3. Файлове и директории  
• Filesystem Hierarchy Standard (FHS) ;  
• Смяна на директории (cd);  
• Преглед на съдържанието на директория (ls);  
• Създаване , копиране, преместване и триене на файлове и директории;  
• Блокове, inodes, връзки (hard и symbolic links);  
• Файлови типове , преглед на съдържанието на файлове (cat, more, less, head, tail);  
• Намиране на файлове (find, locate, which, whereis) .

Част 3

**Файлове и директории**

**В тази секция**

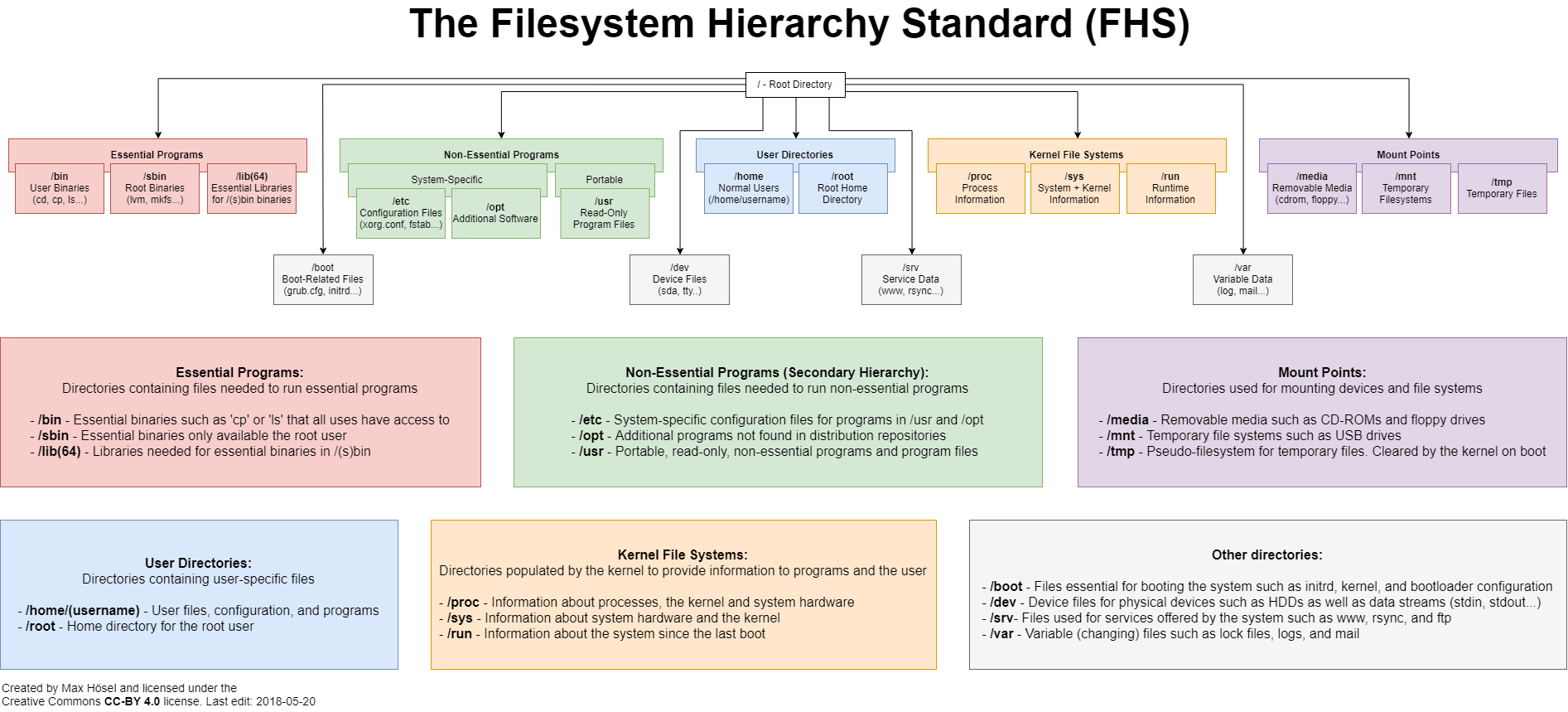
• Filesystem Hierarchy Standard (FHS) ;  
• Смяна на директории (cd);  
• Преглед на съдържанието на директория (ls);  
• Създаване , копиране, преместване и триене на файлове и директории;  
• Блокове, inodes, връзки (hard и symbolic links);  
• Файлови типове , преглед на съдържанието на файлове (cat, more, less, head, tail);  
• Намиране на файлове (find, locate, which, whereis) .

Накратко ще се запознаеме със файловата система в линукс както и да навигираме в нея чрез команди като cd .

Стандартът за йерархия на файловата система (на английски: Filesystem Hierarchy Standard съкр. FHS) в базираните на Linux операционни системи определя структурата на йерархията на директориите, както и съдържанието на файловете в тях. Стандартът е установен и разширен за файловата система на операционната система BSD. Файловата система се поддържа от Linux Foundation с текуща версия 2.3, която бе обявена през 2004 г.

Основната директория в Linux се нарича root и е маркирана с наклонена надясно черта /. Тази директория съдържа всички други директории, които съставляват цялата система.

На следната диаграма е показана стандартната файлова система в линукс операционните системи

