**Лабораторная работа №3**

**Командная оболочка ОС Linux, основные команды, скрипты.**

Зайдите по адресу <https://bellard.org/jslinux/>. Запустите командную оболочку Linux.

К числу основных команд и служебных утилит, используемых при работе в ОС Linux посредством командной оболочки Bash, относятся, приведенные в списке 1:

Список 1:

**cat, cd, clear, cp, date, df, du, echo, exit, grep, head, history, info, less, logout, ls, man, mc, mkdir, mv, nano, passwd, ps, pwd, rm, tail, touch, who**

**Скрипт-файл** – это обычный текстовый файл, содержащий последовательность команд bash, для которого установлены права на выполнение. Пример скрипта, выводящего содержимое текущего каталога на консоль и в файл:

#!/bin/bash

dir

dir > 1.txt

Любой bash-скрипт должен начинаться со строк и:

#!/bin/bash

в этой строке после #! указывается путь к bash-интерпретатору, поэтому если он у вас установлен в другом месте, поменяйте её на ваш путь.

Коментарии начинаются с символа # (кроме первой строки).

Можно создать собственную переменную и присвоить ей значение:

A=121

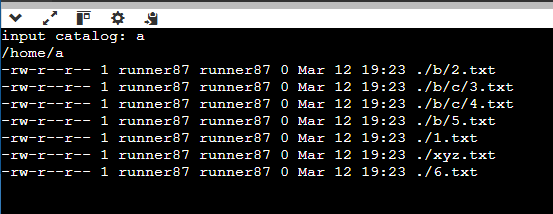
A="121"

let A=121

let "A=А+1"

Вывод значения на консоль:

echo $A



Пример скрипта с переменными:

#!/bin/bash

# указываем где у нас хранится bash-интерпретатор

parametr1=$1

# присваиваем переменной parametr1 значение первого параметра скрипта

script\_name=$0

# присваиваем переменной script\_name значение имени скрипта

echo "Вы запустили скрипт с именем $script\_name и параметром $parametr1"

# команда echo выводит определенную строку, обращение к переменным осуществляется через $имя\_переменной.

echo 'Вы запустили скрипт с именем $script\_name и параметром $parametr1'

# здесь мы видим другие кавычки. Разница в том, что в одинарных кавычках не происходит подстановки переменных.

exit 0

# выход с кодом 0 (удачное завершение работы скрипта)

**Условия**

Структура **if-then-else** используется следующим образом:

if <команда или набор команд, возвращающих код возврата (0 или 1)>

then

<если выражение после if истинно, то выполняется этот блок>

else

<если выражение после if ложно, то этот>

fi

В качестве команд, возвращающих код возврата, могут выступать структуры [[ ]], [ ], test, (( )) или любая другая linux-команда.

test — используется для логического сравнения.

[ ] — синоним команды test

[[ ]] — расширенная версия [ ], внутри которой могут быть использованы || (или), & (и).

(( )) — математическое сравнение.

Для построения многоуровневых условий вида:  
if ... then ... else if ... then ... else ...  
для краткости и читаемости кода, можно использовать структуру  
if ... then ... elif ... then ... elif ....

Если необходимо сравнивать одну переменную с большим количеством возможных значений, то целесообразней использовать оператор case.

#!/bin/bash

echo "Выберите редатор для запуска:"

echo "1. Запуск программы nano"

echo "2. Запуск программы vi"

echo "3. Запуск программы emacs"

echo "4. Выход"

read doing # читаем в переменную $doing со стандартного ввода

case $doing in

1) /usr/bin/nano # если $doing содержит 1, то запустить nano

;;

2) /usr/bin/vi # если $doing содержит 2, то запустить vi

;;

3) /usr/bin/emacs # если $doing содержит 3, то запустить emacs

;;

4) exit 0

;;

\*) #если введено с клавиатуры то, что в case не описывается, выполнять следующее:

echo "Введено неправильное действие"

esac #окончание оператора case.

