди. Именно заради това споменах по-горе, че може да се окаже необходимо създаването на отделен дял за директорията /boot.

**Ext3fs**

За разлика от ReiserFS, ext3fs не е нищо повече от протоколираща надстройка на класическата ext2fs, разработена от Стивън Туиди в компанията Red Hat и поддържана от ядрото на Linux от версия 2.4.16. Като резултат от този произход, тя запазва пълна съвместимост със своя прародител, включително и на ниво обслужващи програми (започвайки от версия 1.21 на обединяващия ги пакет e2fsprogs). И прехода от ext2fs към ext3fs може да бъде осъществен с просто добавяне на журналния файл към първия дял, не само без преформатиране на дяла, но даже и без рестартиране на машината.

От тук идва първото предимство на ext3fs, особено при голям брой компютри. Второто предимство е едва ли не максималната надеждност на системата – ext3fs е единствената файлова система от разгледаните, в която е възможно протоколиране на операциите не само с метаданните на файловете, но и със самите данни.

В ext3fs са предвидени три режима на работа – пълно протоколиране (full data journaling), протоколиране с обратен запис (writeback), а също и активираното по подразбиране последователно протоколиране (ordered).

Режима на пълно протоколиране, както лесно може да се досетим от името му, се разпростира и върху метаданните, и върху данните на файловете. Всичките техни изменения отначало се записват в журналния файл и чак след това се фиксират на диска. В случай на аварийно спиране на работата, може повторно да прочетем журнала и да приведем данните и метаданните в непротиворечиво състояние. Този механизъм практически премахва загубите на данни, но е най-бавният от трите режима на работа.

В режима на отложен запис, обратно – в журналният файл се записват само измененията на метаданните на файловете, подобно на всички разгледни по-надолу файлови системи. Тоест той не дава никаква гаранция за запазване на данните, но осигурява най-голямото (в рамките на ext3fs) бързодействие.

При последователният режим на работа, физически се протоколират само метаданните на файловете, но свързаните с тях блокове данни логически се групират в общ модул, наречен транзакция (transaction). И тези блокове се записват преди записа върху диска на нови метаданни, което макар и да не гарантира запазване на данните, твърде много спомага за това. При това с по-малки допълнителни разходи в сравнение с пълното протоколиране, осигурявайки някакво междинно ниво на бързодействие.

Но по мои наблюдения, ext3fs нито в един от трите режима не може да се похвали с висока производителност при изпълнение на потребителските задачи. А в режим на пълно протоколиране са възможни ситуации, водещи за известно време до пълно парализиране на машината.

**XFS**

Файловата система XFS, за разлика от по-младите ReiserFS и ext3fs се развива от фирмата SGI в продължение на почти 10 години – за първи път тя се появи във версията 5.3 на операционната система Iris, която излезе през 1994 г. Но тя беше портната за Linux сравнително неотдавна (нейната текуща версия[**[4]**](http://www.linux-bg.org/cgi-bin/y/index.pl?page=article&id=advices&key=365258709#sdfootnote4sym) е 1.1, свободно достъпна на страницата за XFS от сайта на SGI – [**http://oss.sgi.com/projects/xfs**](http://oss.sgi.com/projects/xfs)) и досега не се поддържаше от официалните ядра[**[5]**](http://www.linux-bg.org/cgi-bin/y/index.pl?page=article&id=advices&key=365258709#sdfootnote5sym).

XFS е единствената 64 разрядна файлова система от разгледаните. Но нейната уникалност не е само в това. Особеностите на файловата система XFS са:

* използване на механизма allocation group – разделяне на общият дисков дял на няколко равни области, имащи собствени списъци на информационните възли и свободни блокове, за едновременно изпълнение на дисковите операции;
* логическо протоколиране само на променените метаданни, но с чест запис върху диска за минимизиране на възможните загуби при евентуални сривове;
* използване на механизма delayed allocation – присъединяване на дисковото пространство при запис на файла не по време на протоколирането, а при фактическото му прехвърляне върху диска, което заедно с повишаване на производителността, предотвратява и фрагментацията на дисковият дял;
* наличие на списъци за контрол на достъпа (ACL – Access Control List) и разширени атрибути на файловете (extended attributes), разглеждането на които надхвърля рамките на тази статия.

Като резултат от всичко това, XFS представлява много балансирана файлова система, която е надеждна почти колкото ext3fs и не отстъпва с много на ReiserFS в бързодействието на повечето файлови операции. А при операции с (много) големи файлове, тя е просто без конкуренция – както може да се досетите от името на компанията, която я е създала, тя е предназначена за работа с мултимедийни приложения с техните огромни потоци от данни. Не са отбелязани и проблеми със съвместимостта и с другите файлови системи.

Всичко казано досега, позволява да се направи извода, че XFS е оптималната файлова система за Linux, но трябва да се има предвид, че за разлика от ReiserFS и ext3fs, поддръжката на които е стандартно включена в ядрото на Linux, XFS и досега (текуща версия на ядрото 2.4.19) не се поддържа от каноничното ядро на Линус Торвалдс (това, което може да изтеглите от [**http://www.kernel.org**](http://www.kernel.org/)). Макар, че с неотдавнашното включване на такава поддръжка в клон 2.5.Х на ядрото, ни позволява да се надяваме, че скоро тази функция ще стане стандартна.

Възможност за работа с XFS осигурява специална кръпка (xfs-2.4.1X-all-i386.bz2), която може да се изтегли от сайта на [**SGI**](http://oss.sgi.com/projects/xfs), заедно със съответните приложни програми за нейната поддръжка – традиционните средства e2fsprogs не са приложими към XFS. Програмите за поддръжка на XFS са обединени в няколко пакета, от които абсолютно необходим е xfsprogs. За всичко това трябва да се помни при предварителното разделяне на диска.

**Критерии за избор**

По този начин, всяка от четирите разгледани файлови системи има своя уникална положителна особеност (даже и ext2fs – както и да го разглеждаме, тя остава лидер по общо бързодействие) и поне един недостатък (който не пречи на нейното използване). Така, че изборът на файлова система трябва да се определя от задачите на потребителя и характера на преобладаващите му данни.

Освен това, ако, както се разказва в статията за LVM, разтоварим максимално коренната файлова система, отделяйки в логически дялове отделните и клонове, е възможно и комбинирано използване на файловите системи. Тоест за всеки дисков дял избираме най-подходящата за неговото съдържание управляваща структура.

Както нееднократно се споменаваше, ext2fs е най-подходящият избор за зареждащия дял (а при използване на GRUB като програма за начално зареждане, това е почти задължително). Освен това, ext2fs е напълно подходяща за /tmp или /var клоновете от главната директория (/). За първата от тях, по дефиниция устойчивостта към сривове на системата не е критична. За втората, основното изискване е бързодействието (например в source based дистрибуции от типа на Gentoo /var се използва за съхранение на временните продукти от компилацията и бързодействието на файловите операции в нея донякъде ускорява компилацията на пакетите). Накрая, на настолна машина, ext2fs може да се приложи и за корена на файловата система (/) – нали при разбиването на диска на повече дялове, в корена остават най-рядко променяните компоненти на системата.

От друга страна, коренът на файловата система е най-критичният по отношение на устойчивостта елемент от файловата система. И затова, най-оптимална за него е файловата система ext3fs, като най-устойчива. Освен това, в екстремни ситуации тя може да бъде монтирана без проблеми като ext2fs. За дяловете от типа /usr и /usr/local ext3fs също изглежда напълно подходящ вариант.

Най-важната част на файловата система на една настолна машина от гледна точка на потребителя са неговите, потребителски данни, които обикновено са разположени в директорията /home (понеже системата може да се преинсталира, а загубата на данни може да се окаже невъзстановима). Но тези данни са най-често променяната част от файловата система, което поставя високи изисквания към бързодействието на файловите операции. И затова ext3fs не е най-доброто (imho[**[6]**](http://www.linux-bg.org/cgi-bin/y/index.pl?page=article&id=advices&key=365258709#sdfootnote6sym)) решение за директорията /home. По-целесъобразно е за нея да използваме някоя от бързодействащите журнални файлови системи, ReiserFS или XFS. Изборът между тях се определя от личните предпочитания и вида на данните (ще се възползвам от случая и ще отбележа, че от бързодействието на JFS, по мои наблюдения, при типични потребителски действия, има какво още да се желае).

Очевидно, бързодействието на XFS при работа с файлове с (много) голям размер я прави предпочитана, ако става дума за обработка на изображения, мултимедийно съдържание, картографска информация и т. н. В същото време, преимуществата на ReiserFS се проявяват основно при работа с файлове с (много) малка големина (по-малки от един блок на файловата система), каквито между потребителските данни обикновено са малко[**[7]**](http://www.linux-bg.org/cgi-bin/y/index.pl?page=article&id=advices&key=365258709#sdfootnote7sym). И затова моето лично мнение е еднозначно в полза на XFS. Освен това, собственият ми опит в общуването с ReiserFS беше неблагоприятен, особено в съчетание с технологията LVM, докато XFS според моите впечатления, идеално си хармонира с нея.

Вече е време да теглим чертата – оптимално за мен изглежда следното съчетание на файлови системи:

* ext3fs – за корена (/) на файловата система и директорията /usr (и за /usr/local и /usr/X11R6, ако те също ще се отделят в отделни дялове);
* ext2fs – за директорията за начално зареждане /boot и директориите /tmp и /var;
* XFS – за дяла с потребителските директории (/home)

И пак ще повторя, че това е само мое мнение, основано на опита от прилагането на Linux на настолни машини – за различните видове сървъри, то не е в сила.

