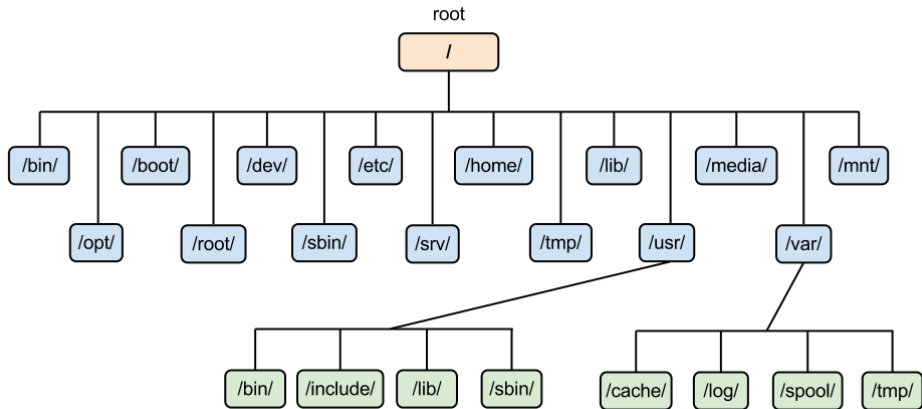


Файловая система в GNU/Linux



Навигиране във файловата система

- ▶ абсолютен и относителен път
- ▶ текуща директория

```
$ pwd
```

- ▶ придвижване из директориите

```
$ cd  
$ cd ~  
$ cd ~peshe  
$ cd -  
$ cd ..  
$ cd ../../../../foo/../../bar
```

Показване на файловете в директория

- ▶ `$ ls`
- ▶ `$ ls -a` - включва скритите файлове (.foo)
- ▶ `$ ls -l` - детайлна информация за всеки файл
- ▶ `$ ls -h` - human readable units

Създаване на файлове и директории

- ▶ `$ mkdir foo` - създава директория "foo".
- ▶ `$ touch bar` - създава празен файл "bar".
- ▶ `$ mkdir /home/human/foo` - директории с абсолютен и относителен път
- ▶ `$ mkdir -p /tmp/foo/bar/baz` - създаване на цялото дърво до директорията

Обем на файловете

- ▶ \$ df - Каква част от файловите системи е заета
- ▶ \$ du - Какъв обем заема директория и файловете в нея
- ▶ -h - human readable units
- ▶ --si - килобайтове и кибибайтове

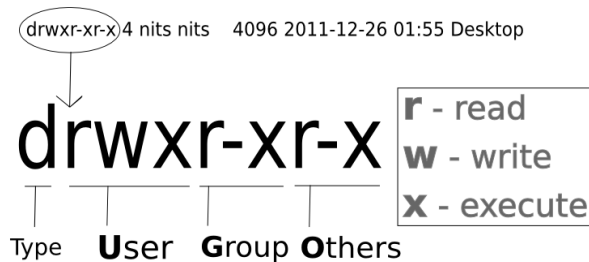
Собственост на файлове и директории

- ▶ всеки файл се притежава от потребител в група (т.е. UID и GID)
- ▶ `chown` - промяна на потребителя собственик на файла
- ▶ `chgrp` - промяна на групата, която притежава файла
- ▶ всъщност, `chown` може да се ползва и за двете:
`chown user:group file`

Активна група

- ▶ текущо логнатият потребител има ефективна група (вижда се с `id`)
- ▶ всеки създаден файл или директория има собственик текущият потребител с група ефективната му група
- ▶ смяна на ефективната група - `newgrp group` (само за текущия терминал, излиза се с `exit`)

Права



File permissions in Linux

- ▶ `ls -l` - виждаме всички права на файлове
- ▶ права за собственика, групата-собственик и всички останали

Права (продължение)



- ▶ Представят се в осмична бройна система
- ▶ $r - 100b - 4$ - само за четене
- ▶ $w - 010b - 2$ - само за писане
- ▶ $x - 001b - 1$ - само за изпълнение

