Commands

- cowsay
- clear - изчиства терминала, като запазва scrollback-а
- reset - изчиства терминала, scrollback-а и всички настройки на терминала
- man < т е м а > дава информация по дадена тема - man-страница (manpage)
- apropos < н и з > - търси в кратките описания на всички man страници
- whatis < н и з > - връща всички секции, в които се среща дадена man страница, и кратки описания на съответните теми
- newgrp < г р у п а > - смяна на активна група в рамките на текущата сесия
- id -u - само UID-то
- whoami отпечатва името на текущия потребител
- who отпечатва всички логнати потребители
- su < п о т р е б и т е л > - изпълнява login shell от името на подадения потребител
- sudo < к о м а н д а > - изпълнява подадената команда от името на r o o t
- pwd - отпечатва текущата директория
- cd < p a t h > - премества текущата директория в дадената
- ls - отпечатва списък със всички обекти в текущата директория
- realpath < п ъ т > - превръща произволен път в абсолютен път
- basename < п ъ т > - връща базовото име на файл (символите след последната наклонена черта)
- dirname < п ъ т > - връща пътя на директорията, в която е файл (символите преди последната наклонена черта)
- touch < п ъ т > - създава празен файл
- mkdir < п ъ т > - създава празна директория
    - - p - създава и всички директории по пътя
- cp < п ъ т - о т - к ъ д е > < п ъ т - к ъ д е > - копира файл
    - - r - копира директория
- mv < п ъ т - о т - к ъ д е > < п ъ т - к ъ д е > - преименува/премества файл
- rm < п ъ т > - изтрива файл
    - rm < п ъ т > - изтрива файл
    - rmdir - изтрива празна директория
- date - без аргументи гледаме текущото време
- d a t e + < format: ' % Y - % m - % d % H : % M : % S ' > - казваме какъв формат
- d a t e - d <format> - може да превежда от един формат във друг
- date %s - сукенди изминали от 1970-01-01 00:00 утц
- stat < ф а й л > и $ ls - l показват информация за атрибутите на файловете
- stat -c ' % s ' < ф а й л > - точен размер на файла в байтове
- stat -c ' % U ‘ - име на потребителя-собственик на файла
- stat -c ' % Y ‘ - момент на последна модификация на файла (UNIX timestamp)
- chown < п о т р е б и т е л > : < г р у п а > < ф а й л > - променя и двете
- chown < п о т р е б и т е л > < ф а й л > - променя само потребителя
- chown : < г р у п а > < ф а й л > - променя само групата / c h g r p , която също променя само групата
- chmod < п р а в а > < ф а й л > - Можете да промените правата за достъп на файл
- chmod u=rwx , g=rx , o= - конкретни права

- chmod a=rw - същото като $ chmod u=rw , g=rw , o=rw
- chmod g+x - добавя право на изпълнение за групата-собственик
- chmod u - w - премахва право на запис за потребителя-собственик
- chmod 755 - конкретни права в осмичен вид
- chmod -R - работи рекурсивно за дадената директория и нейните поддиректории
- umask < м а с к а > можем да зададем множество от права, които се премахват от тези по подразбиране при създаване на нов обект
- umask без аргументи можем да видим текущата маска
- ln < н о в п ъ т > < т е к у щ п ъ т > - създава hardlink
- ln -s < н о в п ъ т > < т е к у щ п ъ т > - създава symlink
- realpath < п ъ т > - превръща произволен път в абсолютен път
- basename < п ъ т > - връща базовото име на файл
    - работи върху пътя като низ, не се интересува от символни връзки
- dirname < п ъ т > - връща пътя на директорията, в която е файл
    - работи върху пътя като низ, не се интересува от символни връзки
- readlink < п ъ т > - дереферира конкретна подадена връзка
- stat -L - дереферира символни връзки
- df < п ъ т > - отпечатва статистика за заето и свободно място на файловата система, в която е подаденият път (или на всички файлови системи, ако не е подаден път)
    - - h , -- human-readable
    - - H , -- si
    - - i , -- inodes
    - - T , –print-type
- du <path> - рекурсивно изписва заетото място на подадената директория и нейните поддиректории
    - -h, —human-readable
    - - -si
    - -s, –summarize - показва само едно число, общото заето място
    - -a - инфо за обикновените файлове, не само за дир
- find < д и р е к т о р и я > изписва на нови редове имената на всички файлове в подадената директория, и в нейните поддиректории
- find [options] [dirs] [filters] [actions]
    - глобални опции, които променят начина, по който открива файлове
    - филтри, които филтрират файловете в подадения път
    - действия, които казват на findкакво да направи с откритите файлове
- find -type
    - f - обиновени файлове
    - d - само директории
    - l - само символни връзки
- find -name - ограничава файловете по име
    - -iname - case insensitive
    - Опциите - name и - iname на командата find могат да приемат globing
- find -maxdepth <num> - ограничаваме се само до файлове на зададените дълбочини от дървото, спрямо подадения път
- find -mindepth <num>
- find -o -> logical OR