Условия **сравнения**:

* Файлы:
  + -e - Проверить что файл существует (-f, -d)
  + -f - Файл существует (!-f - не существует)
  + -d - Каталог существует
  + -s - Файл существует и не пустой
  + -r - Файл существует и доступен на чтение
  + -w - Файл существует и доступен на запись
  + -x - Файл существует и доступен на выполнение
  + -h - Символическая ссылка
* Строки:
  + -z - Пустая строка
  + -n - Не пустая строка
  + == - Равно (!= - не равно)
* Числа
  + -eq - Равно
  + -ne - Не равно
  + -lt - Меньше
  + -le - Меньше или равно
  + -gt - Больше
  + -ge - Больше или равно

**Циклы**

Оператор for-in предназначен для поочередного обращения к значениям перечисленным в списке. Каждое значение поочередно в списке присваивается переменной.

Синтаксис оператора следующий:

for переменная in список\_значений do

команды

done

Пример использования **цикла for**:

#!/bin/bash

for i in 0 1 2 3 4

# переменной $i будем поочередно присваивать значения от 0 до 4 включительно

do

echo "Console number is $i" >> /dev/pts/$i

# Пишем в файл /dev/pts/$i (файл виртуального терминала) строку "Console number is $i"

done

#цикл окончен

exit 0

Обход списка файлов в каталоге, используя **цикл for**:

for f in $HOME/tmp/\*; do

filename=$(basename "$f")

extension=${filename##\*.}

if [ "$filename" == "stop.txt" ]; then

break

fi

if [ $extension != 'png' ]; then continue; fi

echo $f

done

**Операции**

Команда let производит арифметические операции над числами и переменными.

Рассмотрим небольшой пример, в котором произведем некоторые вычисления над введенными числами:

#!/bin/bash

echo "Введите a: "

read a

echo "Введите b: "

read b

let "c = a + b" # сложение

echo "a+b= $c"

let "c = a / b" # деление

echo "a/b= $c"

let "c <<= 2" # сдвигает c на 2 разряда влево

echo "c после сдвига на 2 разряда: $c"

let "c = a % b" # находит остаток от деления a на b

echo "$a / $b. остаток: $c "

**Функции**

**Функции** в bash могут принимать аргументы, возвращать вычисленное значение и позволяют исключить дублирование кода в скриптах.

some\_sunction() {

# Объявляем переменную $str локальной и читаем в нее стандартный поток ввода

local str

read str

first\_argument="$1"

second\_argument="$2"

# Читаем построчно входной поток

while read line; do

# Возвращаем список строк для последующей обработки

echo -n "${first\_argument} and $second\_argument"

done <<< file.txt

# Вернуть код завершения (0 - при успешном завершении)

# Код ответа доступен после выполнения ф-ции в переменной $?

return 0

}

# Пример вызова функции

echo 'content' | some\_sunction arg1 arg2

# или так

some\_var='второй аргумент'

result=$(some\_sunction 'arg1' "$some\_var")

# или так

result=`some\_sunction`

ret\_code=$? # получить код возврата функции

**Потоки**

Файл, из которого осуществляется чтение, называется **стандартным потоком ввода**, а в который осуществляется запись — **стандартным потоком вывода**.

Стандартные потоки:

* 0, stdin, ввод;
* 1, stdout, вывод;
* 2, stderr, поток ошибок.

При перенаправлении потоков, вы можете указывать ссылки на определенные потоки. Например, перенаправим вывод и ошибки команды в файл:

command 2>&1 # ошибки (stderr) в stdout

command > ~/out.txt 2>&1 # stdout в файл

command &> ~/out.txt # весь вывод в файл

Для перенаправления потоков используются основные команды: <, >, >>, <<<, |. Рассмотрим, как можно перенаправлять стандартные потоки.

Перенаправление потока вывода:

* > - перенаправить поток вывода в файл (файл будет создан или перезаписан)
* >> - дописать поток вывода в конец файла

Перенаправление потока ввода (прием данных):

* < - файл в поток ввода (файл будет источником данных)
* <<< - чтение данных из строки вместо содержимого файла (для bash 3 и выше)

Перенаправление вывода ошибок:

* 2> - перенаправить поток ошибок в файл
* 2>> - дописать ошибки в файл (файл будет создан или перезаписан)

**Конвейеры**

**Конвейер** — очень мощный инструмент для работы с консолью Bash. Синтаксис команда1 | команда 2 — означает, что вывод команды 1 передастся на ввод команде 2.