**ПРАКТИЧЕСКИ ПОСЛЕДСТВИЯ**

Надявам се, че предишният раздел даде достатъчно информация за правилен избор на файлова система, най-подходяща за конкретните условия. И затова вече може да разгледаме накратко последователността от действия, след като този избор е направен.

Обикновено при инсталацията на Linux, създаването на файловите системи е от компетенцията на инсталатора, който го осъществява с няколко възможности по подразбиране. От контекста на разгледаното в предишния раздел става ясно, че тези опции не винаги са най-добрите от гледна точка на потребителя. Да се променят характеристиките, определени за файловата система при нейното създаване е невъзможно без повторно изпълнение на този процес (и съответно, загуба на данни). Но всички source based дистрибуции допускат ръчна намеса в процеса на създаване на файловата система, а например в Gentoo това е единственият начин за нейното изпълнение.

Файловата система ext2fs може да бъде създадена с всяка една от следните команди – /sbin/mke2fs, /sbin/mkfs, /sbin/mkfs.ext2 с указан файл на устройство в качеството на аргумент, например:

**$ /sbin/mke2fs /dev/hd?#**

Всяка от тези команди (a /sbin/mkfs.ext2 е символна връзка към /sbin/mke2fs) има редица възможности, като:

* + -b за определяне размера на блока на файловата система (възможни значения са 1024, 2048 или 4096 байта, по подразбиране е приета последната стойност);
  + -с за проверка на повредени участъци от диска;
  + -N и -i за задаване на броя на информационните възли и съответно количеството байтове на един възел.

Подробно може да се запознаете с възможностите на командите от съответните man-страници, например man (8) mke2fs.

За създаване на файловата система ext3fs може да се използва същата команда mke2fs с опция -j, като в този случай тя ще получи някои характеристики по подразбиране. Ръчното им определяне може да стане с помощта на командата:

**$ /sbin/mke2fs -J опция\_на\_протоколиране /dev/hd?#**

Възможните стойности на опцията за протоколиране са:

* + size=размер – задава размера на журналния файл в мегабайтове и
  + device=външен\_дневник – включва нова файлова система към дневника, предварително създадена на друг дисков дял.

Може да използваме и специалната команда /sbin/mkfs.ext3 – нейните възможности са идентични на възможностите на командата /sbin/mke2fs (и няма как да е другояче, тъй като тя е символна връзка към нея). Но най-интересното е възможността за преобразуване на съществуващата файлова система ext2fs към ext3fs не само без загуба на данни, но и без рестартиране на системата (даже и без демонтиране на файловата система) чрез просто добавяне на дневник към нея. Това става с командата:

**$ tune2fs -j /dev/hd?#**

Тя просто добавя файла на дневника /.journal в главната директория на модифицираната файлова система (ако последната не е била демонтирана) или стартира за него скрит информационен възел (ако преди модификацията файловата система е била демонтирана). Трябва да добавя, че обратното преобразуване е още по-лесно и се осъществява с командата за монтиране (което ще се обсъжда в следващата статия).

Файловата система ReiserFS се създава със специално предназначена за това команда – /sbin/mkreiserfs от пакета reiserfsprogs. Тази команда има много възможности (-s за създаване на големината на дневника, -f за принудително преформатиране на файлови системи от друг тип и т. н.). с които може да се запознаете с помощта на командата man (8) mkreiserfs. И за избягване на неочаквани проблеми, ще напомня, че ако коренният дял се форматира като ReiserFS, няма да е излишно да предвидим един малък дял за директорията /boot и да го форматираме с ext2fs.

За създаване на файловата система XFS също съществува собствена команда mkfs.xfs от пакета xfsprogs. В нея са предвидени няколко опции, всяка от които има няколко аргумента, приемащи числени значения. Най-важните от тях са:

* -b, която с помощта на аргумента size=## ни позволява да зададем размера на блока данни в байтове, който трябва да бъде кратен на размера на страницата оперативна памет (за платформата i386 – 4 kbytes) и може да варира в интервала от 512 до 65536 bytes (по подразбиране той е 4096 bytes);
* -d – определя параметрите на областта за данни на файловата система, такива като брой на самостоятелните области в дяла (Allocation groups, аргумент agcount) или техният размер (аргумент agsize);
* -l – определя параметрите на журналния файл, например неговата големина (size).

При използване на mkfs.xfs за постигане на максимална производителност, се препоръчва в явен вид да се зададе броя на allocation groups – в противен случай то се определя автоматично, което води до непроизводителен преразход на ресурси. Определянето на този брой се основава на допускането, че една allocation group заема 4 GB дисково пространство. След това може да установим и големината на дневника – тук препоръчваното значение е 32 MB, тоест за дисков дял с големина 20 GB командата ще има следния вид:

**$ mkfs.xfs -d agcount=5 -l size=32m /dev/hda1**

Освен всички изброени опции, командата mkfs.xfs има и опция -f за принудително създаване на файловата система XFS върху вече съществуваща файлова система. За нея е достатъчно тя да е била ext2fs (и воден от общи съображения и ext3fs, макар, че не съм го проверявал). Ако създаваме XFS върху ReiserFS, са възможни грешки при монтирането на новата файлова система. Това се отнася и за обратната процедура – при замяна на XFS с ReiserFS, а също и за случаите, когато заменяме някоя от “напредналите” файлови системи с ext2fs. Тези проблеми са свързани с факта, че командата за монтиране може да разпознае новосъздадената XFS файлова система като повредена ReiserFS и обратно.

За тяхното избягване, преди подобно заместване се налага да прибегнем към един малко шамански подход – да нулираме началните области на дяла (съхраняващ метаданните на файловата система) с командата:

**$ dd if=/dev/zero of=/dev/hd?#**

Не е необходимо да чакаме запълването с нули на цялото устройство – достатъчно е да оставим командата да работи 10-20 sec, след което да прекъснем нейното изпълнение с Control-D и да започнем създаването на новите файлови системи.

И последното, което трябва да се спомене – за swap дяла, създаден по време на разделянето на диска. Макар, че той не съдържа файлова система в себе си, той има нужда от такава, която се създава с командата

**$ mkswap име\_на\_устройство**

към която трябва да се подхожда с внимание – прилагането и върху обикновен дял ще унищожи всички данни върху него.

**БИБЛИОГРАФИЯ**

Много сложни въпроси около устройството на файловите системи в тази статия бяха засегнати само накратко. За по-подробна информация за тях трябва да се обърнете към допълнителни източници. Общата организация на файловата система на Unix се разглежда в много ръководства за нея, например в [**С. Д. Кузнецов, Операционная система UNIX**](http://www.citforum.ru/operating_systems/unix/contents.shtml).

Устройството на файловата система ext2fs е подробно описано в статията на Виктор Хименко “Файлове, файлове, файлове”, (Мир ПК, 2000, [**част 1 – No 2**](http://www.osp.ru/pcworld/2000/02/064.htm); [**част 2 – No 3**](http://www.osp.ru/pcworld/2000/03/050.htm),).

Подробно описание на съвременните журнални файлови системи, използвани в Linux е дадено в цикъла статии на Даниел Робинс, руският превод на които, направен от Владимир Холманов, е достъпен на сайта на [**Ярославската група на потребителите на Linux**](http://linux.yaroslavl.ru/Docum/Fs/l-fs_ru.html)[**[8]**](http://www.linux-bg.org/cgi-bin/y/index.pl?page=article&id=advices&key=365258709#sdfootnote8sym).

И така запознахме се с доста теория която ни показа какви видове файлови системи се използват в различните операционни системи и как да изберем подходящата за нашият случай .

Да разгледаме и в детайли командите които ще ни трябват преди да преминем към практическата им страна :

Du df parted fdisk mkfs tune2fs, dumpe2fs, fsck

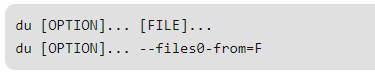
Да се запознаем с командите и коя какво може преди да ги обясниме на практика

Ще започнем по редът по който сме ги изброили

Командата du, съкратено за използването на диска, се използва за оценка на използването на файловото пространство.

Командата du може да се използва за проследяване на файлове и директории, които консумират прекомерно много място на твърдия диск.

Синтаксисът е както следва



И съответно опциите

-0, –null: приключва всяка изходна линия с NULL

-a, –всички: брой записи за всички файлове, а не само за директории

-Намаляващ размер: отпечатва се видими размери, а не използване на диск.

-B, -block-size = SIZE: мащабирайте размерите до SIZE преди печат на конзолата

-c, - общо: произвежда общия сбор

-d, –max-depth = N: отпечатва общо за директорията само ако е N или по-малко нива под аргумента на командния ред

-h, - четене от човека: размери за печат в четлив за човека формат

-S, -разделни-директории: за директории, не включват размер на поддиректории

-s, –summarize: показва само общо за всяка директория

–Time: показва времето на последната модификация на всеки файл или директория.

-Exclude = PATTERN: изключва файлове, които съответстват на PATTERN

След малко ще видим и как се използва .

Следващата команда е df

Възможно е да възникне ситуация, когато използвате Линукс, когато искате да знаете количеството пространство, изразходвано от определена файлова система във вашата система LINUX, или колко място е налично за дадена файлова система. LINUX се командва приятелски предоставя помощна програма за командния ред за това, т.е. df команда, която показва размера на дисковото пространство на разположение на файловата система, съдържаща всеки аргумент име на файл.