- find -a -> logical AND
- find -not
- find -print - ограничаваме се само до файлове на зададените дълбочини от дървото, спрямо подадения път
- find -ls - отпечатва данни за файловете във формат, подобен на l s - l
- find -delete - изтрива всички открити файлове
- find -exec <command> ‘;’ - изпълнява подадената команда за всеки файл, заменяйки низа { } в командата с името на файла - find /dir -type f -exec cat {} ‘;’
- find -print0 - изписва имената разделени с нулев символ вместо символ за нов ред
- find -user
- find
    - -less -> четем файл със скрол
    - -path -> търсим вс файлове които някъде в пътя си имат нещото, case sensitive
    - -ipath -> case insensitive
    - -type {type} - търси определен тип файл
    - -links -> връща брой на хард линкове
    - -exec -> тръгва и изпълнява вс до ';', може да приема колкото си искам арг
    - -{a/c/m}min -> преди колко минути се е случило нещо
    - -{a/c/m}time -> гледа 24 часови периоди, реже fractional parta, закръгля надолу
    - -atime -> за да мачнем +1 файлът трябва да е бил променен преди 2 дни
    - -{a/c}newer -> всеки файл който файнд намери ще гледа change time спрямо modificationprintfx
    - -print
    - -printf -> най-полезната опция, за пищова ни трябва цялата ман страница, не принтира нов ред по подразбиране - за по-сигурно търсене заграждаме в кавички
    - - стакване find -type {} -name {}
    - -type -name -> няма значение как ги подреждаме, разглеждат се като if
    - -perm -> търси права
    -
- xxd < file > можете да прегледате съдържанието на подаден файл байт по байт
- cat < file > - изписва съдържанието на файл в конзолата
- head < file > - изписва първите 10 реда от файл в конзолата
    - -n <N> <file> - първите N реда
    - -n -<N> <file> - всички без последните N реда
- tail <file> - изписва последните 10 реда от файл в конзолата
    - -n <N> <file> - последните N реда
    - -n +<N> <file> - изписва редовете от файла, започвайки от <N>тия
    - -f <file> - изписва последните редове от файла и чака, изписвайки нови редове, добавени след това
- less <file> - pager за файлове (програма, която показва само една страница от текста наведнъж на екрана) - работи по същия начин, както man
- xxd - универсален инструмент, ползвайте го много - вече говорихме за него

- strings < file > - отпечатва всички последователности от символи от файла, които могат да се интерпретират като текст
- file < file > - показва информация за формàта на подадения файл
- tar -c -f test.tar test.txt test2.txt
    - създава файл “архив” test.tar с файловете test.txt и test2.txt
- tar -x -f test.tar
    - разархивира архива test.tar
- tar -cf test.tar test - създава архив test.tar , който съдържа цялата директория test
- tar -v - изписва имената на файловете вътре
- tar -t - кара tar да не прави нищо с архива
- gzip < file > - създава < file >.gz , компресирайки < file > с алгоритъма DEFLATE, и изтрива оригиналния файл
- gzip -d < file >.gz - създава < file > , декомпресирайки и изтривайки < file>.gz
    - -k - кара g z i p да не трие оригиналния файл
- tar - - zstd -cf test.tar.zst test - създава zstd архив test.tar.zst
- tar - -zstd -cf test.tar.gz test - създава zstd архив test.tar.gz (!!лошо)
- tar -caf test.tar.zst test - създава zstd архив
    - -a измисля формата в зависимост от разширението
- tar -xf test.zst - разархивира test.zst в текущата директория
    - -x - работи за произволни (компресирани) архиви, които tar поддържа
- neshto > file
    - пренасочва stdout на neshto във файла file
- neshto 1> file
    - пренасочва stdout на neshto във файла file
- neshto 2> file
    - пренасочва stderr на neshto във файла file
- neshto &> file
    - пренасочва stdout и stderr на neshto във файла file
- neshto < file
    - пренасочва file към stdin на neshto
- neshto 2>&1 - данните, които командата извежда на stderr отиват в stdout и се смесват с данните, които извежда на stdout
- neshto 2>&1 1>test/file - stdout се пренасочва в /test/file а stderr се пренасочва в stdout
- neshto 1>test/file 2>&1 - stderr се пренасочва в stdout, който вече е пренасочен в /test/file : и двете отиват в /test/file
- neshto 3>&1 1>&2 2>&3 - разменяме stdout и stderr
- find /tmp -type f 2>&1 1>/dev/null | head > errors.txt - записваме първите 10 грешки, изведени от find във файл, и игнорираме изведените имена на файлове
- cat <file1> <file2> … - конкатенира съдържанията на подадените файлове
- paste <file1> <file2> … - отпечатва редовете един до друг паралелно, така че образуват колони
    - смяна на разделителя от tab става с -d
- wc - брои
    - -m - байтове
    - -c - символи
    - -l - редове
    - -w - думи