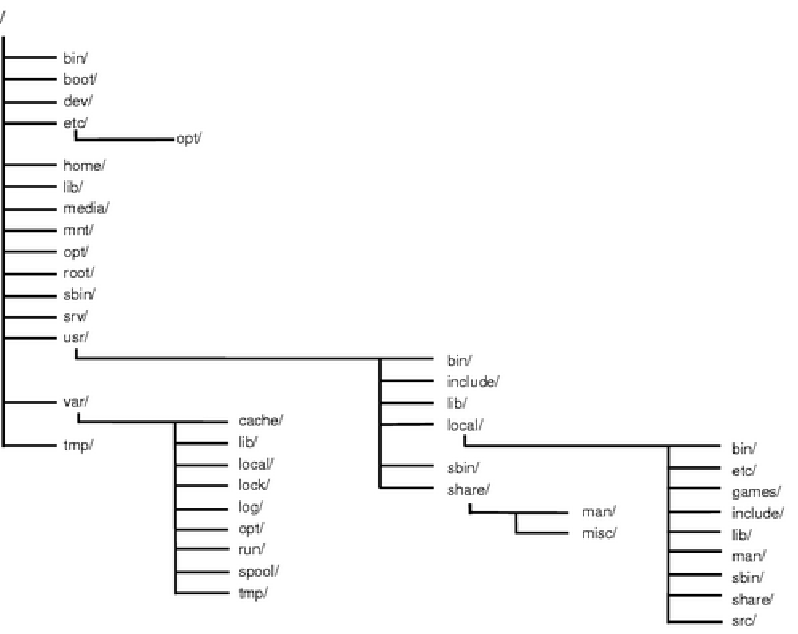


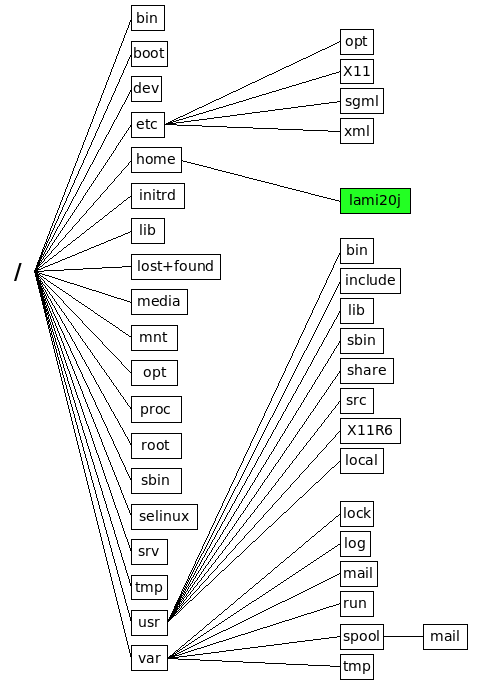
За тези които желаят да се запознаят със целият стандарт той е общодостъпен на следният линк : <https://refspecs.linuxfoundation.org/FHS_3.0/fhs-3.0.pdf>

Макар че ще го обясниме и с настоящата лекция .

Както споменахме главната директория се нарича root директория в нея са поместени всички компоненти на операционанта система .

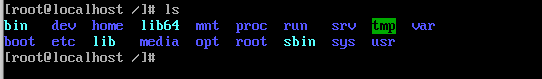
Както ще видите на следващата диаграма структурата

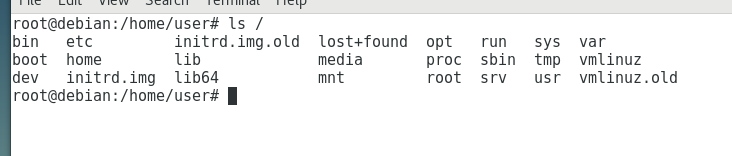




Всяка отделна директория съдържа поддиректории които варират в зависимост от инсталираните приложения тоест първата колона на горната схема е еднаква за всички операционни системи като в зависимост от инсталираните приложения втората колона може да има или няма някои от въпросните директории .

Да погледнем във нашите виртуални системи как реално изглежда въпросната структура :



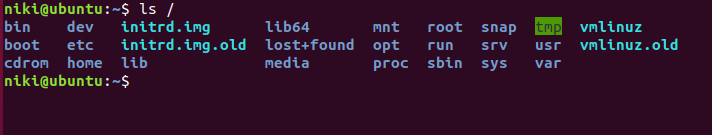


Както се вижда основните директории и в двете операционни системи са еднакви като има и няколко доопълнителни директории които релано правят разликите между различните линукс дистрибуции . Като пример ще ви покажа и файловата структура в BSD и убунту

BSD



Ubuntu

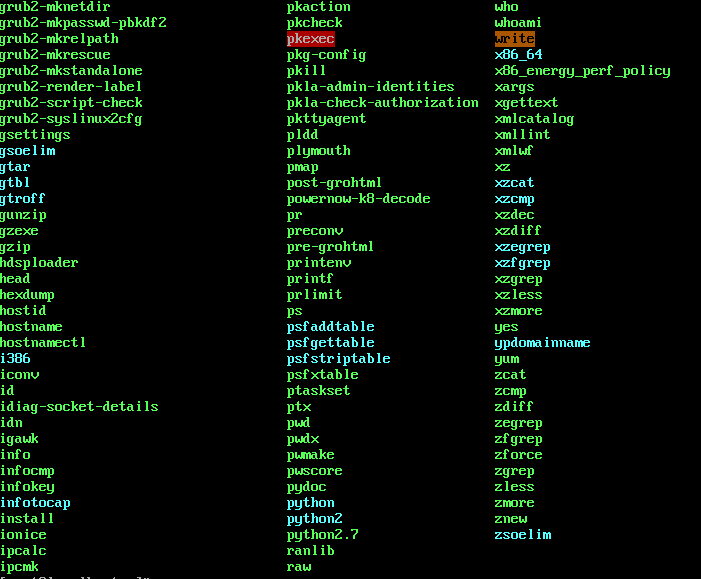


Както се забелязва между убунту и дебиан няма разлика във файловата структура .

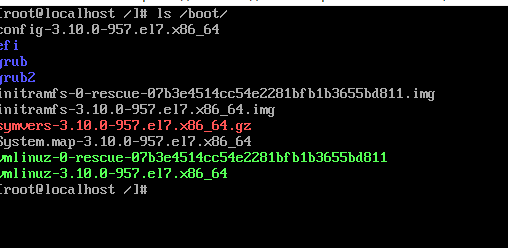
Другото което се забелязва че всичките операционни системи имат сходни папки . Сега ще обясниме по подробно коя папка за какво е

Както споменахме основната директория е root и се озобразява с /

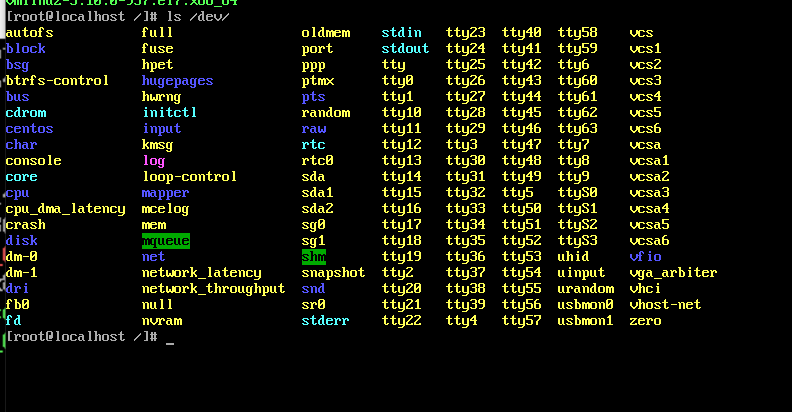
Следващата директория е /bin в нея се съдържа списък със командите в операционната система като например ls ,cd , cat …