Ако не е подадено име на файл като аргумент с командата df, то показва наличното пространство за всички текущо монтирани файлови системи

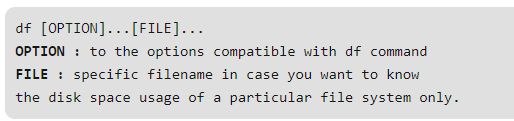
, Това е нещо, което бихте искали да знаете, защото командата df не е в състояние да покаже наличното пространство на немонтирани файлови системи и причината за това е, че за това в някои системи се изисква много задълбочено познаване на структурите на файловата система.

По подразбиране df показва дисковото пространство в 1 K блока.

df показва стойностите в единиците от първия наличен РАЗМЕР от -block-size (което е опция) и от променливите на среда DF\_BLOCK\_SIZE, BLOCKSIZE и BLOCK\_SIZE.

По подразбиране единиците са настроени на 1024 байта или 512 байта (ако е зададен POSIXLY\_CORRECT). Тук SIZE е цяло число и незадължителната единица и единици са K, M, G, T, P, E, Z, Y (като K в килобайт).

Синтаксиса е



Най често се ползва df –h

Следващата команда е mkfs

В Unix-подобни операционни системи командата mkfs създава и изгражда файлова система.

описание

mkfs се използва за изграждане на файлова система на Linux на устройство, обикновено дял на твърдия диск. Аргументът на устройството е или името на устройството (например, / dev / hda1, / dev / sdb2), или обикновен файл, който ще съдържа файловата система. Аргументът size е броят на блоковете, които ще се използват за файловата система.

Статусът на излизане, върнат от mkfs, е 0 при успех и 1 при неуспех.

В действителност, mkfs е интерфейс за различните съставители на файлови системи (mkfs.fstype), налични под Linux. Специфичният за файловата система строител се търси в редица директории, като например / sbin, / sbin / fs, /sbin/fs.d, / etc / fs, / etc (точният списък се определя при време на компилация, но поне съдържа / sbin и / sbin / fs), и накрая в директориите, изброени в променливата PATH. За повече подробности, моля, вижте ръководството за ръководителя на конкретната файлова система.



Възможните опции са :

-t, - тип тип

Посочете типа на файловата система, която ще бъде изградена. Ако не е посочен, се използва типът на файловата система по подразбиране (в момента ext2).

FS-опции

Опциите, специфични за файловата система, се предават на създателя на реалната файлова система. Въпреки че не са гарантирани, следните опции се поддържат от повечето създатели на файлови системи.

**-V, --verbose**

Изгответе подробен изход, включително всички специфични за файловата система команди, които се изпълняват. Задаването на тази опция повече от веднъж възпрепятства изпълнението на всякакви специфични за файловата система команди. Това наистина е полезно само за тестване.

-V, - версия

Показване на информация за версията и изход. Option -V ще показва информация за версията само когато е единственият параметър, в противен случай ще работи като --verbose.

-h, -help

Показване на помощ и изход.



Следващата команда е tune2fs

настройте параметрите на файловата система, които могат да се настройват, на файловите системи ext2 / ext3 / ext4

и съответно опциите

Настроики

-c max-mount-counts

Настройте броя на монтиране, след което файловата система ще бъде проверена от e2fsck (8). Ако max-mount-counts е 0 или -1, броят на монтиране на файловата система ще бъде игнориран от e2fsck (8) и ядрото.

Зашеметяването на броя на планините, при които файловите системи са принудително проверени, ще предотврати проверка на всички файлови системи едновременно, когато се използват файлови системи с журнали.

Трябва силно да обмислите последиците от пълното изключване на проверката, зависима от планирането. Лошите дискове, кабели, памет и грешки в ядрото могат да повредят една файлова система, без да маркират файловата система мръсна или погрешно. Ако използвате журналиране във вашата файлова система, вашата файлова система никога няма да бъде маркирана като мръсна, така че обикновено не се проверява. Грешката на файловата система, открита от ядрото, ще принуди fsck при следващото рестартиране, но може вече да е твърде късно да се предотврати загубата на данни в този момент.

Вижте също опцията -i за проверка в зависимост от времето.

-C планиране

Задайте колко пъти е монтирана файловата система. Ако е зададена по-голяма стойност от параметъра max-mount-counts, зададен от опцията -c, e2fsck (8) ще провери файловата система при следващото рестартиране.

-e поведение на грешка

Промяна на поведението на кода на ядрото при откриване на грешки. Във всички случаи грешката на файловата система ще предизвика e2fsck (8) да провери файловата система при следващото зареждане. поведението за грешка може да бъде едно от следните:

продължи

Продължи нормалното изпълнение.

отнасям се към-ро

Премествайте файловата система само за четене.

паника

Причинява паника на ядрото.

-E разширени опции

Задайте разширени опции за файловата система. Разширените опции са разделени със запетая и могат да вземат аргумент с помощта на знак equals ('='). Поддържат се следните разширени опции:

крачка = крачка размер

Конфигурирайте файловата система за RAID масив с блокове на файловата система с размер на стъпките. Това е броят на блоковете, които се четат или записват на диска, преди да се премине към следващия диск. Това най-вече засяга поставянето на метаданни на файловата система като растерни изображения в mke2fs (2) време, за да се избегне поставянето им на един диск, което може да навреди на производителността. Може да се използва и от блок разпределител.

stripe\_width = ивица ширина

Конфигурирайте файловата система за RAID масив с блокове на файловата система с ширина на лента за лента. Това обикновено е размер на стъпката \* N, където N е броят на дисковете с данни в RAID (например RAID 5 N + 1, RAID 6 N + 2). Това позволява на блок разпределителя да предотврати четене-модифициране-запис на паритета в RAID лента, ако е възможно, когато данните са написани.

hash\_alg = хеш-ALG

Задайте алгоритъма по подразбиране за хеш, използван за файлови системи с хеширани b-tree директории. Приети са валидни алгоритми: наследство, half\_md4 и чай.

mount\_opts = mount\_option\_string

Задайте набор от опции за монтиране по подразбиране, които ще се използват, когато файловата система е монтирана. За разлика от опциите за монтиране по подразбиране, базирани на битова маска, които могат да бъдат зададени с опцията -o, mount\_option\_string е произволен низ с максимална дължина от 63 байта, който се съхранява в суперблока.

Първо се прилага драйверът на файловата система ext4

опциите по подразбиране на базата на битова маска и след това анализират mount\_option\_string, преди да анализират опциите за монтиране, преминали от програмата mount (8).

Тази настройка на суперблок се признава само в ядрата на 2.6.35+;

и въобще не от драйверите на файловата система ext2 и ext3.

test\_fs

Задайте флаг в суперблоката на файловата система, показващ, че той може да бъде монтиран с помощта на експериментален код на ядрото, например файловата система ext4dev.

^ test\_fs

Изчистете флага test\_fs, като посочите, че файловата система трябва да бъде монтирана само с помощта на код на файловата система на производствено ниво.

-f

Принуди операцията tune2fs да се завърши дори и в лицето на грешки. Тази опция е полезна, когато премахвате функцията на файловата система has\_journal от файлова система, която има външен журнал (или е повредена, така че изглежда, че има външен журнал), но външният журнал не е достъпен.

ПРЕДУПРЕЖДЕНИЕ: Премахването на външен журнал от файлова система, която не е била чиста, без да се преиграва външният журнал, може да доведе до тежка загуба на данни и повреда на файловата система.

-g група

Задайте групата, която може да използва резервираните блокове на файловата система. Параметърът на групата може да бъде цифров гид или име на група. Ако е дадено име на група, тя се преобразува в цифров гид, преди да се съхрани в суперблока.

-i "интервал между проверките [d | m | w]"

Настройте максималното време между две проверки на файловата система. Никакви суфикс или d няма да интерпретират интервалите между проверките като дни, m като месеци и w като седмици. Стойност нула ще деактивира зависещата от времето проверка.

Силно се препоръчва да се активира или -c (зависеща от mount-count) или -i (зависеща от времето) проверка, за да накара периодичната пълна проверка на e2fsck (8) на файловата система. Ако не го направите, това може да доведе до повреда във файловата система (поради лоши дискове, кабели, памет или грешки в ядрото), което в крайна сметка ще доведе до загуба на данни или корупция.

-j

Добавете ext3 журнал към файловата система. Ако опцията -J не е зададена, параметрите на журнала по подразбиране ще се използват за създаване на журнал с подходящ размер (предвид размера на файловете

Следващата програма е dumpe2fs

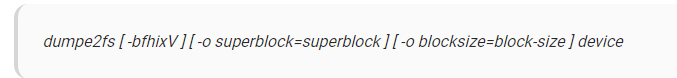
Командата dumpe2fs се използва за отпечатване на суперблока и блокира групова информация за файловата система, която присъства на устройството.

Може да се използва с ext2 / ext3 / ext4 файлова система за информация.

Отпечатаната информация може да е стара или несъвместима, когато се използва с монтирана файлова система.

Не забравяйте да демонтирате дяла си, преди да използвате тази команда.

Устройство



Настроики:

b: Използва се за получаване на блокове, които са запазени толкова лошо в файловата система.

o superblock = superblock: Използва суперблок по време на прегледа на файловата система. Обикновено тази опция не е необходима, освен от помощника на файловата система, който разглежда останките на много зле покварена файлова система.

o blocksize = blocksize: Използва блоковете по време на прегледа на файловата система. Обикновено тази опция не е необходима, освен от помощника на файловата система, който разглежда останките на много зле покварена файлова система.

f: Използва се, за да принуди dumpe2fs да покаже файлова система, въпреки че може да има някои флагове за функциите на файловата система, които dumpe2fs може да не разберат.

h: Тя ще показва само информацията за суперблока, а не детайлната информация на дескриптора на блоковата група.

i: Използва се за показване на данните от файловата система от файл с изображение, създаден от e2image, като се използва устройството като път към файла с изображението.

x: Показване на блоковете с подробна информация за групата в шестнадесетичен формат.