Конвейеры можно группировать в цепочки и выводить с помощью перенаправления в файл, например:

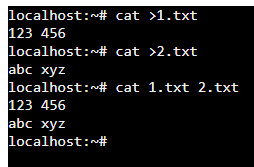
ls -la | grep "hash" | sort > sortilg\_list.txt

Вывод команды ls -la передается команде grep, которая отбирает все строки, в которых встретится слово hash, и передает их команде сортировке sort, которая пишет результат в файл sorting\_list.txt.

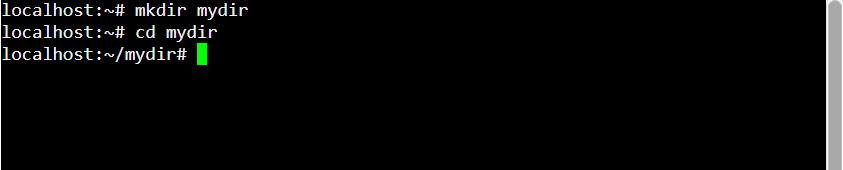
**Задания 1.**

Для каждой из команд, приведенных в списке 1 выяснить её назначение и привести пример (скриншот) работы команды.

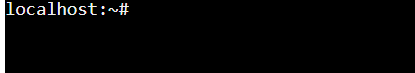
**Cat или дословно concatenate – утилита в Linux подобных системах, позволяющая объединить потоки ввода и вывода направленные в утилиту.**

****

**Cd – изменение текущего каталога**

****

**Clear – очистка консоли**

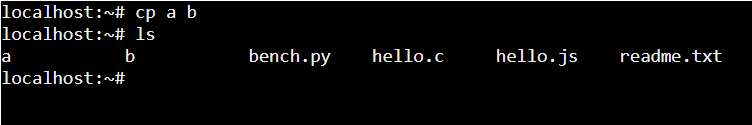
****

**Cp – копирование файла**

Команда cp копирует файл или каталог, указанный в параметре Исходный\_файл или Исходный\_каталог, в файл или каталог, заданный в параметре Целевой\_файл или Целевой\_каталог.

Если Целевой\_файл существует, то его содержимое заменяется без предупреждения. При копировании нескольких Исходных\_файлов должен быть указан целевой каталог.

Если файл с указанным именем уже существует в целевом каталоге, то его содержимое будет заменено на содержимое исходного файла. В связи с этим рекомендуется указывать новое имя для копии файла, чтобы в целевом каталоге не было файла с тем же именем.

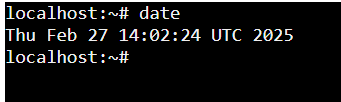
****

**Date**

Команда **date** позволяет задействовать одноименную утилиту, которая используется для двух целей. Обычно она используется для получения текущих даты и времени при необходимости в заданном формате, но при этом она также позволяет устанавливать их.

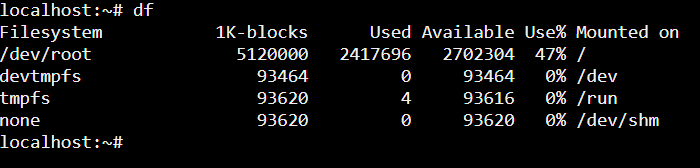
Стандартный синтаксис команды выглядит следующим образом:

$ date [описание времени или формата вывода]