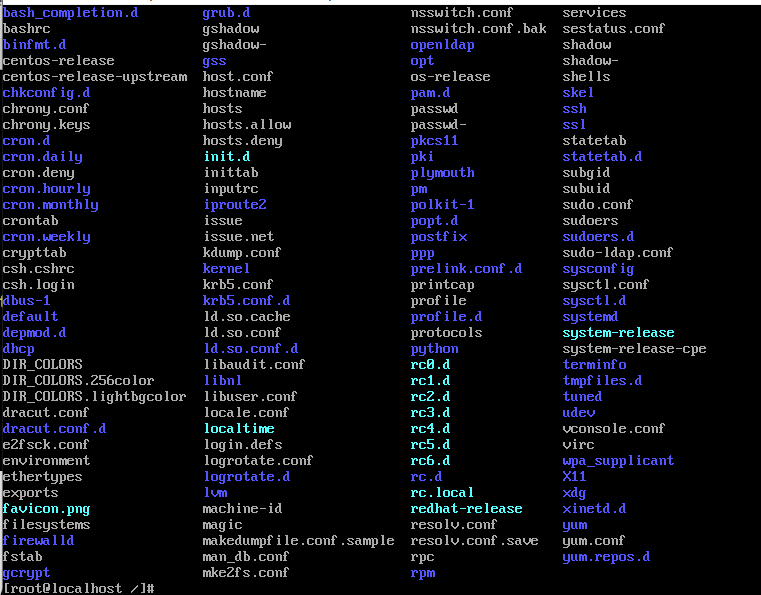
Следващата директория е /boot в нея се съдържат всички необходими файлове за стартиране на операционната система както и самият кернел



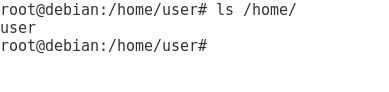
Следващата директория е /dev в нея се съдържа списък на всички компоненти на системата на системата

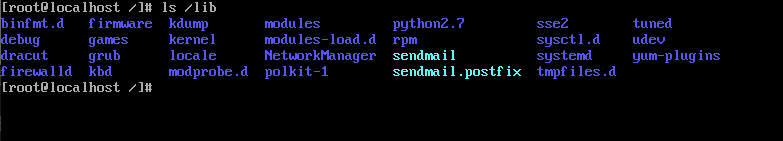


Следващата директория е /etc в нея се съдържат конфигурационните файлове на операционната система и всичкият инсталиран софтуеар на нея .

 както се вижда има приложения които имат самостоятелни папки .

Следващата директория е /home в нея се съдържат всички файлове на потребителите в системата както и домашните им папки



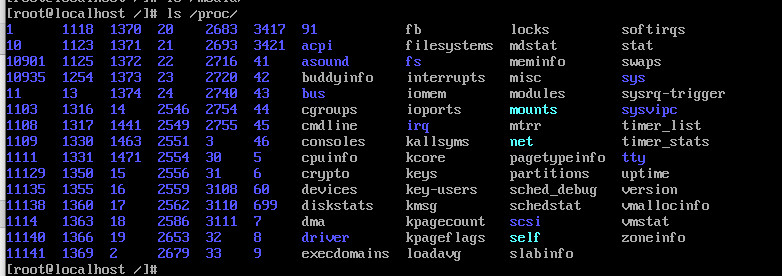
Следващата директория /lib съдържа библиотеки необходини за работата на отделните команди 

Следващата директория е /media в нея както ще видите е празна но предназначението е и ако закачите флашка или cd например там се показва както и други устройства които временно присъстват в операционната система

Със сходна идея и е /mnt директорията само че с разликата че тук се закачат папки от отдалечени сървъри както и мрежово споделени ресурси

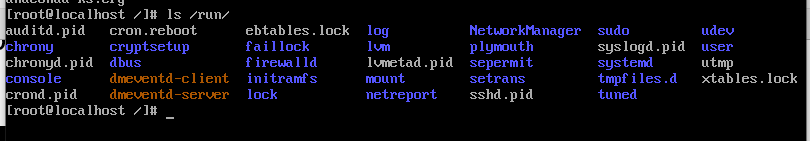
Доректорията /opt съдържа допълнителен софтуеар

Директорията /proc съдържа списък на всички стартирани процеси в операционната система

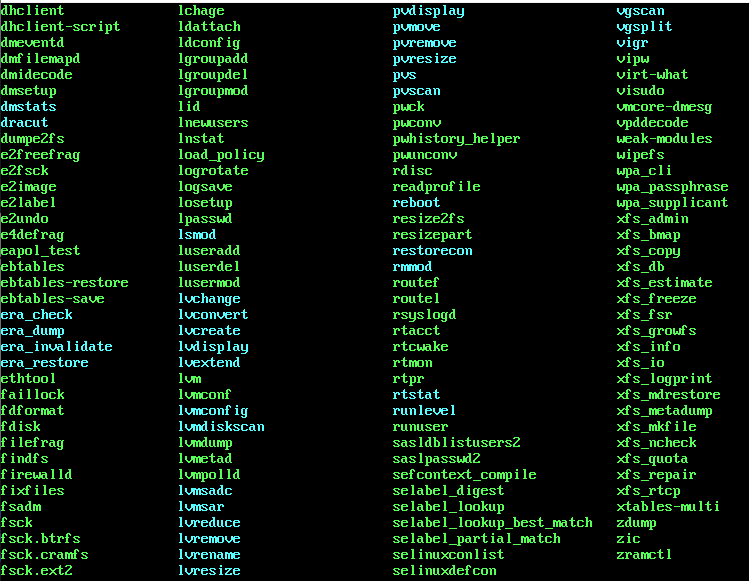


Съответно /root да не се бърка със / е домашната папка на root потребителят

/run директорията съдържа променливи за стартиране на определени програми

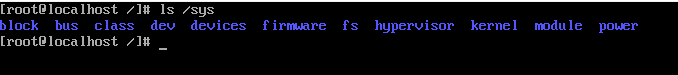


/sbin директорията съдържа системни файлове .



/srv директорията се ползва за специфична информация която споделя сървърът като например фтп файлове уеб страници и т.н.

/sys директорията съдържа информация за драйвъри и някои кернел компоненти



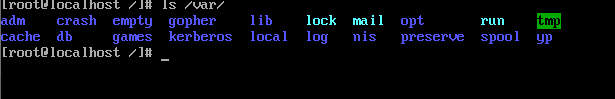
/tmp директорията съдържа временни файлове на някои приложения които след рестарт не винаги се изтриват



Следващата директория е /usr в нея се съдържа информация за приложенията които могат да бъдат изпълнявани от всички потребители в системата



Директория на променливите /var в тази директория се намират файлове които често се променят като например лог файловете на системата



Общо взето това са по важните директории в линукс операционните системи .