V: Показване на номера на версията на dumpe2fs и изход.

Superblock: Това е запис на характеристиките на файловата система. Тя включва информация за размера, размера на блока, празните и запълнените блокове. Linux поддържа и копие на своя суперблок <в паметта. Ако не можете да монтирате устройството си, това може да се дължи на повредена суперблок.

Следващата команда е fsck

В Unix-подобни операционни системи командата fsck проверява и ремонтира файлова система на Linux.

описание

fsck означава "проверка на последователността на файловата система". В повечето системи fsck се стартира по време на зареждане, ако се открият определени условия. Обикновено тези условия са:

Файловата система е маркирана като "мръсна" - нейното писмено състояние е несъвместимо с данните, които е планирано да бъде написано; или,

Файловата система е монтирана определен брой пъти, без да бъде проверена.

Самата команда fsck взаимодейства със съответстваща на файловата система команда fsck, създадена от авторите на файловата система. Независимо от типа на файловата система, fsck има три режима на работа:

Проверете за грешки и помолете потребителя да интерактивно да реши как да разрешат индивидуалните проблеми;

Проверка за грешки и опит за автоматично отстраняване на грешки; или,

Проверете за грешки и не се опитвайте да ги поправите, но показвайте грешките на стандартния изход.

Докато обикновено се стартира по време на зареждане, fsck може да се стартира ръчно от демонтирани файлови системи от суперпотребител.

Техническо описание

fsck се използва за проверка и по избор ремонт на една или повече файлови системи.

filesys може да бъде име на устройство (например, / dev / hdc1, / dev / sdb2), точка на монтиране (например, /, / usr, / home), или етикет ext2 или спецификатор UUID (например "UUID = 8868abf6- 88c5-4a83-98b8-bfc24057f7bd "или" LABEL = root ").

Обикновено fsck ще се опитва да управлява паралелно файловите системи на различни физически дискове, за да намали общото време, необходимо за проверка на всички тях.

Ако в командния ред не са посочени файлови системи и опцията -A не е зададена, fsck ще проверява по подразбиране файловите системи в / etc / fstab серийно. Това е еквивалентно на комбинацията от опциите -A и -s.

Кодът на изход, върнат от fsck, е уникален номер, представляващ сумата от следните стойности на условията:

0 - Няма грешки

1 - Коригирани са грешки в файловата система

2 - Системата трябва да се рестартира

4 - Грешките на файловата система остават некоригирани

8 - Оперативна грешка

16 - Използване или синтактична грешка

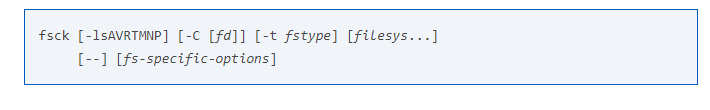
32 - Fsck анулирано от потребителска заявка

128 - Грешка в споделена библиотека

Изходният код, който се връща, когато се проверяват множество файлови системи, е битовото OR от кодовете за изход за всяка файлова система, която се проверява.

В действителност, fsck е интерфейс за различните контролери за файлови системи (fsck.fstype), налични под Linux. Проверката за файловата система се търси първо в / sbin, след това в / etc / fs и / etc, и накрая в директориите, изброени в променливата на средата PATH.

Прочетете ръководството за проверка на файловата система за повече подробности. Например, за да научите повече за конкретната fsck проверка в ext3, изпълнете:



Опциите са :

-l

Заключете цялото дисково устройство от изключително стадо. Тази опция може да се използва само с едно устройство (това означава, че -A и -l са взаимно изключващи се). Тази опция се препоръчва, когато се изпълняват повече fsck копия по едно и също време. Опцията се игнорира, когато се използва за няколко устройства или за дискове, които не се въртят. fsck не заключва подлежащите устройства, когато се изпълнява, за да провери натрупаните устройства (напр. MD или DM); тази функция все още не е приложена.

-с

Serialize fsck операции. Това е добра идея, ако проверявате няколко файлови системи и чешмите са в интерактивен режим. Забележка: e2fsck се изпълнява по подразбиране в интерактивен режим. За да стартирате e2fsck в неинтерактивен режим, трябва или да укажете опцията -p или -a, ако искате грешките да бъдат коригирани автоматично, или опцията -n, ако не.

-t fslist

Задава типа (овете) файлова система, която трябва да бъде проверена. Когато е зададен флаг -A, се проверяват само файлови системи, които съвпадат с fslist. Параметърът fslist е разделен със запетая списък на файловите системи и спецификаторите за опции. Всички файлови системи в този разделен със запетая списък могат да бъдат предшествани от оператор за отхвърляне "no" или "!", Който изисква да бъдат проверявани само тези файлови системи, които не са изброени в fslist. Ако нито една от файловите системи в fslist не е поставена с префикс от оператор за отхвърляне, тогава ще бъдат проверявани само изброените файлови системи.

Спецификаторите за опции могат да бъдат включени в разделения със запетаи fslist. Те трябва да имат формат опт = fs-option. Ако е налице спецификатор на опции, тогава ще бъдат проверени само файлови системи, които съдържат fs-option в полето за опции за монтиране на / etc / fstab. Ако спецификаторът на опциите е с представка от оператор за отхвърляне, тогава ще бъдат проверени само тези файлови системи, които нямат fs-опция в полето за опции за монтиране на / etc / fstab.

Например, ако opts = ro се появи в fslist, тогава ще бъдат проверени само файловите системи, изброени в / etc / fstab с опцията ro.

За съвместимост с дистрибуции на Mandrake, чиито скриптове за зареждане зависят от неразрешена промяна на потребителския интерфейс към програмата fsck, ако типът на файловата система на цикъла е намерен в fslist, той се третира като, ако opts = loop са зададени като аргумент към опцията -t.

Обикновено типът файлова система се извежда чрез търсене на filesys в файла / etc / fstab и използване на съответния запис. Ако типът не може да бъде изведен, и има само една файлова система, дадена като аргумент към опцията -t, fsck ще използва посочения тип файлова система. Ако този тип не е наличен, тогава се използва типът на файловата система по подразбиране (в момента ext2).

-А

Преминете през файла / etc / fstab и опитайте да проверите всички файлови системи в един цикъл. Тази опция обикновено се използва от файла за инициализация на системата / etc / rc, вместо от множество команди за проверка на една файлова система.

Коренната файлова система ще бъде проверена първо, освен ако не е посочена опцията -P (вижте по-долу). След това файловите системи ще бъдат проверени в реда, определен от полето fs\_passno (шестото) в файла / etc / fstab. Файловите системи с fs\_passno стойност от 0 се пропускат и въобще не се проверяват. Файловите системи с стойност fs\_passno по-голяма от нула ще бъдат проверени по ред, като файловите системи с най-нисък номер на fs\_passno се проверяват първо. Ако има множество файлови системи със същия номер на преминаване, fsck ще се опита да ги провери паралелно, въпреки че ще избегне провеждането на няколко проверки на файлова система на същия физически диск.

fsck не проверява натрупани устройства (RAID, dm-crypt, ...) успоредно с всяко друго устройство. Вижте настройката FSCK\_FORCE\_ALL\_PARALLEL по-долу. Файловата система / sys се използва за определяне на зависимости между устройства.

Следователно, много често срещаната конфигурация в / etc / fstab файловете е да настроите кореновата файлова система да има fs\_passno стойност от 1 и да настроите всички други файлови системи да имат fs\_passno стойност от 2. Това ще позволи на fsck да изпълнява паралелно файловите системи паралелно ако е благоприятно да се направи това. Системните администратори може да изберат да не използват тази конфигурация, ако се налага да избягват няколко проверки на файловата система, които се изпълняват паралелно по някаква причина - например, ако въпросната машина е в недостиг на памет, така че прекомерното пейджинг е проблем.

fsck обикновено не проверява дали устройството действително съществува, преди да извика конкретна проверка на файловата система. Следователно несъществуващи устройства могат да накарат системата да влезе в режим на поправка на файловата система по време на зареждане, ако конкретната проверка на файловата система върне фатална грешка. Опцията / etc / fstab mount nofail може да се използва, за да пропусне fsck несъществуващи устройства. fsck също пропуска несъществуващи устройства, които имат специален тип файлова система auto.

-C [fd]

Покажете лентите за завършване / напредък за тези контролери за файлови системи (в момента само за ext2 и ext3), които ги поддържат. Fsck ще управлява контролерите на файловата система, така че само един от тях ще покаже лента на прогреса в даден момент. Предната част на GUI може да определи дескриптор на файла fd, в който случай информацията за лентата на прогрес ще бъде изпратена към дескриптора на файла.

-М

Не проверявайте монтираните файлови системи и връщайте код за изход от 0 за mo

Специфични за файловата система опции

Опциите, които не се разбират от fsck, се предават на проверяващата програма за файловата система. Тези аргументи не трябва да вземат аргументи, тъй като няма начин fsck да може правилно да предположи кои опции вземат аргументи и кои не.

Опциите и аргументите, които следват - се третират като опции, специфични за файловата система, които трябва да се предадат на контролера, специфичен за файловата система.