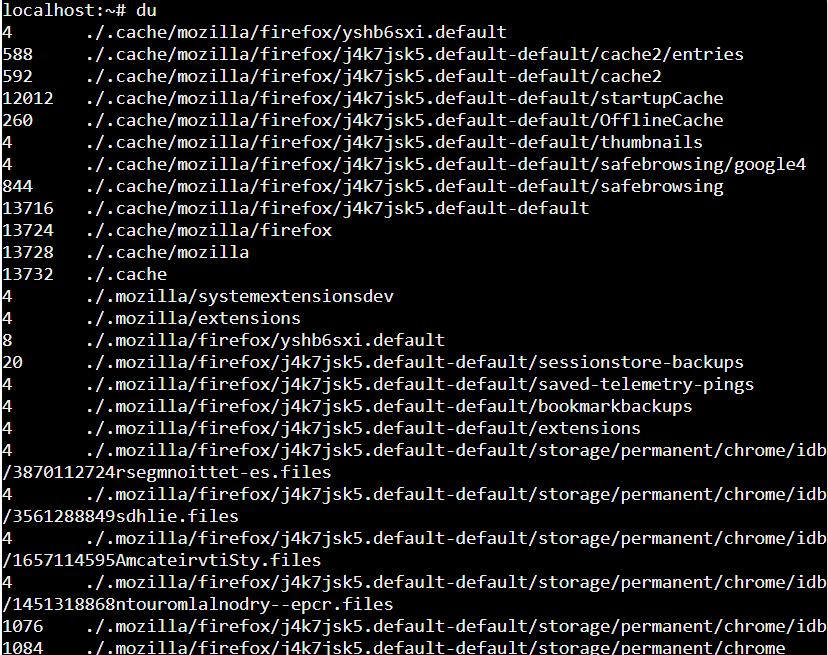
****

**Df**

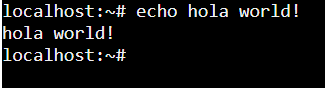
Утилита df (от англ. **disk free**) — инструмент командной строки **Linux**, который позволяет **проверить использование дискового пространства в операционной системе**. Эта команда отображает информацию об **общем, свободном и используемом объеме памяти файловых систем**



**du - оценка места на диске, занимаемого файлом**

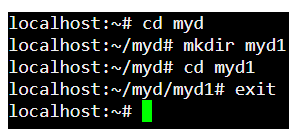
****

**Echo – вывод на консоль**

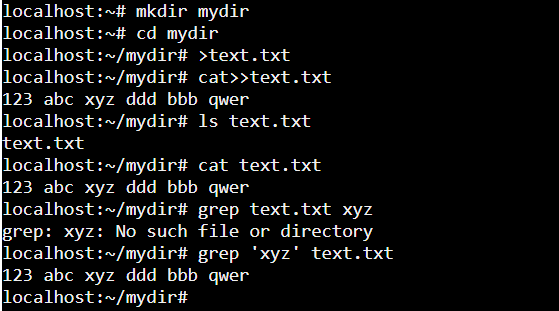
****

**Exit**

Команда **exit** является встроенной командой, предназначенной для завершения процесса командной оболочки с кодом успешного завершения или кодом ошибки, если такой был передан в качестве аргумента.

****

Grep это утилита командной строки Linux, который даёт пользователям возможность вести поиск строки. С его помощью можно даже искать конкретные слова в файле. Также можно передать вывод любой команды в grep, что сильно упрощает работу во время поиска и траблшутинга.

****

**Head - выводит начальные строки (по умолчанию — 10) из одного или нескольких документов. Также она может показывать данные, которые передает на вывод другая утилита.**

**Синтаксис у команды head следующий:**

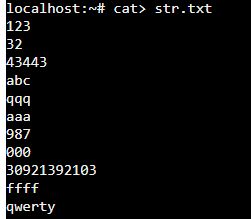
**$ head опции файл**

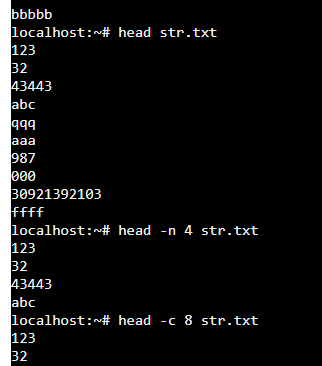
**Здесь:**

* **Опции — это параметр, который позволяет настраивать работу команды таким образом, чтобы результат соответствовал конкретным потребностям пользователя.**
* **Файл — это имя документа (или имена документов, если их несколько). Если это значение не задано либо вместо него стоит знак «-», команда будет брать данные из стандартного вывода.**