- tr - чете текст от stdin, прави някаква операция със символите в него и извежда резултата на stdout
    - -d <кои символи> - може да трие символи
    - -s - “смачква” поредици от еднакви символи до 1
- cut - разглежда текста ред по ред и отрязва специфични колони
    - -c <n-m> - режи всеки ред от n до m символ
    - -d ‘разделител’ - колоните са с този разделител
    - -f <интервал, нещо такова> - изрежи еди кои си колони/а
- sort - тира редовете в подадения ѝ текст
    - -f - по азбучен ред (default), ignore case
    - -n - numeric sort
    - -h - приеми, че всеки ред започва с число със SI суфикс (k, m, g), и ги сортирай по числата (1M 2M 1G, примерно където M е мегабайта и тн, греда по суфикса)
    - -k <n,m> - сортирай първо по n колонка и после по m
    - -t ‘sep’ - приеми ‘sep’ за разделител на колоните
    - -r - reversed
- uniq - свежда групи от последователни еднакви редове до едно копие на съответния ред (може да се ползва заедно със sort)
    - -c - показва брой срещания на всеки ред, в отделна колонка
- comm <file1> <file2> - извежда 3 колонки, съдържащи:
    - Редовете, които се срещат само в първия файл
    - Редовете, които се срещат само във втория файл
    - Редовете, които се срещат и в двата файла
    - comm работи само върху сортирани файлове
    - Опциите - 1 , - 2 и - 3 на $ comm премахват съответната колонка от изхода
- join - работи само върху сортирани файлове, съединява редовете в двата файла по общи стойности в дадена колонка
- grep <низ> <file> - извежда само редовете от файла, които съдържат подадения низ
    - -n - отпечатва и номера на редовете
    - -B <n> - за всяко съответствие отпечатва n реда преди него и него
    - -A <n> - за всяко съответствие отпечатва него и n реда след него
    - -C <n> - - за всяко съответствие отпечатва n реда преди него, него и n реда след него
    - -i - ignore case, case insensible
    - -v - връща редовете, които НЕ съдържат търсения низ
    - -r <низ> <директория> - търси файлове съдържащи подадения низ, рекурсивно в една директория
    - -F - търси редове БУКВАЛНО съдържащи подадения низ
    - -E ползва extended regex
    - -P използва PCRE
    - -q - не извеждай нищо
- cat file | grep <низ> - когато не е подаден файл като аргумент, grep чете от stdin
- sed (Stream EDitor) - чете текст, променя го по някакво правило, и го извежда
    - ‘s/text1/text2’ - заменя text1 с text2

    - /g - на края на израз за замяна кара замяната да се извърши за всички срещания в рамките на всеки ред, а не само за първото
    - -E - extended regex
    - -i - модифицира файл in-place
- awk -f <script> - чете a w k -скрипт от файл вместо от аргумент
    - -F - позволява да изберем разделител за колонки
    - < т е с т > { < д е й с т в и е > ; < д е й с т в и е > . . . }
    - BEGIN {<action>} - изпълнява се в началото на цялото изпълнение
    - END {<action>} - изпълнява се в края на цялото изпълнение
    - -v = - можем да подадем глобална променлива
    - можем да отпечатваме форматиран изход с printf( ) ,
    - vim < file > - стартира редактора vim върху някакъв файл
- ps - показва информация за дадени процеси в момента на извикване
    - -e - изписва данните за всички процеси
    - -u <user> - изписва процесите на дадения потребител
    - -f - изписва повече информация
    - -o - можем да включим специфични колонки в изхода на ps, можем и да ги именуваме
- top - показва информация за процесите в интерактивен режим
- pstree - рисува дърво с процесите
- kill -<signal> <PID> - праща сигнал на дадения процес
    - -TERM 42 - праща SIGTERM с номер 42
    - -INT - SIGINT на всички bash процеси
- killall - <signal> < name > праща дадения сигнал на всички процеси с даденото име
- jobs - показва всички процеси в текущата сесия и техните job ID-та
- <command> & - пуска команда във фонов режим
    - командата си работи в отделен процес, а през това време можем да продължим да използваме shell-a
- fg <job id> - закача фонов процес към терминала
- bg <job id> - стартира паузиран працес във фонов режим
- which <command> - можем да разберем коя е програмата, асоциирана с дадена команда
- Всеки скрипт ще започва с #!/bin/bash
- alias <name>=<command > - можем да създадем кратко име за често използвани команди
- unalias <name> - трие alias
- set - показва всички променливи
    - -o pipefail - Можем да конфигурираме shell-a така, такава поредица да завършва неуспешно ако която и да е от командите е неуспешна
    - -e - shell-a да прекратява скрипта в момента, в който някоя команда е неуспешна
    - -u - shell-a да прекратява скрипта при реферирane на несъществуваща променлива
- <име на променлива>=<стойност> - задава променлива
    - реферираме я с ${<име на променлива>}
- env - извежда цялата таблица с environment променливи на текущия shell