Моля, обърнете внимание, че fsck не е проектиран да предава произволно сложни опции за конкретни за файловата система пулове. Ако правите нещо сложно, моля, изпълнете директно контролера, специфичен за файловата система. Ако преминете fsck някои ужасно сложни опции и аргументи, и това не прави това, което очаквате, почти сигурно правите нещо, което не трябва да правите с fsck. Опциите за различните файлови системи, специфични за fsck, не са стандартизирани. Ако имате съмнения, моля, консултирайте се с man страниците на контролера, специфичен за файловата система. Въпреки че не са гарантирани, повечето опции за файлова система поддържат следните опции:

-а

Автоматично поправя файловата система без никакви въпроси (използвайте тази опция с повишено внимание). Имайте предвид, че e2fsck поддържа -a само за обратна съвместимост. Тази опция е съпоставена с опцията -p на e2fsck, която е безопасна за използване, за разлика от опцията -a, която някои от файловите системи поддържат.

-н

За някои контролери, специфични за файловата система, опцията -n ще доведе до fsck, специфичен за fs, за да избегне опитите за отстраняване на проблеми, но докладва за такива проблеми на stdout. Това обаче не е вярно за всички контролери, специфични за файловата система. По-специално fsck.reiserfs няма да съобщава за корупция, ако се даде тази опция. fsck.minix изобщо не поддържа опцията -n.

-r

Интерактивно поправете файловата система (поискайте потвърждения). Забележка: Като цяло е лоша идея да използвате тази опция, ако няколко fsck се изпълняват паралелно. Също така имайте предвид, че това е поведението по подразбиране на e2fsck; той поддържа тази опция само заради обратната съвместимост.

-y

За някои контролери, специфични за файловата система, опцията -y ще доведе до fsck, специфичен за fs, винаги да се опитва автоматично да коригира корупцията във всяка открита файлова система. Понякога експертът може да успее да управлява ръчно fsck. Обърнете внимание, че не всички контролери за файловата система използват тази опция. По-специално fsck.minix и fsck.cramfs не поддържат опцията -y.

Примери

Внимание! Не стартирайте fsck на монтирана файлова система! Ако стартирате fsck във файлова система, която в момента е монтирана, ще получите (в най-добрия случай) съобщение за грешка и (в най-лошия случай) ще повредите данните на вашата файлова система. Ако искате да проверите файловата си система с fsck, пуснете го в режим на един потребител и се уверете, че файловата система е демонтирана, преди да продължите.

Отпадане в режим на един потребител

Тези инструкции трябва да ви помогнат да приведете вашата Linux система в режим на един потребител и да демонтирате всички файлови системи, които искате да проверите с fsck.

Повече за командата после

Следващата команда е mount и съответно umount

mount - монтиране на файлова система

umount - демонтиране на файлова система

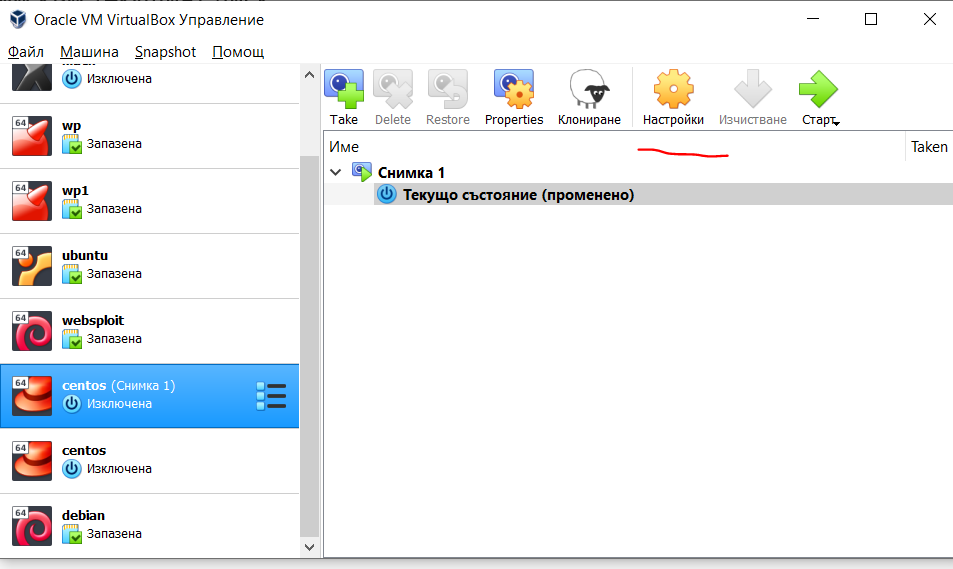
и съответно командите за форматиране на дискове и дялове

fdisk и parted

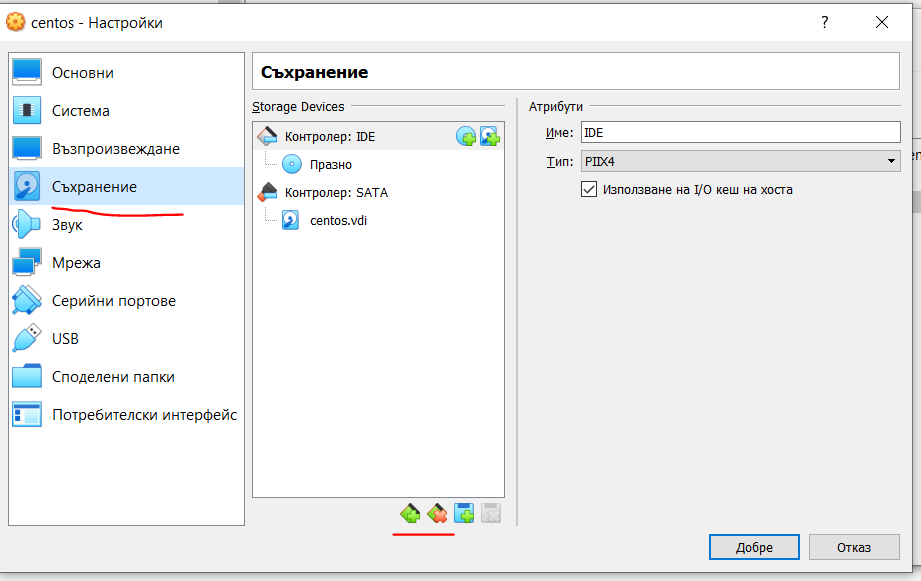
Ще разгледаме следният сценарий да кажем че на нашият сървър му свършва дисковото пространство и трябва да добавиме нов диск съответно да го форматираме и добавим към текущият диск.