**Чаще всего к команде head применяются такие опции:**

* **-c (--bytes) — позволяет задавать количество текста не в строках, а в байтах. При записи в виде --bytes=[-]NUM выводит на экран все содержимое файла, кроме NUM байт, расположенных в конце документа.**
* **-n (--lines) — показывает заданное количество строк вместо 10, которые выводятся по умолчанию. Если записать эту опцию в виде --lines=[-]NUM, будет показан весь текст кроме последних NUM строк.**
* **-q (--quiet, --silent) — выводит только текст, не добавляя к нему название файла.**
* **-v (--verbose) — перед текстом выводит название файла.**
* **-z (--zero-terminated) — символы перехода на новую строку заменяет символами завершения строк.**

****

****

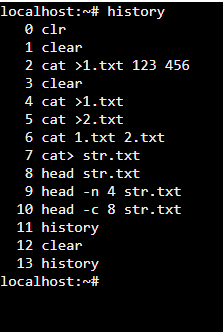
**History – вывод истории последних используемых команд (по умолчанию 1000)**

**$ history опции файл**

**В качестве файла можно указать файл истории. По умолчанию история для текущего пользователя хранится в файле ~/.history, но вы можете задать, например, файл другого пользователя.**

**Опции:**

* **-c - очистить историю;**
* **-d - удалить определенную строку из истории;**
* **-a - добавить новую команду в историю;**
* **-n - скопировать команды из файла истории в текущий список;**
* **-w - перезаписать содержимое одного файла истории в другой, заменяя повторяющиеся вхождения.**

****

**Info**

**Команда info отображает информацию в формате документа. Она похожа на команду man, но более надёжна для связывания страниц.**

**Информационные страницы создаются с помощью инструментов texinfo, могут связываться с другими страницами и создавать меню.**

**Содержание**

* [**Синтаксис**](https://linuxcookbook.ru/articles/komanda-info#1)

**Синтаксис**

**$ info [опции] [файл]**

**Опции**

**Часто используемые параметры:**

**-a, --a**

**lИспользовать все подходящие справочники**

**-k, --apropos=СТРОКА**

**Поиск СТРОКИ во всех указателях всех справочников**

**-d, --directory=КАТ**

**Добавить КАТалог в INFOPATH**

**-f, --file=СПРАВОЧНИК**

**Задать открываемый Info-справочник**

**-h, --help**

**Показать эту справку и выйти**

**--index-search=СТРОКА**

**Перейти к ноде, указанной в пункте СТРОКА  в указателе**

**-n, --node=ИМЯ\_НОДЫ**

**Задать ноды в файле Info, посещаемом впервые**

**-o, --output=ФАЙЛ**

**Запись выбранных нод в ФАЙЛ**

**--subnodes**

**Рекурсивно выводить пункты меню**

**-v, --variable ПЕР=ЗНАЧ**

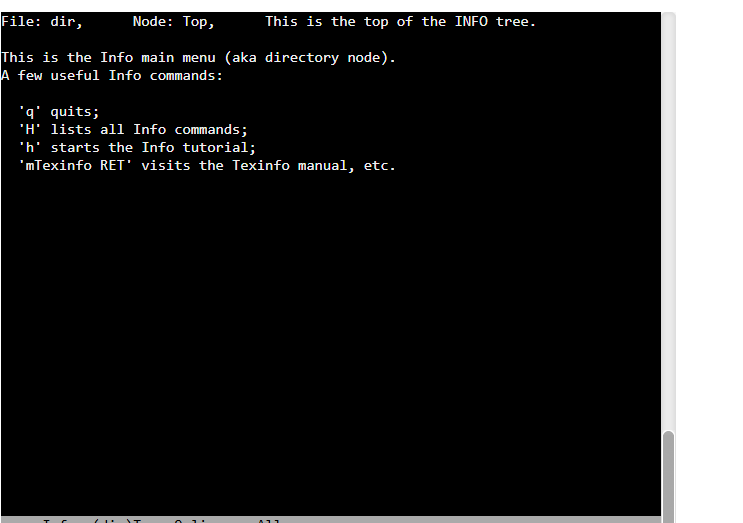
**Назначить ЗНАЧЕНИЕ ПЕРЕМЕННОЙ Info**

**--version**

**Показать информацию о версии и выйти**

**-w, --where, --location**

**Показать физическое расположение файла Info**

****

**Less**

**позволяет перематывать текст не только вперёд, но и назад, осуществлять поиск в обоих направлениях, переходить сразу в конец или в начало файла.**