Промяна на права

- ▶ ...и отново `chmod`
- ▶ `-R` за рекурсивна промяна на права
- ▶ `chmod` с числа - \$ `chmod 0664 foo.txt`
- ▶ `chmod` със символи \$ `chmod a=rwx,`
\$ `chmod u=rw,g=rw,o=r foo.txt`
- ▶ `chmod` с частични права \$ `chmod u+r,o-w`

umask

- ▶ Права по подразбиране за новосъздадени обекти във файловата система
 - ▶ Файлове: 666 (rw rw rw)
 - ▶ Директории: 777 (rwx rwx rwx)
- ▶ umask
 - ▶ Дефинира какви права да се **удържат** от правата по подразбиране
 - ▶ Използва се за показване (без параметри) или за смяна на маската
 - ▶ Обикновено се конфигурира от потребителски или системни dot файлове

Специални права

- ▶ set UID (SUID)
- ▶ set GID (SGID)
- ▶ sticky
- ▶ имат напълно различно значение върху директории от това върху файлове

SUID

- ▶ Set **User ID** upon execution
- ▶ задава се с 4 в първата група от правата (e.g. 4644)

4 2 1 4 2 1 4 2 1
rwxrwxrwx

SUID

rwsrwxrwx

USER

The diagram illustrates the process of setting the Set User ID (SUID) bit on a file. It starts with the octal permission 4644, which is broken down into its components: 4 (owner execute), 2 (group write), 1 (group execute), 4 (owner execute), 2 (group write), 1 (group execute), 4 (owner execute), 2 (group write), and 1 (group execute). These are represented as rwxrwxrwx. The SUID bit is then set, changing the first '1' to an 's', resulting in rwsrwxrwx. The 'rws' part is bracketed and labeled 'USER', indicating that the file will be executed with the permissions of the user who owns it.

SGID

- ▶ Set **Group ID** upon execution
- ▶ задава се с 2 в първата група от правата (e.g. 2644)

4 2 1 4 2 1 4 2 1

rwxrwxrwx

SGID

rwxrwsrwx

GROUP

SUID и SGID върху файлове

- ▶ Един изпълним файл обикновено се стартира с контекста на сигурност на потребителя, който го стартира
- ▶ SUID променя този контекст на сигурност
- ▶ Изпълним файл със SUID бит се стартира с контекста на сигурност на потребителя собственик, независимо кой го стартира
- ▶ `$ chmod gu+s`

SUID върху директория

- ▶ се игнорира напълно в UNIX и Linux

SGID върху директория

- ▶ Файловете в тази директория наследяват group ID-то на директорията, а не това на създателя
- ▶ Файловете наследяват и SGID бит-а
- ▶ SGID бит-ът се предава само на новите файлове. На вече съществуващите трябва да се сетне ръчно (при желание, разбира се)
- ▶ Получава се споделено пространство между хората в групата, без да се налага участниците да сменят главната си група, преди да създадат файл в директорията

sticky бит

- ▶ `chmod +t foo; chmod 1000 foo`
- ▶ Ако директория има сетнат sticky бит, потребителите могат да трият и преименуват само собствените си файлове в нея.
- ▶ `/tmp`

UPG - Private Group scheme

- ▶ Това вече не е специален бит
- ▶ Удобен начин за споделяне на файлове в директория, която се използва от група хора
- ▶ Как се имплементира:
 - ▶ Всеки потребител се слага в отделна група
 - ▶ `umask` се слага на 0002
 - ▶ `gid` на директорията се слага на споделена между потребителите GID (e.g., teachers)
 - ▶ Слага се SGID на директорията на проекта

Важни команди за работа с файлове (и директории)