За делта ще си направиме нов диск във virtualbox и ще го добавим към системата

Процедурата е следната от началната страница се избират настройки

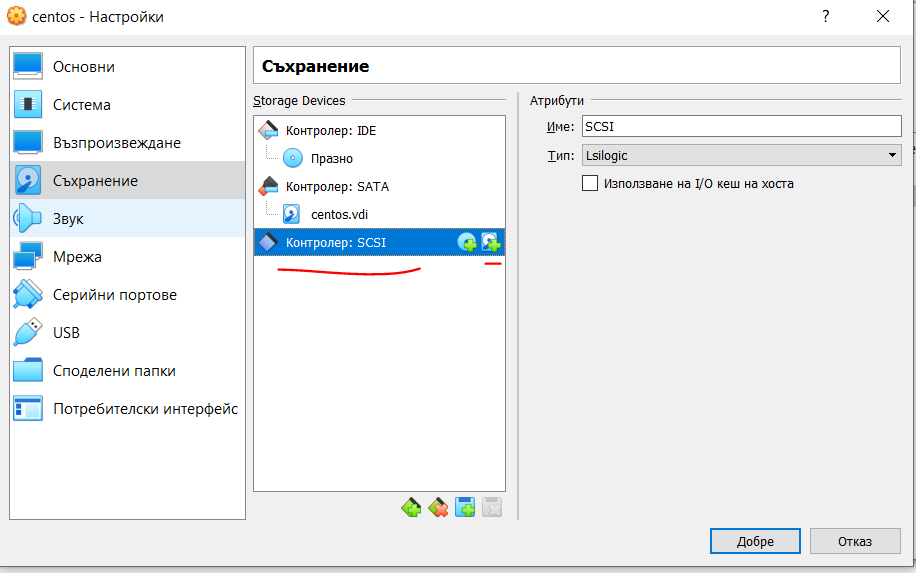


След това на съхранение и долу в дясно на плюса

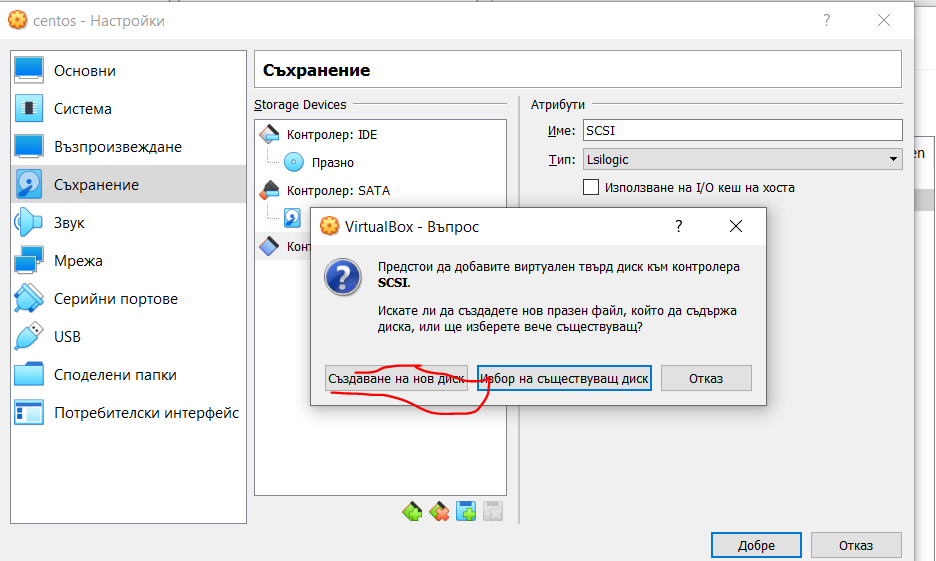


Добавяна на scsi контролер

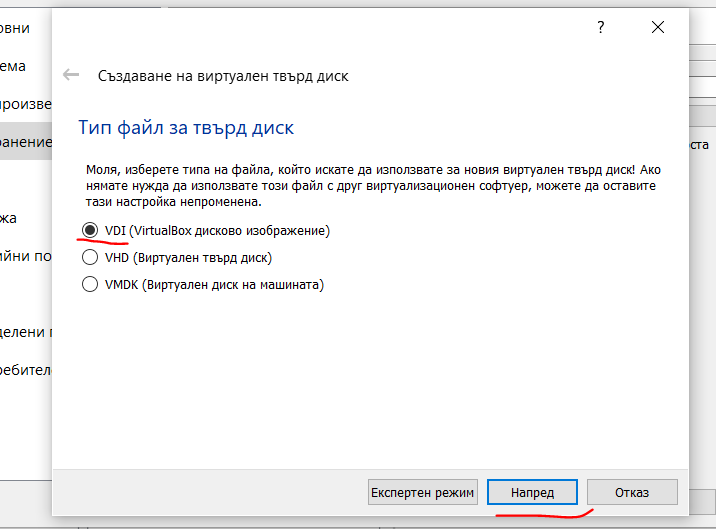
Така създадохме към машината устройство което все още не е диск за целта даваме на плюса срещу новото устройство където е нарисувано като хардиск



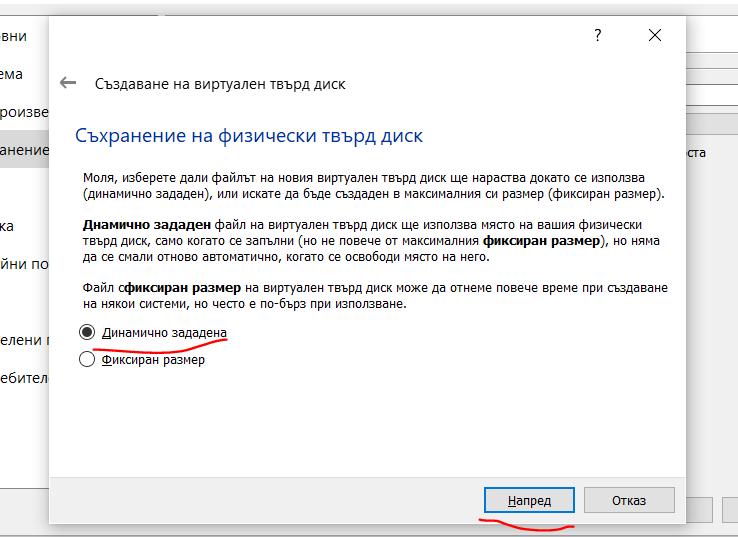
И на въпроса дали да се избере съществуващ хардиск или да се създаде нов избираме създай нов



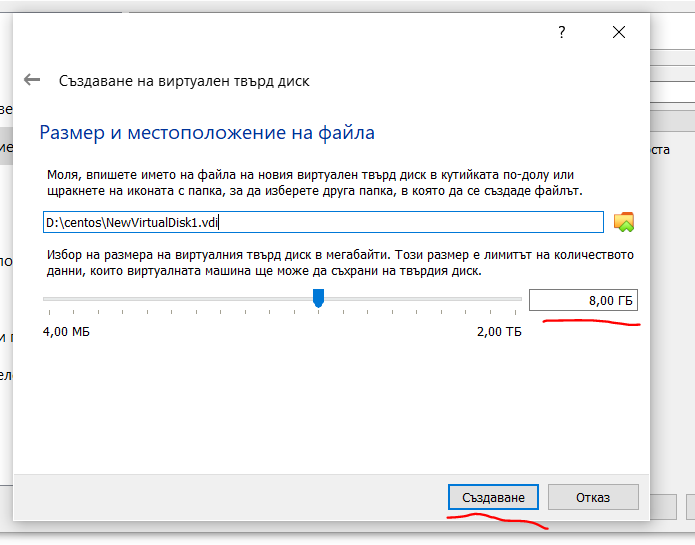
Оставяме настройките както са и даваме напред



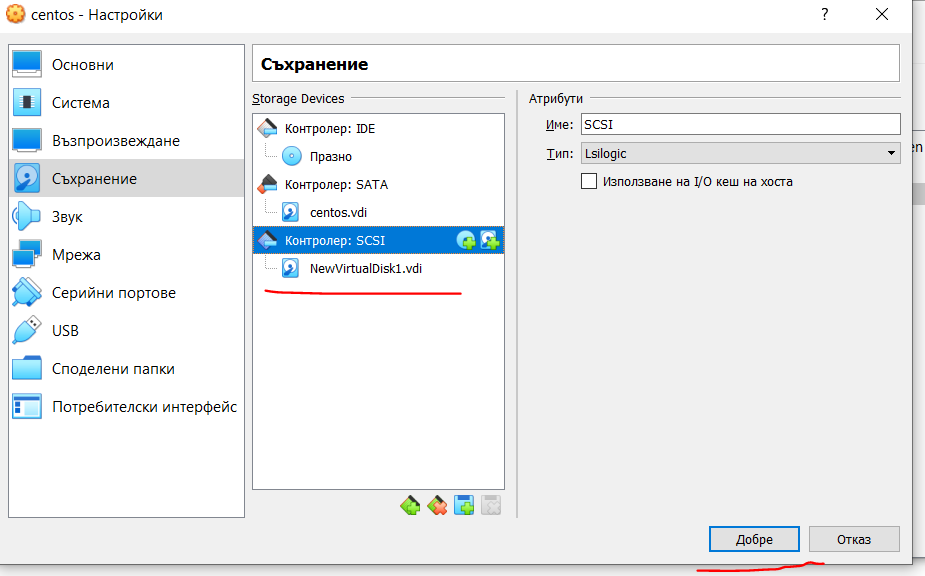
На следващият екран пак даваме напред



След това задаваме размерът на дискът ( в случаят ще го оставим колкото е 8Гб)



След създаваме прозорецът ще изчезне и вече имаме създаден нов диск



Даваме добре и стартираме нашата операционна система .

До тук създадохме нов хардиск но как да го монтираме на нашият сървър къде се намира и защо не се вижда след стартирането на машината .

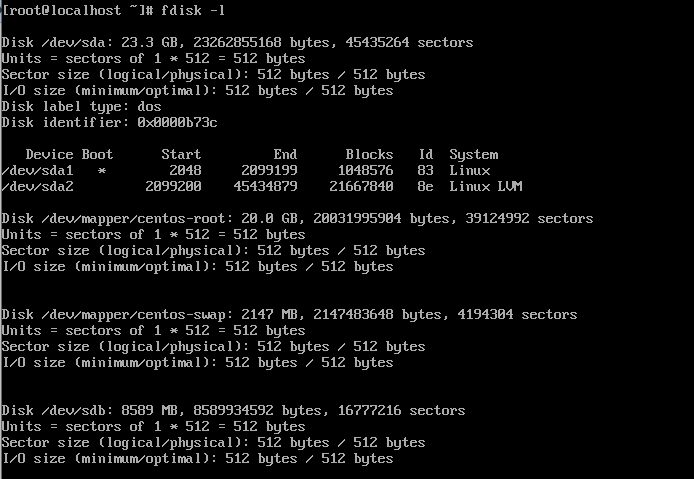
Първо да видиме дали нашият диск се вижда от машината най лесният начин е с командата fdisk или parted

Накратко тези 2 команди реално вършат едно и също а именнно форматират дискове но за това след малко.

Та да се върнем да си намерим диска за целта ще използваме командата fdisk със следните параметри ( ше ги обясниме след малко )

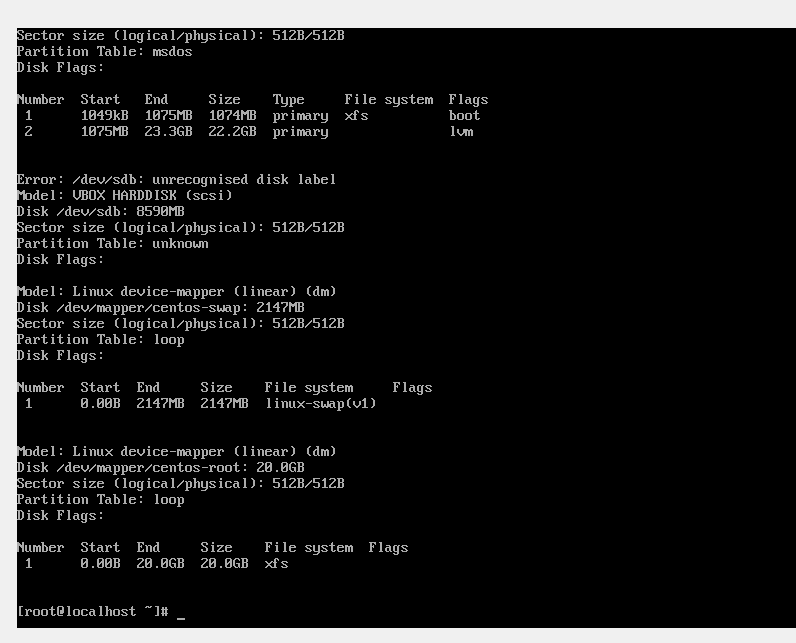
Fdisk –l

И виждаме следният резултат



Съответно с parted командата

Parted –l



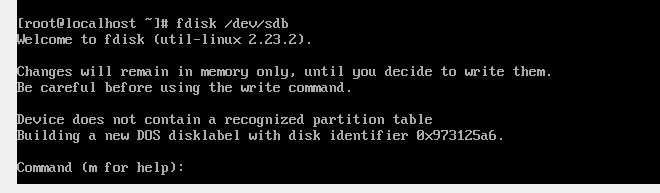
Както се вижда имаме нов диск /dev/sdb с обем 8 гигабайта

Следващата стъпка е да го форматираме в подходящата файлова система ( както виждаме нашата система използва xfs файлова система ( реално е ext4)).