- export <име на променлива>=<стойност> задава стойност на environment променлива
- env <var1>=<val1> <var2>=<val2> … <command > - може да се използва за да стартираме дадена команда с модифицирани environment променливи
  - -i - игнорира всички съществуващи environment променливи и стартира командата само с тези, подадени директно на env
- echo - изписва аргументите си
  - -n - не изписва символ за нов ред накрая
  - -e - приема интерпретира специални последователности
    - \n
    - \t
- read <име на променлива> - четем данни от stdin
  - -p <prompt> - показва някакъв prompt на потребителя преди да чака за вход
  - -d <symbol> - кара да спира да чете, когато види дадения символ (вместо нов ред) .

1

- Вградената конструкция `$(( <израз> ))` пресмята произволен аритметичен израз и се заменя с резултата
- В израза могат да участват променливи

2

Внимание! Запомнете разликата между пареметри на команда и данни, прочетени от stdin

- Командата `grep` приема **или** име на файл като параметър **или** текст на stdin
- Командата `stat` приема име на файл като параметър и игнорира данните, които получава на stdin
  - всеки път, когато някой се опита да изпълни `find ~ -type f | stat` вместо `find ~ -type f -exec stat {} \;` , умира пингвин
- Командата `cowsay` прави едно и също нещо с параметрите си и с текста, който получи на stdin

3

- Конструкцията `( <команда 1>; <команда 2>; ... <команда n>; )` се нарича *subshell* и много прилича на блок
  - също както блоковете, е начин да третираме поредица от команди като една команда
  - разликата е, че командите се изпълняват в отделен shell от текущия

```
$ ( echo 'The time is:'; date; ) | tr a-z A-Z
THE TIME IS: SUN 12 MAR 17:42:08 EET 2023
```

4

- Конструкцията `{ <команда 1>; <команда 2>; ... <команда n>; }` се нарича *блок* и третира поредицата от команди като една команда

```
$ { echo 'The time is:'; date; } | tr a-z A-Z
THE TIME IS: SUN 12 MAR 17:42:08 EET 2023
```

5

- Можем да комбинираме тези неща:

```
#!/bin/bash
set -euo pipefail

...
```

- По този начин може да сме сигурни, че скриптовете ни ще се прекратяват при грешка
- Недостатъкът е, че ако искаме да проверим дали дадена команда е завършила с неуспех, тя задължително трябва да е част от логически израз или `if`
  - това ще видим как става след малко

- bc - чете израз от stdin и връща резултата на stdout.
- wait <pid> - изчаква процеса с даден pid да завърши и приключва с неговия exit status
- [[<израз>]] - пресмята логически израз

  - Операции за низове
    - `[[ -n <низ> ]]` проверява дали низ е не-празен
    - `[[ -z <низ> ]]` проверява дали низ е празен
  - Операции за файлове
    - `[[ -f <име> ]]` проверява дали обект с такова име съществува и е файл
    - `[[ -d <име> ]]` проверява дали обект с такова име съществува и е директория
    - `[[ -r <име> ]]` проверява дали можем да четем файл
    - `[[ -w <име> ]]` проверява дали можем да пишем файл

- if <command>; then
  <bock of commands >
  else
  <bock of commands >
  fi
- case <string> in
  <glob1>)
  <block>
  ;;
  <glob2>)
  <block>
  ;;
  …
  <globN>)
  <block>
  ;;
  esac
- for <name of var> in <args> ; do
  <block>
  done
- seq <beg> <end> - изписва всички числа в дадения интервал
- while <command>; do
  <block>
  done
- break и continue - могат да се ползват в тялото на цикъл
- function <name> {
  <body>
  }
  local <name>=<value> - локална променлива в рамките на функция
- xargs чете записи от stdin и извиква някаква команда, подавайки ѝ ги като аргументи
- mkfifo <path> - създава виртуален файл, наречен именована тръба
- mktemp - създава временен файл
  - -d - временна директория
- diff - сравнява файлове ред по ред
  - -q - дали са еднакви файловете ще върне es 0
  -