- ▶ `$ mkdir foo`
- ▶ `$ mkdir -p foo/bar/baz`
- ▶ `$ mkdir -m 1755 foo`
- ▶ `$ rmdir foo` - трие директория, ако тя е празна
- ▶ `$ rm foo` - трие файл
- ▶ `$ rm -r bar` - трие директория и всички файлове в нея, ако има такива
- ▶ `$ cp foo bar` - копира файла foo във файла bar
- ▶ `$ cp -r foo bar`
- ▶ `$ mv foo bar` - мести foo в bar. Ползва се и за преименуване
- ▶ `$ touch baba` - създава празен файл baba или, ако има вече такъв файл, променя времето на последно достъпване

Физическа структура

Всяка файлова система се състои от следните неща:

1. Super block - описва състоянието на файловата система (големина на файловата система, големина на блока, указатели към свободни блокове, брой inodes, magic numbers, cure for AIDS, etc)
2. Inodes - всеки файл в системата има inode и всеки inode има файл. В тях се съдържа метадата за файловете

Thus, while users think of files in terms of file names, Unix thinks of files in terms of inodes.

3. Data blocks - в блоковете се пази реалното съдържание на файловете

Inode указатели

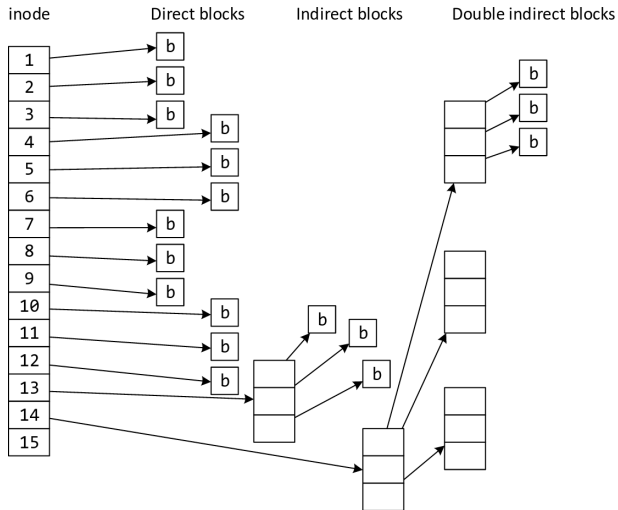
Съдържат метаинформация и препратка към данните. Според POSIX стандарта за всеки обикновен файл трябва да се пази:

- ▶ Големината на файла в байтове
- ▶ UID
- ▶ GID
- ▶ Правата на файла
- ▶ Timestamp - кога за последно е променян inode-ът (ctime, inode change time), кога за последно е променян файлът (mtime, modification time), последно достъпване (atime, access time)

Device ID (this identifies the device containing the file).

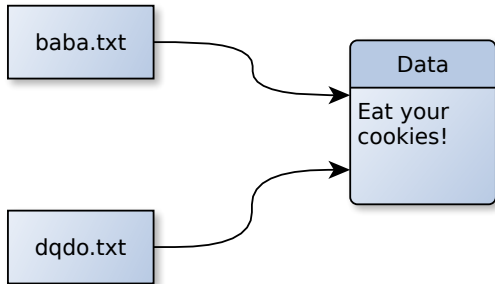
- ▶ Брой твърди връзки
- ▶ Указатели към блоковете, заемани от файла
- ▶ още малко допълнителни неща

Структура на inode указателя



Линкове

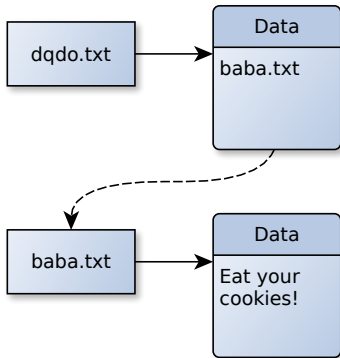
- ▶ **Hard links** / Твърди линкове (хи-хи-хи)
 - ▶ Един запис сочи към inode, към който сочи и друг запис
 - ▶ Все едно един файл да има няколко имена в различни директории
 - ▶ Всички имена сочат към едни и същи данни на диска
 - ▶ Записите не зависят един от друг
 - ▶ Всеки файл в системата има поне един hard link (информацията за броя се пази в inode-овете)



Линкове

► **Symlinks** / symbolic links / soft links

- В data block-а на симлинка се съдържа името на файла, към който сочи
- Пътят може да е абсолютен и релативен
- Ако се изтрие симлинка, това не влияе на таргет файла



Линкове и со.