Следващото с което ще се запознаеме е командата **cd** или change directory

Командата е една от най често използваните във всички операционни системи и се използва за навигация между отделни директории както видяхте по рано .

Синтаксисът е изключително лесен да бъде запомнен но ще покажеме и някои от по малко известните опции на командата .

Ако искаме да отидем например във нашата /var директория изпълняваме командата по следният начин cd /var

 както виждате от лявата срана наклонената чертичка се смени на var което означава че се намираме в тази директория вече ако искаме да се върнем директория назад cd ..

Може да прескачаме от директория в директория като спазваме правилото cd /директорията която искаме

За да се върнем едно ниво нагоре cd ..

Ако желаем да се вътрнем в предишната директория където сме били вместо двете точки използваме тире и командата изглежда така cd –

Ако желаем да се върнем в директорията откъдето сме дошли добавяме още едно тире cd - –

Ако искаме да се качиме 2 директории нагоре синтаксиса е cd ../ ../

за 3 cd ../ ../ ../ и т.н.

за да се върнем в нашата домашна директория е достатъчно само cd ~ или cd само .

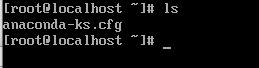
Следващата команда се ползва не по малко от cd а именно **ls** или list

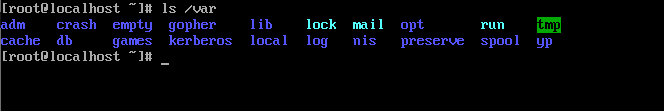
Командата се ползва за да се види съдържанието да дадената папка в която сме или на някоя папка в която ни интересува какво има

Например намираме се в нашата домашна папка и иснаме да видиме съдържанието на var

Синтаксисът е следният ls /директорията която искаме да разгледаме

Ако не се посочи директория ни показва съдържанието на тази в която се намираме

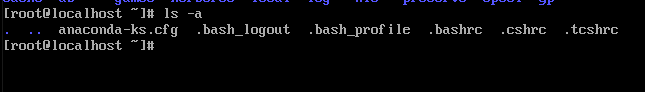




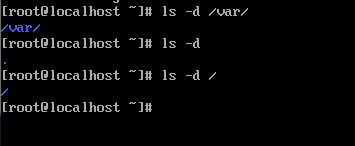
Съответно като всяка команда в линукс може да се комбинира с различни опции

Ето го и списъкът на всички променливи и какво се показва

Ls –a показват се и скритите файлове в директорията обинкновенно са със точка отпред



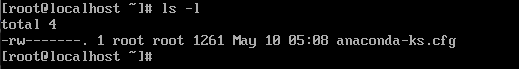
ls –color осветява по различен начин папките и файловете за да е по лесно различимо

ls –d показва папките само

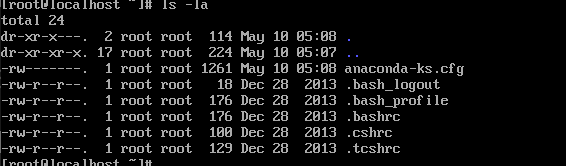
ls –F добавя екстра символ \*/=>@| към записите

ls –i показва и индекс номерът на файла в операционната система 

[ls -l](https://www.rapidtables.com/code/linux/ls/ls-l.html) покзва повече информация за файловете



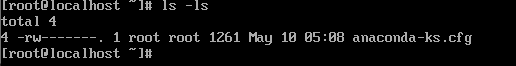
ls –la повече информация за файловете включително и скритите такива



Ls –lh показва файловете в дълъг формат но големината им в разбираем формат



Ls –ls в дълъг формат със размер на файла



Ls –r сортира файловете в обратен ред

Ls –R сортира рекурсивно директориите

Ls –s покзва размерът на файловете

Ls –S сортира файловете по размер

Ls –t сортира по време и дата

Ls –x сортира спрямо разширението на файлът

Създаване , копиране, преместване и триене на файлове и директории;

Като във всяка операционна система и в линукс могат да се създават файлове и папки

Командите които ще използваме са **cp mv rm** **tourch mkdir**

**Съответно cp е съкратено от copy и се ползва за копиране на файл или папка**

Mkdir съкратено от make directory (създай папка)

Mv съкратено от move (премести)

Rm (remove ) или премахни

Tourch е малко по специфична команда която създава празен файл без да влиза в него като в зависимост от параметрите може да създава празни файлове с определена големина ще видиме по подробно след малко .

Първо в нашата home директория ще си създадеме папка test с командата mkdir test



Ако изпълниме командата ls ще видим нашата нова папка test



Сега ще влезем в папката с командата cd test/

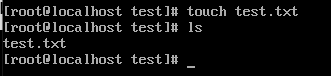


Ако изпълниме ls ще видиме че е празна

Сега ще си създадем празен файл test.txt с командата touch

Touch test.txt

Проверяваме с ls

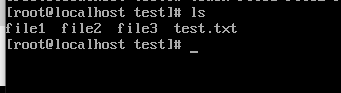


И вече имаме създаден празен файл

Ако искаме да създадеме повече от един файл да кажем file1 file2 file2 не е нужно за всеки един по отделно да изпълняваме горната команда може просто да ги изброиме след командата

Touch file1 file2 file3

 и създадохме 3 файла с една команда



Съответно както по рано споменахме всяка команда може да бъде изпълнена с различни параметри които имат различна функция

touch -a сменя последната дата на модификация на файла

touch -c проверява дали файлът е създаден или не ако не е създаден не го променя

touch -c-d използва се за да промени последната дата и час на файла

touch –m променя само часа

touch -r копира времето на промяната от един файл на друг като синтаксиса е touch -r second\_file\_name first\_file\_name

touch -t YYMMDDHHMM файл използва се за да се зададе специфична дата на файлът

следващата команда която ще разгледаме е cp

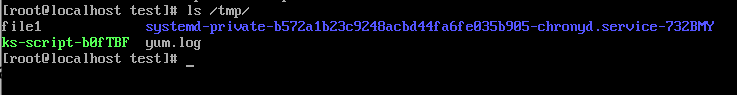
както споменахме използва се за да копира един файл в друг файл или на друго място