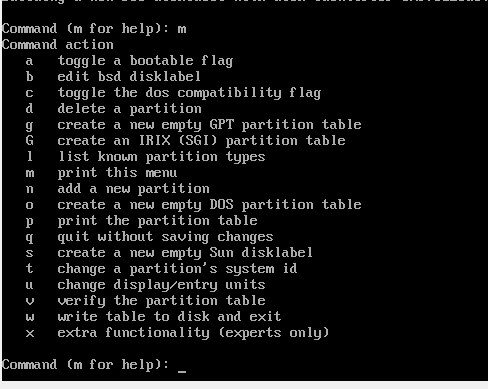
Е следва да го форматираме нашият диск

За целта ще използваме fdisk

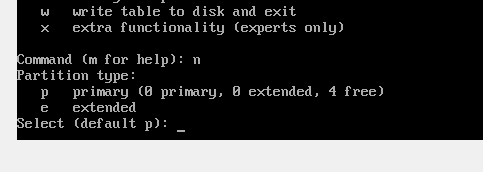
Командата ще ни изглежда по следният начин



Тъй като все още не знам как се работи с инструмента ще изпомзваме m за помощ



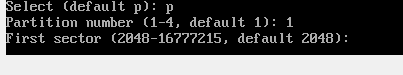
Опцията която търсим е n или добавяне на нов партишън



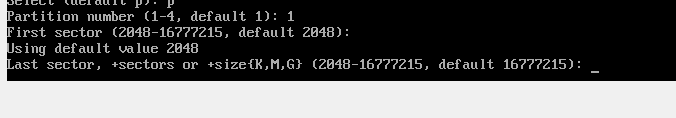
След това по подразбиране ни подканва да дадем p (primary)



Номер на партишъна по подразбиране 1



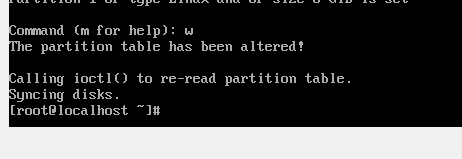
Размер на секторите сам одаваме ентър



Тук също само ентър

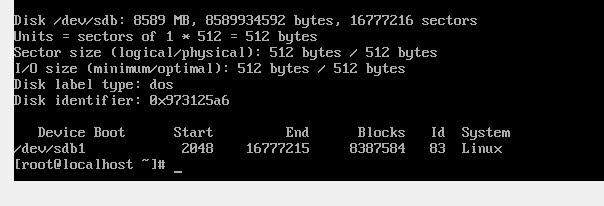


Следва да запишем промените на нашият диск с w



Проверяваме дали сме работили вярно с командата fdisk –l

И виждаме че нашият диск е вече форматиран в линукс формат



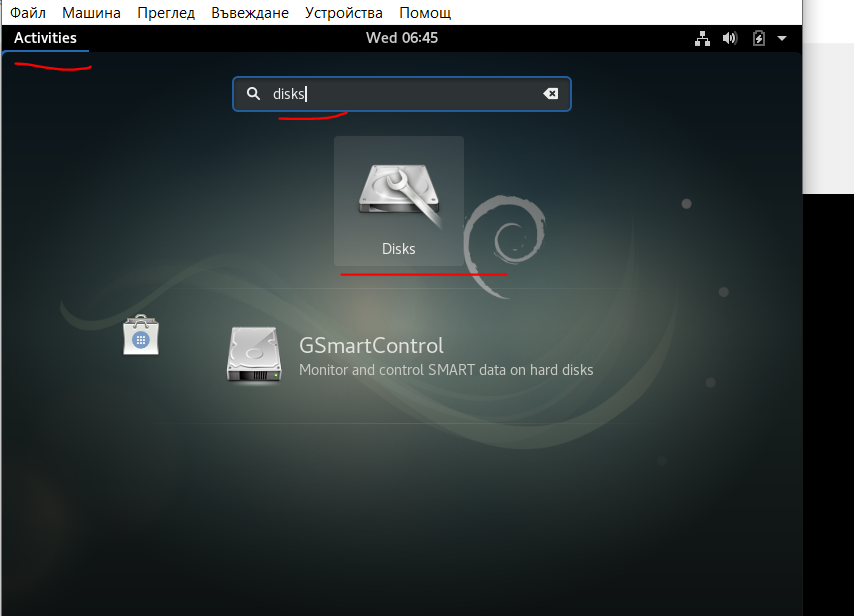
До тук си създадохме нов хардиск но все още операционната система не може да го ползва .

За разлика от fdisk parted има и графичен интерфейс

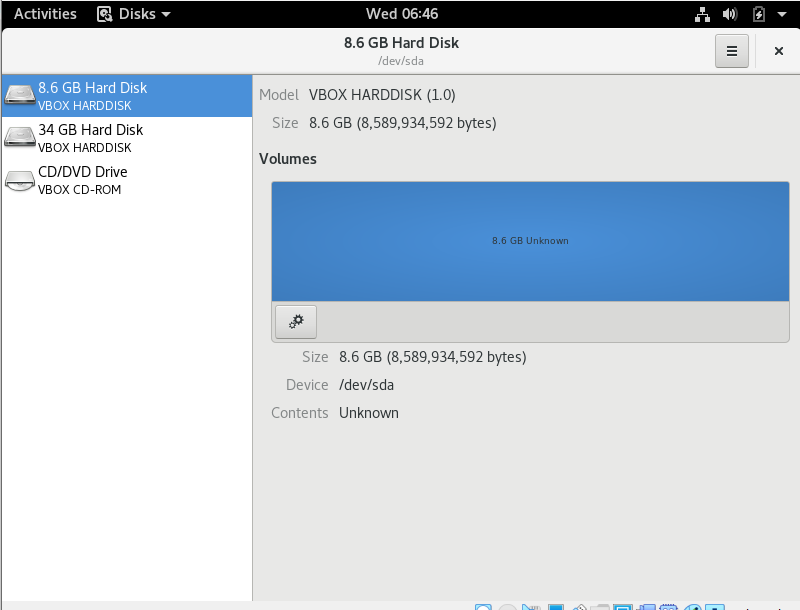
За целта ще направиме и на нашият дебиан дсървър по подобие на centos нов диск но ще ползваме графичната среда на parted

Процедурата п осъздаване на нов диск е същата така че ще прескочим тази част ( вижте по нагоре как става добавянето ) и направо ще преминем към стартирането на нашата система дебиан

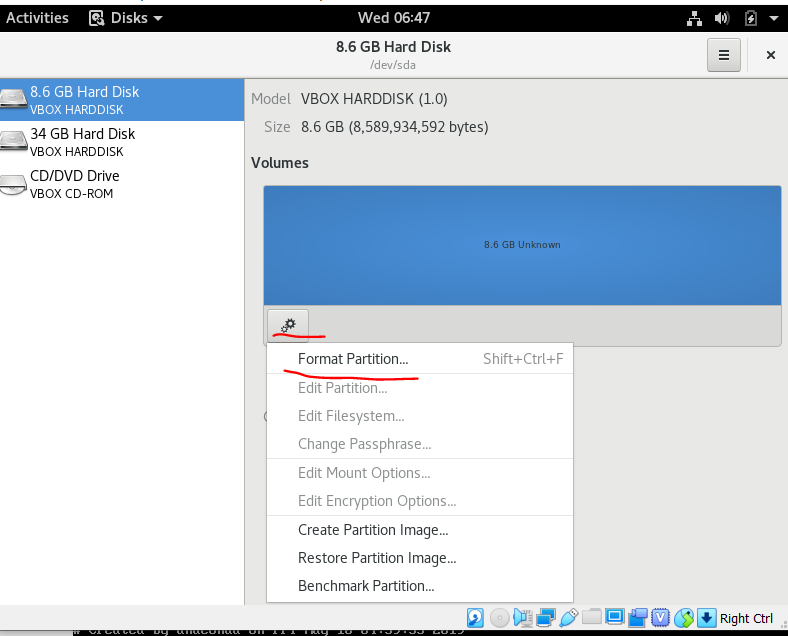
В activities полето където пише search пишем disks и избираме иконката