- ▶ `$ ln foo bar`
 - ▶ създава твърда връзка bar
 - ▶ bar и foo споделят inode и съответно data blocks
 - ▶ `$ stat foo` - **показва информация за файла като брой линкове, номер на inode, etc.**
 - ▶ не може да се създаде твърда връзка към файл извън тази файлова система
 - ▶ не може да се създаде твърда връзка към файл, който не съществува
 - ▶ не може да се създаде твърда връзка към директория
 - ▶ твърдите връзки не заемат допълнителни дата блокове

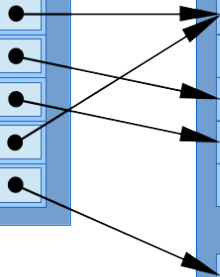
Линкове и со.

Directory /path/to/somedir

filename	inode #
A	1 ●
B	3 ●
C	4 ●
D	1 ●
E	i ●

Filesystem file table

inode #	ref. cnt
1	2
2	0
3	1
4	1
5	0
...	
i	1
...	
n	0



Линкове и со.

- ▶ `$ ln -s foo bar`
 - ▶ създава symlink-a bar
 - ▶ в дата блоковете на bar пише foo
 - ▶ може да се създаде симлинк към директория
 - ▶ може да се създаде симлинк към друга файлова система, през мрежа
 - ▶ може да се създаде симлинк към несъществуващ файл
 - ▶ заема отделни дата блокове
 - ▶ `$ ls -li` - показва inodes

Разширение на файл

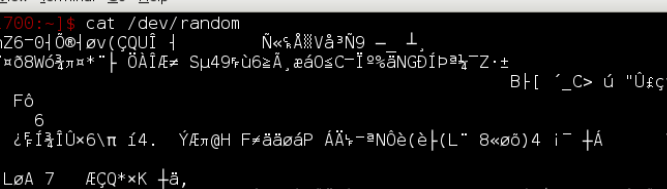
- ▶ В Windows разширенията имат значение
- ▶ В UN*X нямат специално значение за кърнъла
 - ▶ baba.dqdo.chicho.strinka е напълно валиден файл
- ▶ Някои програми използват разширенията (Apache, LibreOffice)
- ▶ `$ file foo` - се използва за определяне на типа на `foo`
- ▶ `magic.mgc`

Баба, дядо и работа с текстови файлове

- ▶ `$ cat foo bar baz` - конкатенира съдържанието на файловете и извежда резултата на екрана
- ▶ `$ more foo bar baz` - показва съдържанието на файловете на отделни страници
- ▶ `$ less foo bar` - показва съдържанието на файловете по умен начин. Удобен е за големи файлове
- ▶ `$ head foo` - извежда първите `n` реда от `foo` (10 по подразбиране, `-n 42`)
- ▶ `$ tail foo` - извежда последните `n` реда от `foo` (10 по подразбиране, `-n 42`)
- ▶ `$ tail -f foo` - чака да се добави нещо към `foo` и го визуализира

Binary data

- ▶ Понякога показването на binary data в терминала може да го прецака



The screenshot shows a terminal window with the title "Terminal". The command prompt is `[lode@A1700:~]$`. The user has entered `cat /dev/random`, and the terminal displays a large block of random, non-printable characters. The output is truncated by the terminal's scrollback buffer, showing only the first and last few lines of the random data. The first line starts with `-.ÍZ\ JñZ6-0{00{0v(ÇOUÎ {` and the last line shows `[r~@A1700:~]$` followed by a green cursor.