да кажем че искаме да копираме нашият файл test.txt във text1.txt

за целта в терминала изпълняваме командата cp test.txt text1.txt



Като може да копираме и в различни директории не е задължително да е въщата директория . Да кажем че новосъздаденият file1.txt трябва да е във /tmp директорията тоест командата ни ще изглежда по следният начин

Да предположиме че по погрешка сме преместили нашият file1 В Друга директория

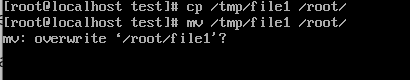
И искаме да го върнем обратно без да оставяме копие за целта ще използваме командата mv

Ще преместин нашият фаил1 от /tmp директорията в нашата домашна папка

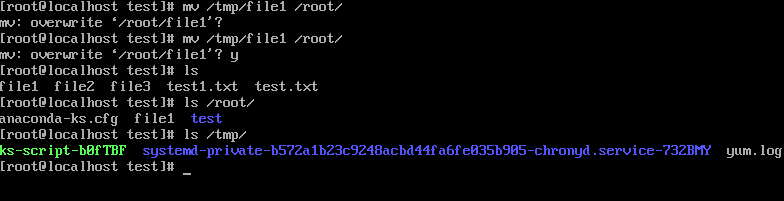
Както знаете се намираме в нашата папка тест обаче файлът вече съществува там да предположиме че е със същото име но различно съдържание за това ще го копираме в /root

Командата е mv /tmp/file1 /root/

Като всяка система ако файлът съществува ще попита дали желаеме да го презапишеме като опциите са y/n по подразбиране винаги е да



Може да въведете от клавиатурата Y и да натиснете ентър или само да натиснете ентър



Както виждате файлът се премести в нашата папка или където му посочиме

Синтаксисът е много прост mv /път/до/файла /място/където/да/се/съхрани

ако искаме да ни пита за всеки файл по отделно дали желаем да го преместим може да ползваме mv –i това ще ни пита за всеки файл по отделно

ако искаме да преместиме файл в папка в която съществува файл със същото име без да го презаписваме може да ползваме mv –n

може също да местим и папки не само файлове но и папки по същият начин

следващата команда е rm

работата с нея е много проста rm файлът\_който\_искаме\_да\_изтрием

или ако искаме да изтрием нашият файл1 от руут директорията просто изпълняваме

rm file1

може да се посочи и пълният път до файлът който ще триеме

командата пита дали наистина сте сигурни че желаете да изтриете файлът



**ВАЖНО изпълнението на командата е необратимо и файлът не може да бъде възстановен освен ако няма бекъп . файлове и папки които са изтрити в линукс не могат да бъдат възстановени така че използвайте внимателно командата .**

Потвърждаваме че желаем да изтрием файлът и вече несъществува

Командата rm не може да трие папки за това се използва в комбинация с rf

Или ако искаме да изтрием нашата папка test1 командата изглежда така

rm –rf test1/

със root потребителят може да триеме всичко включително и операционната система за това никога не изпълнявайте командата rm –rf / това ще изтрие цялата операционна система и винаги проверявайте дали това което искате да изтриете е това което в действителност триете

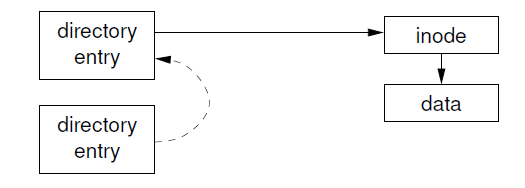
Ако сте обърнали внимание по нагоре някои от файловете бяха с различен цвят например /tmp

Директорията на следната картинка



# 1.     Symbolic Links

Символична файлова връзка (Symbolic link или  symlink) е псевдо-файл, който представлява алтернативно име за друг файл или директория. Съдържанието на symlink е реалното име на сочения файл.



При използване на symlink в команда, ядрото замества symlink с неговото съдържание преди изпълнението на командата.

Този механизъм позволява един файл да бъде  на едно място, а да претендира, че се намира на други места. Например да се разпространи информацията от една индивидуална файлова система върху различни дискови дялове.

Командата ***ln -s*** се използва за създаване на символична връзка със синтаксис първи параметър- реалното име, а втори – създаваното:

***$ ln -s real-file file-link***

***$ ln -s real-dir dir-link***

Командата ***ls -l*** показва символичната файлова връзка:

Пример:

***$mkdir testd***създаване на директория

***$cd testd***

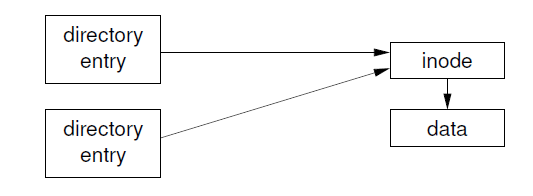
***$echo test information >file\_org***създаване на оригиналния файл

***$ln -s file\_org file\_ln***създава символичната връзка

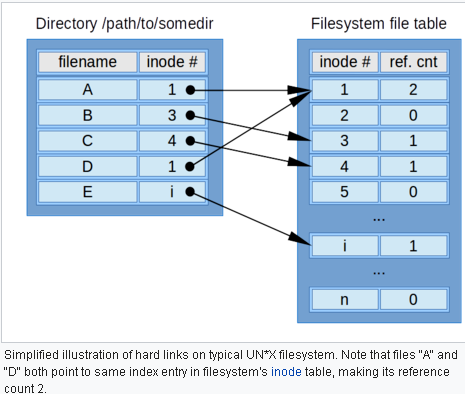
***$ls -l file\_ln***

# 2.     Hard Links

За разлика от символичните връзки, които сочат към името на друг файл, „твърдите връзки“ се насочват към inode на съществуващ файл. По този начин различни имена на файлове се свързват към един и същ inode.



Една илюстрация от wikipedia



Командата е същата както при символичните връзки но  без опцията ***–s***:

***$ ln real-file file-hrd\_link***

Като продължение на горния пример:

***$echo test information2 >file\_org2***създаване на оригиналния файл

***$ln file\_org2 file\_hrdl***създава символичната връзка

***$ls –i file\_hrdl***проверка на inode

***$ls –i file\_org2***същият inode

***$stat file\_org2***обърнете внимание на броя на връзките в inode

# 3.     Сравнение между Symlinks and Hard Links

|  |  |
| --- | --- |
| Symlinks | Hard Links |
| Symlinks са съществено различни от нормалните файлове и ясно могат да бъдат различени от файловете към които сочат    Symlinks могат да сочат към всички видове файлове (нормални файлове, директории, специални файлове, Symlinks …)    Symlinks са насочени към имената на файловете и затова могат да сочат към файлове от различни индивидуални файлови системи.    Ако бъде преименуван, преместен или изтрит оригиналния файл Symlink става невалидна (broken) | Множеството Hard Links стил имена на един и същи файл са неразличими. Понятието Hard Link е до голяма степен условно    Hard Links не могат да сочат към директории (и в някои не Linux системи към Symlinks)    Hard Links работят с inode и са валидни само в рамките на една индивидуална файлова система.      Преименуването преместването или изтриването на „оригиналния“ файл няма ефект върху Hard Links файловете. |

# 4.     Команди и файлови връзки

Повечето команди работят с файловите връзки по същия начин, както с нормалните файлове, като някои имат опции, които показват дали Symlink трябва да бъдат следвани. Има обаче особености за някои команди.