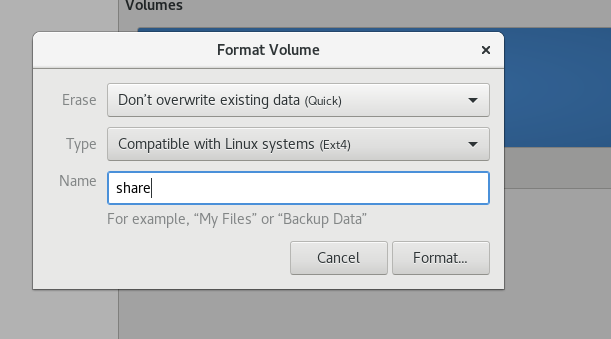


Появи ни се прозорецът на програмата



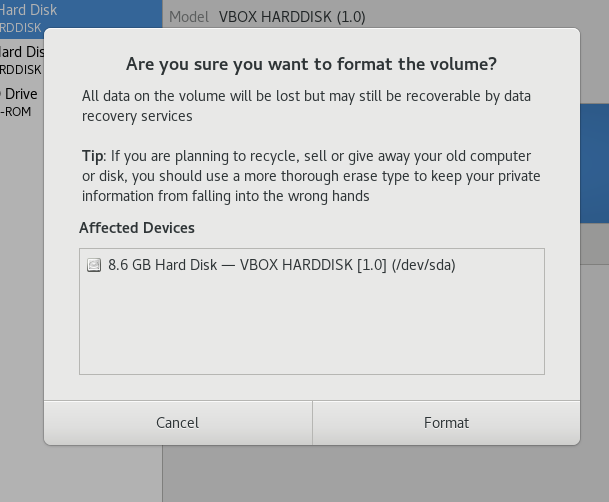
Както виждаме на първо място е нашият новосъздаден диск и няма нищо на него както и не може да се ползва все още за това от зъбните колелца избираме format partition



В новото поле ни пита за име на новят партишън в случаят ще го кръстим share 

Както се вижда в другите полета ще форматираме диска в ext4 файлова система бързо

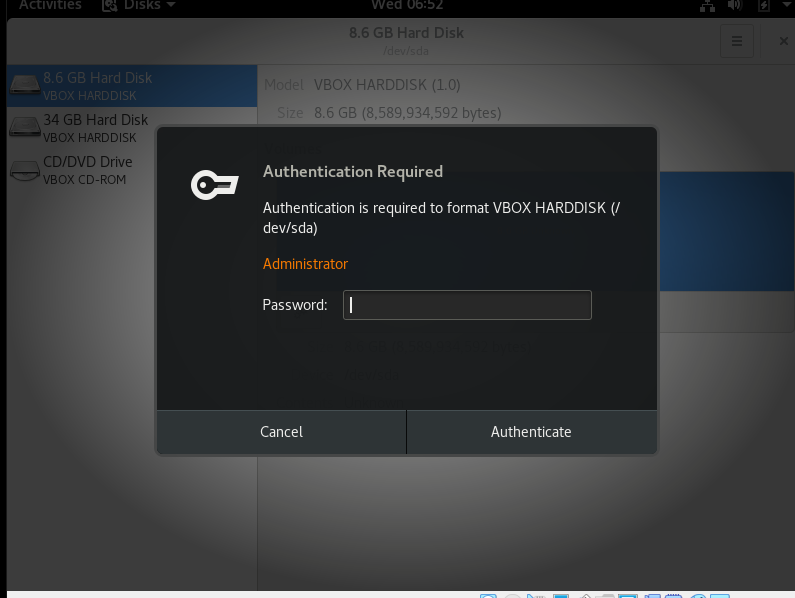
Следващата стъпка е просто да натисмен формат



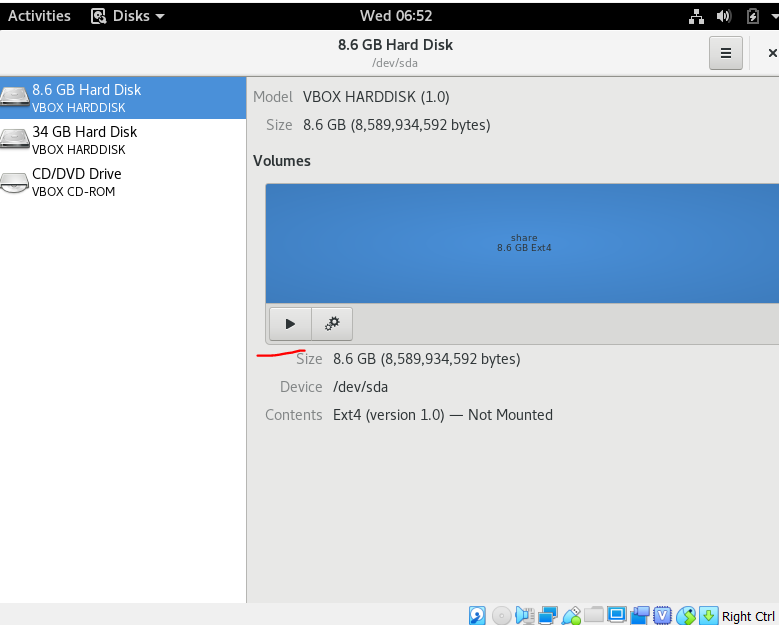
Излиза ни предупреждение в което ни предупреждава че всичката налична информация от дискът ще бъде безвъзвтарно изтрита

( имайте предвид че ако сте избрали грешният диск или диск с информация начин за възстановяване няма така че го правете само когато сте сигурни че това е дискът )

Даваме формат и ни пита за потвърждение чрез root паролата



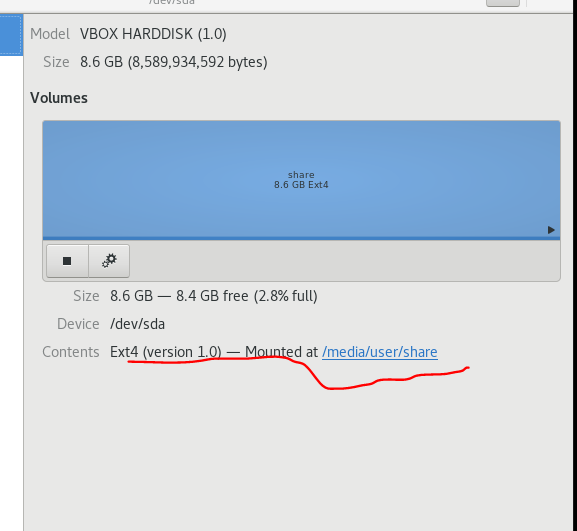
Сега ситемата е готова да запише промените на диска като за целта даваме на триъгълничето долу



Пак ни пита за потвърждение имайте предвид че от тук на сетне промените да необратими

Потвърждаваме с нашата root парола

И както се вижда вече дискът е и маунтнат



Както се вижда parted направи почти всичко автоматично но да се върнем на нашият центос и как нещата се случват когато нямаме графичен интерфейс

Вече форматирахме диска в правилната файлова система но все още не е монтиран

За целта ще използваме командата vgextend

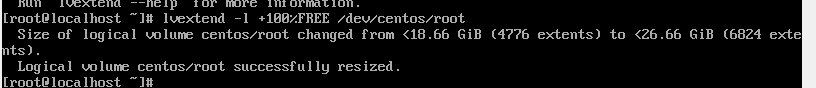


Какво направихме в случаят увеличихме дисковото пространство на нашата операционна система с новият диск

Но ще видиче че ако напишете df –h все още промените не са записани

За целта ще използваме командата lvextend

Синтаксиса който ще ползваме е :



По този начин зададохме файловата система centos да ползва пълният капацитет на дисковете но все още не сме прилкючили

Сега остава да запишеме промените във файловата система

Последната стъпка е да изпълниме следната команда

Resize2fs –p /dev/centos/root

( тук ще отворя една голяма скоба и ще кажа че по този начин командата винаги ще върне грешка и при рестарт операционанта система ще се счупи и ще обясня защо:

Това което направихме е добавихме нов дисков дял към основният и записахме промените като по този начин счупихме основната файлова система ( не се притиснявайте поправимо е ) Но все пак още по време на инсталацията е хубаво да се разделят файловите системи с цел да не се случи точно това проблемът е че вече файловата система не е dev/sda както беше а стана dev/sdb както ще видите от следващият скройншот



Както се вижда dev/sda все още е наличен но вече не е главният в системата

И по този начин бут лоудъра не може да намери файловата система която е на новият дял .

Затваряме скобата )

Сега да го оправиме и да го направим по правилният начин

Ако ще ресайзваме руут паришъна никога не се прави на работеща машина тъй като се получава точно гореописаният пример . За целта се буутва от лайв дистрибуция и се прави същата процедура като се внимава с дисковете и съответно дяловете на тях.

Тъй като преди да започнеме си направихме снимка на виртуалните машини ще се върнем и ще повториме процедурата като си създадеме изцяло нов партишън

И ще повториме процедурата по правилният начин

За целта ще си създадеме нова volume група с името share

Командата е



Имайте предвид че дискът предварително трябва да е форматиран с fdisk

Следващата стъпка е да си направим нова група share на диска за целта командата ще е



По този начин вече имаме правилно конфигуриран диск за последната стъпка

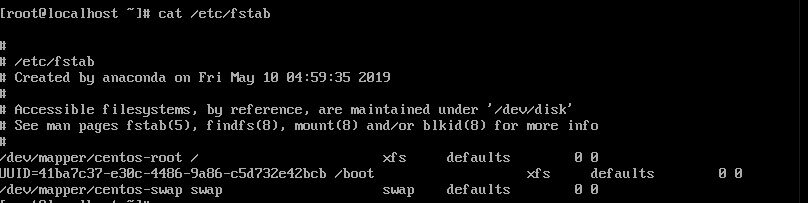
А тя е именно да си създадеме пряк път в нашата домашна директория към него

За целта трябва да редактираме следняит файл /etc/fstab

Но първо да обясниме какво представлява и какво се съдържа в него преди да престъпим към самото редактиране

Това е скройншот на въпросният файл

Както се вижда въпросният файл съдържа информация кой диск в коя директория се намира както и каква файлова система използва



За целта трябва да добавим въпросният дял като спазваме синтаксиса