Binary data

- ▶ `$ reset` - оправя терминала
- ▶ `Ctrl+J reset Ctrl+J` - в най-лошия случай
- ▶ `$ strings foo` - показва ASCII символите в бинарния файл `foo`
- ▶ `$ xxd foo` - показва hexdump и ASCII dump на файла `foo`.
- ▶ `$ clear` - изчиства буфера на терминала

Търсене из файловата система

- ▶ `find` - супер мощна команда
- ▶ като първи аргумент се подават пътищата, в които да търси. По подразбиране `."`, сиреч - текущата.
- ▶ следващите аргументи задават критерий на търсене
 - ▶ `$ find -name foo` - търси по име на файл
 - ▶ Може да се търси с регулярен израз
 - ▶ Може да се зададе критерий за име, големина, права, създател, група, тип, timestamps
- ▶ последният аргумент казва какво да се прави с откритите файлове. По подразбиране ги изпринтва
 - ▶ `-print` - извежда ги на екрана
 - ▶ `-ls` - извежда допълнителна информация
 - ▶ `-exec` - позволява да се изпълни команда на всеки от намерените файлове

Архивиране и компресиране

- ▶ Архивиране - комбиниране на няколко файла в един, без да променя размера
- ▶ Компресиране - смачкване на един файл до намаляване на размера му
- ▶ Някои команди правят само едното, някои само другото, а някои и двете

Архивиране

- ▶ `$ tar -c -f foo.tar bar baz` - създава архива foo от файловете bar и baz
- ▶ `.tar` - tarball - пази структурата на файловете и директориите, пази метадатата
- ▶ `$ tar -x -f foo.tar` - разархивира архива foo
- ▶ `-v` for Vendet.... verbose

Компресиране

- ▶ `$ gzip foo` - изтрива файла `foo` и създава компресирания файл `foo.gz`
- ▶ `$ gzip -d foo.gz; $ gunzip foo.gz` - за декомпресия
- ▶ `$ xz foo` - изтрива файла `foo` и създава компресиран файл `foo.xz`
- ▶ `$ xz -d foo.xz` - за декомпресия

► коефициент на компресия

gzip	bzip2	lzma	lzma -e	xz	xz -e	lz4	lzop
26.8%	20.2%	18.4%	15.5%	18.4%	15.5%	35.6%	36.0%
25.5%	18.8%	17.5%	15.1%	17.5%	15.1%	35.6%	35.8%
24.7%	18.2%	17.1%	14.8%	17.1%	14.8%	35.6%	35.8%
22.0%	17.6%	14.9%	14.6%	14.9%	14.6%	-	35.8%
21.5%	17.2%	14.4%	14.3%	14.4%	14.3%	-	24.9%
21.4%	16.9%	14.1%	14.0%	14.1%	14.0%	-	24.6%
gzip	bzip2	lzma	lzma -e	xz	xz -e	lz4	lzop

► време за компресия

gzip	bzip2	lzma	lzma -e	xz	xz -e	lz4	lzop
8.1s	58.3s	31.7s	4m37s	32.2s	4m40s	1.3s	1.6s
8.5s	58.4s	40.7s	4m49s	41.9s	4m53s	1.4s	1.6s
9.6s	59.1s	1m2s	4m36s	1m1s	4m39s	1.3s	1.5s
14s	1m1s	3m5s	5m	3m6s	4m53s	-	1.5s
21s	1m2s	4m14s	5m52s	4m13s	5m57s	-	35s
33s	1m3s	4m48s	6m40s	4m51s	6m40s	-	1m5s
gzip	bzip2	lzma	lzma -e	xz	xz -e	lz4	lzop

Архивиране И компресиране

- ▶ Доста команди могат да правя и двете
- ▶ `tar -c --gzip -f foo.tar.gz bar baz qux`
- ▶ `tar -c --xz -f foo.tar.xz bar baz qux`