Архивиращата tar команда отчита ако в архива има два или повече Hard Links файла и съхранява тази информация правилно (без дублиране на информацията от файла). По подразбиране тя съхранява Symlinks в архива. Ако бъде използвана с опция  ***-h  (--dereference),***тя съхранява не Symlink, а сочения файл в архива.

Командата ***cp*** по подразбиране игнорира файловите връзки. За да бъдат запазени се използва опция ***-d (--no-dereference)***. При рекурсивно копиране за да бъдат запазени Symlinks се използва опцията ***-R (--recursive)***. Опцията ***-a (--archive)*** включва и двете опции  ***-d*** и ***-R***.

# Inodes и директории

В Unix/Linux индивидуална файлова система всеки файл се представя от inode (index node). Това е структура от данни в традиционна Unix/Linux файлова система като ext2, ext3 или ext4, която съхранява метаданните на файла и се намира на отделно място, а не в директорията. В индивидуалната файлова система inode са номерирани (номерът често се наричат inum).

Всяка индивидуална файлова система (дисков дял, CD, DVD …) притежава свое собствено множество от inodes. В обединената файлова система може да има няколко файла с еднакъв inode номер.

Когато се инициализира (форматира) физическо устройство с Unix/Linux файлова система се създават фиксиран брой inode. Броят им задава максималният брой файлове от всички типове, които едновременно могат да се поместят в индивидуалната файлова система. Типично се отделя по 1 inode за от 2 до 8 KB памет.

В момента на създаването на файл той получава свободен inode, съхраняващ следната информация:

* Собственика и потребителската гупа на файла.
* Типа на файла (regular, directory, ...)
* Параметрите за достъп (read, write …)
* Датата и времето на последния достъп и последната проняна (modification) на файла. UNIX /Linux не съхранява кога първоначално е създаден файла.
* Дата и времето на последната промяна (change) в inode ( например в параметрите за достъп или собственика на файла)
* Броят на връзки (links) към файла (soft/hard)  – ще бъдат описани по-късно.
* Размера на файла
* Адрес дефиниращ местоположението на файла.

С други думи може да се счита, че inode идентифицира еднозначно файла.

* Името на файла се съхранява не в inode, а в директорията.
* Директорията се съхранява на диск, като списък от файлове и имена на директории.
* Всяко име има асоцииран inode номер.
* Отделянето на имената от inode означава, че може да има много имена свързани към един inode.

Командата ***ls –i*** извежда inode номера на файла:

***$ ls -i /etc/passwd***

Командата ***stat*** позволява извеждането на inode номера и атрибутите на файла:

***$ stat /etc/passwd***

До тук се запознахме със съдържанието на операционната система линукс но не всичко .

В тази секция ще се научиме как да четем файлове и да извличаме различна информация за тяс с командите cat, more, less, head, tail

Както по нагоре показахме ls показва файловете във папките но ако искаме да прочетем някой специфичен файл да кажем нашият файл1 (както знаем той е празен )

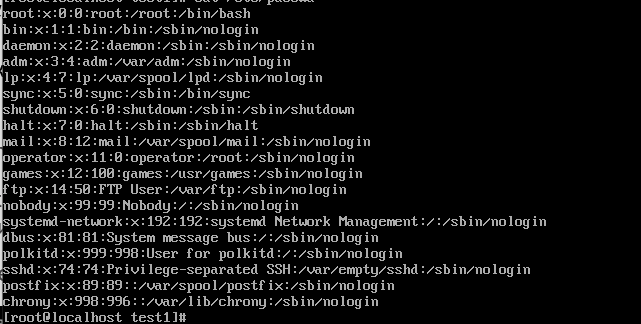
Командата cat се използва за да визуализира съдържанието на файл

Ще използваме командата cat file1



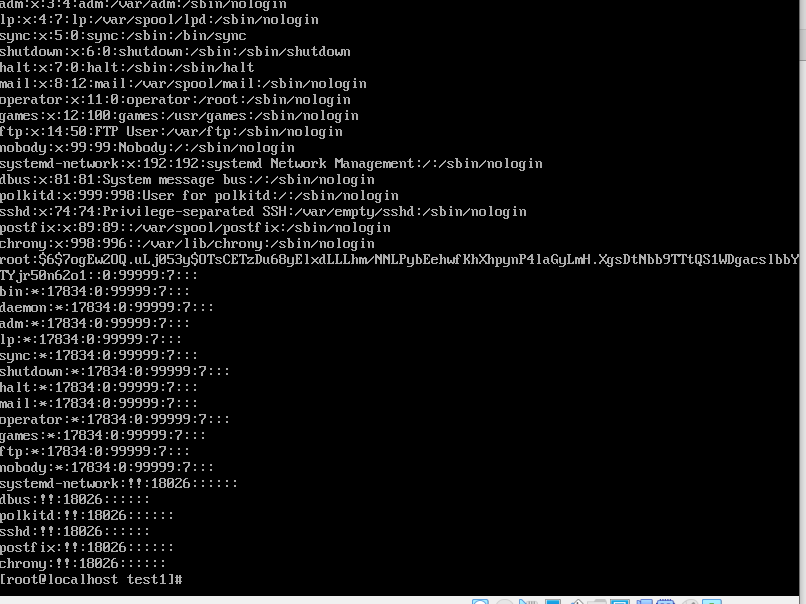
Само че в реалният свят това е идеалният сценарий и често файловете не само че не са празни ами съдържат и хиляди редове код .

Например да кажеме файлът със потребителите в линукс



Или ако се налага да визуализираме 2 файла едновременно

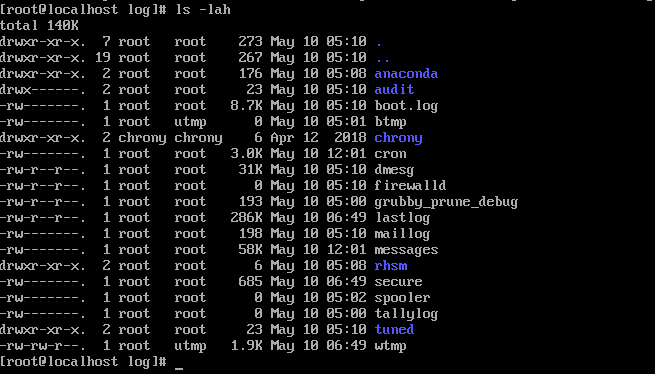
Да кажем освен файлът с потребителите искаме да видим и файлът с паролите



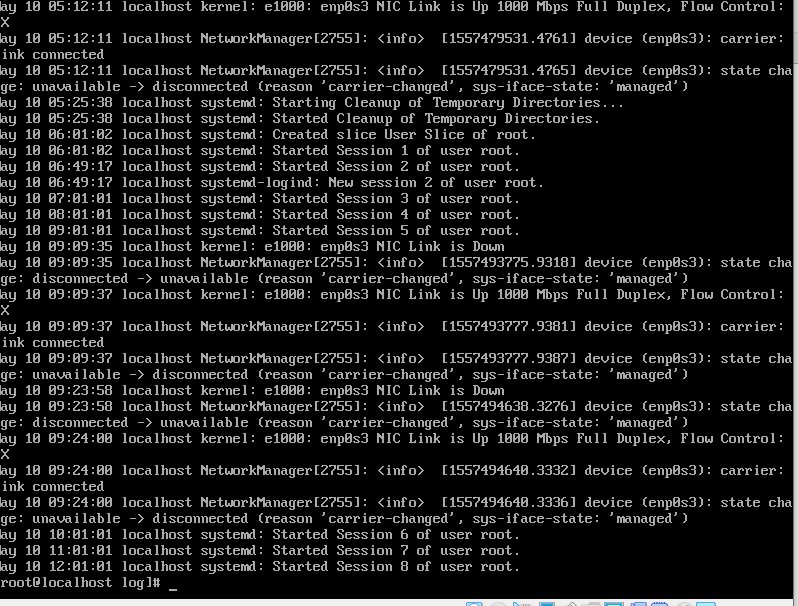
Много често командата се комбинира с команди като more less head и tail

Това се налага тъй като много често снея се четат огромните лог файлове които могат да стигнат и над гигабайт и доста от текстовите редактори ще се затруднят да ги отворят .

Мапример ще отидеме в нашата лог директория на линукс и ще покажеле как се комбинират въпросните команди с цел по лесно четене на лог файлове.



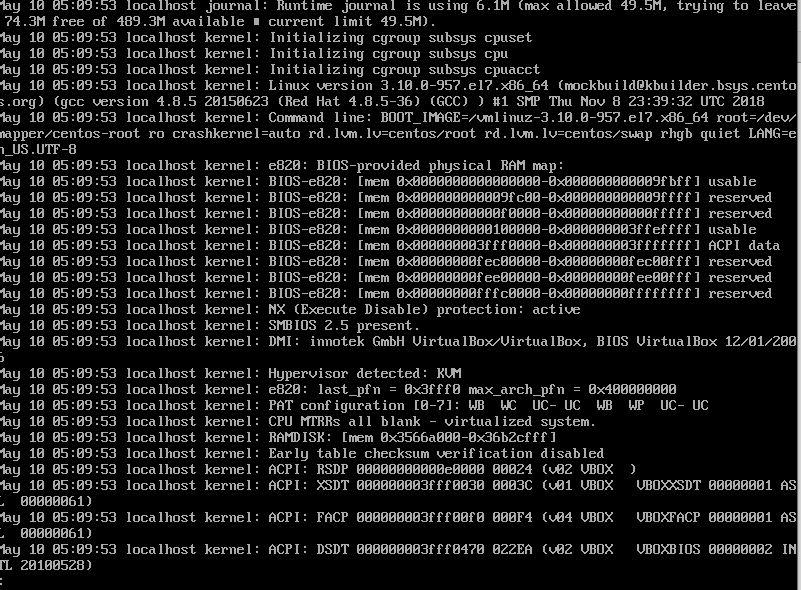
Ако изпълниме командата cat messages интерпретатора ще ни изкара вскичко от файлът на екрана



Както забелязахте по голямата част от файлът само премина през екрана без да имаме възможност да го прочетем

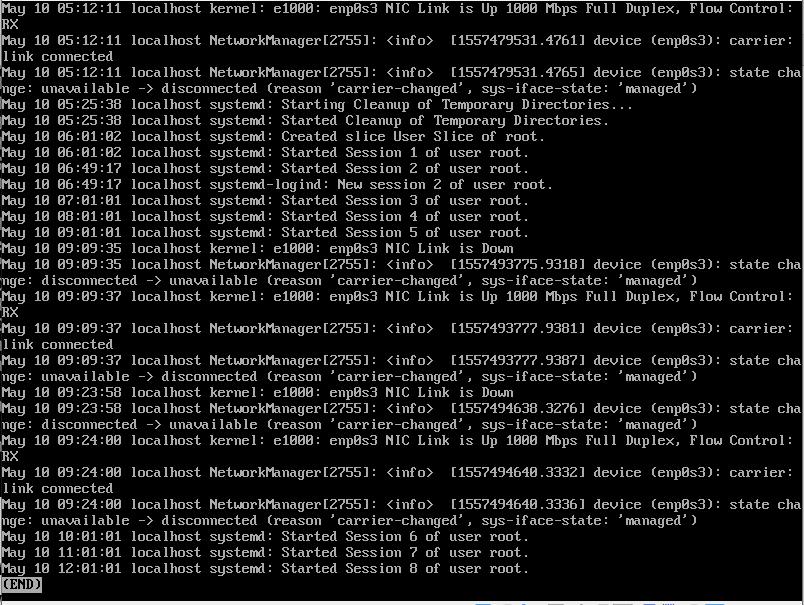
За това ще използваме командата в комбинация с less

И съответно нашата команда ще стане cat messages | less



Както виждате четеме файлът от самото начало като накрая има 2 точко които ни показват че има още от файлът ако натиснеме ентър ще се продължи и така до краят или може да ползваме PgDn на клавиатурата да ни ги мести екран по екран ако не искаме ред по ред

Накраят излиза end във файлът



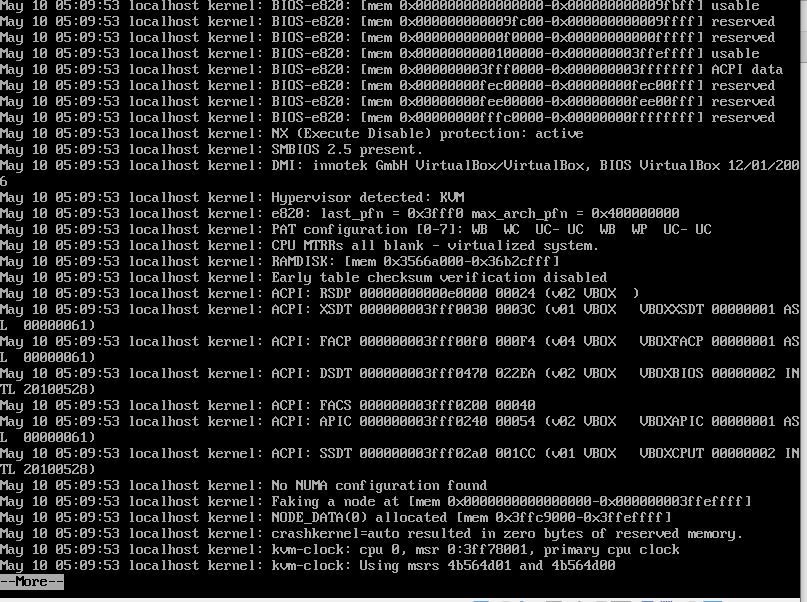
За да излеземе натискаме q (quit)

Това е много полезна опция при четене на големи файлове

Обаче ако ни трябва повече информация опцията която ще ни я покаже е more

Въпросната наша команда вече ще изглежда по следният начин **cat messages |more**

Както ще забележие на екрана излиза повече информация за това което се случва във системата



Навигацията е п осходен начин като с less тоест ентър за ред надолу и накрая на файла q

За да излезнем .

Командите more и less може да се ползват и самостоятелно като не се различават много от предишният пример

Например less messages е същото като cat messages |less

Съответно и more messages е същото като cat messages |more

За зазлика от тези 2 комди следващите дават доста по ограничена информация а именно head и tail

Head командата изкарва само първите 10 реда на файлът ако не е посочен параметър

Обаче често се налага да се четат повече от 10 реда

Синтаксисът е като на всички линукс команди а именно

head [options] [file(s)]

опциите са следните

-n брой линии които да ни покаже

-c байтове които да ни покаже

-q в тих режим не визуализира нищо

-v жизуализира всичко

Да се върнеме на нашият пример с лог файлът да кажем че искаме да прочетем първите 15 реда

Командата ще изглежда по следният начин head –n 15 messages

Tail командата следи за промени във файла и изкарва само промените

По подобие на head и tail може да се ползва с различни опции

Синтаксисът е същият

tail [OPTION]... [FILE]...

ето го и списъкът с възможните опции

**-c, --bytes=*K***

output the last K bytes; alternatively, use **-c** +K to output bytes starting with the Kth of each file

**-f, --follow[={name|descriptor}]**

output appended data as the file grows; **-f**, **--follow**, and **--follow**=*descriptor* are equivalent

**-F**

same as **--follow**=*name* **--retry**

**-n, --lines=*K***

output the last K lines, instead of the last 10; or use **-n** +K to output lines starting with the Kth

**--max-unchanged-stats=*N***

with **--follow**=*name*, reopen a FILE which has not changed size after N (default 5) iterations to see if it has been unlinked or renamed (this is the usual case of rotated log files). With inotify, this option is rarely useful.

**--pid=*PID***

with **-f**, terminate after process ID, PID dies

**-q, --quiet, --silent**

never output headers giving file names

**--retry**

keep trying to open a file even when it is or becomes inaccessible; useful when following by name, i.e., with **--follow**=*name*

**-s, --sleep-interval=*N***

with **-f**, sleep for approximately N seconds (default 1.0) between iterations. With inotify and **--pid**=*P*, check process P at least once every N seconds.

**-v, --verbose**

always output headers giving file names

**--help**

display this help and exit

**--version**

output version information and exit

за разлика от head tail следи за промени в краят на файлът като за дебъгване на грешки в реално време най често се използва със –f опцията

например искаме да видиме в реално време какво се случава в нашата система тоест трябва постоянно да правиме cat на нашият messages файл .

Е tail ни спестява това упражнение

За целта нашата команда ще изглежда tail –f messages



Както забелязвате при настъпването на някъкво събитие в системата излиза нов ред

За да излезнеме от командата използваме клавишната комбинация cntr+c

**След като се запознахме как да четем файлове да се запознаем и как да ги намираме**

**Командите в тази секция са find, locate, which, whereis**

Ще ги разгледаме по отделно командите съответно с опциите които вървят с тях

За начало ще започнем с командата find

Смисълът на командата е точно да намери файл или папка в операционанта система

Има малко по дълъг синтаксис ама това се дължи на фактът че се ползва с повече параметри

**find [ folder]  [ parameters] criterion pattern [action]**

или по просто

fine (къде да се търси обикновенно е / ) –name (какво да се търси) file1 (името на това което търсиме) -print (да ни покаже къде се намира )

или в нашият случай 

много често командата find се комбинира с различни други команди което прави използването и много широко списък на всички функции на командата може да видити в [man](http://man7.org/linux/man-pages/man1/find.1.html) страницата или в конзолата каот напишете man find

Следващата команда е locate  която търси файлове по име

Като повечето линукс команди и тя върви със опции или синтаксисът е

**locate** [*OPTION*]... *PATTERN*...

за разлика от find която търси в определени директории locate ползва база данни в която се съдържат Ид номерата на файловете

като за разлика от find може да търси и част от името

по подразбиране командата не е инсталирана в линукс поради фактът че рядко се ползва повече за опциите както и за синтаксиса може да намерите [тук](https://linux.die.net/man/1/locate)

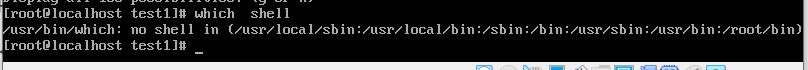
Следващата команда която ще разгледаме е [which](https://linux.die.net/man/1/which)

Командата изкарва пълният път на всички шел команди

Синтаксисът е като на повечето команди

**which** [options] [--] programname [...]

да кажем че искаме да разбереме пълният път на shell



Както виждаме има различни шелове в общият случай командата се ползва без аргументи

Другата полезна команда е whereis която показва пътят до бинарният файл източникът и страницата с ръководството на програмата

Без аргументи показва само къде се намира бинари файлът и източникът .