**Синтаксис**

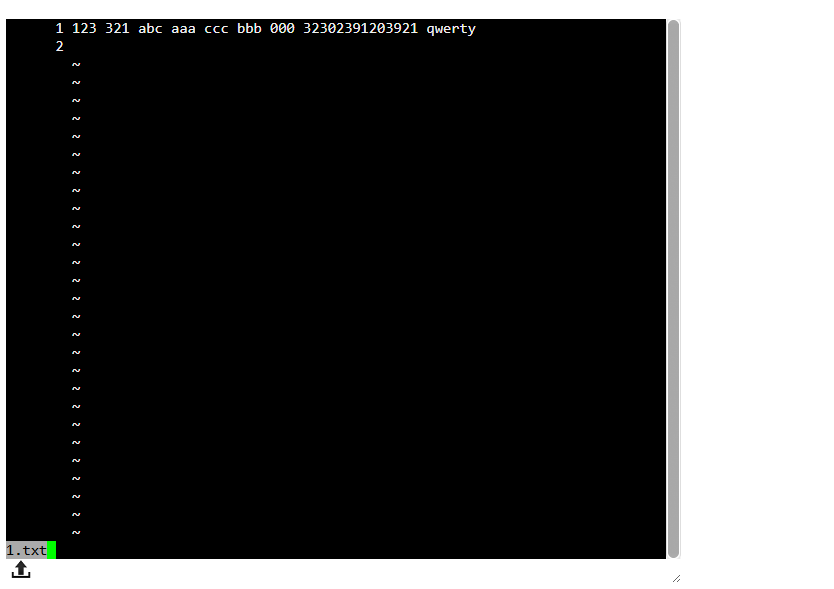
**команда опции файл**

**Наиболее популярные опции:**

* **-a, --search-skip-screen — не осуществлять поиск в тексте, который в данный момент отображен на экране;**
* **-bn, --buffers=n — задать размер буфера памяти;**
* **-c, --clear-screen — листать текст, полностью стирая содержимое экрана (построчная прокрутка работать не будет);**
* **-Dxcolor, --color=xcolor — задать цвет отображаемого текста;**
* **-E, --QUIT-AT-EOF — выйти, когда утилита достигнет конца файла;**
* **-e, --quit-at-eof — выйти, когда утилита второй раз достигнет конца файла;**
* **-F, --quit-if-one-screen — выйти, если содержимое файла помещается на одном экране;**
* **-f, --force — открыть специальный файл;**
* **-hn, --max-back-scroll=n — задать максимальное количество строк для прокрутки назад;**
* **-yn, --max-forw-scroll=n — задать максимальное количество строк для прокрутки вперёд;**
* **-i, --ignore-case — игнорировать регистр;**
* **-I, --IGNORE-CASE — игнорировать регистр, даже если паттерн для поиска содержит заглавные буквы;**
* **-jn, --jump-target=n — указать, в какой строке должна быть выведена искомая информация;**
* **-J, --status-column — пометить строки, соответствующие результатам поиска;**
* **-n, --line-numbers — не выводить номера строк;**
* **-N, --LINE-NUMBERS — вывести номера строк;**
* **-s, --squeeze-blank-lines — заменить множество идущих подряд пустых строк одной пустой строкой;**
* **-w, --hilite-unread — выделить первую строку нового фрагмента текста.**

**Во время просмотра текста утилитой можно управлять при помощи внутренних команд, набирая их на клавиатуре компьютера. Наиболее часто используемые из них:**

* **h, H — справка;**
* **Space, Ctrl+V, f, Ctrl+F — прокрутить текст на один экран вперёд;**
* **Enter, Return, Ctrl+N, e, Ctrl+E, j, Ctrl+J — прокрутить текст на n строк вперед, по умолчанию n=1;**
* **y, Ctrl+Y, Ctrl+P, k, Ctrl+K — прокрутить текст на n строк назад, по умолчанию n=1;**
* **Ctrl+→ — прокрутить текст по горизонтали в конец строки;**
* **Ctrl+← — прокрутить текст по горизонтали в начало строки;**
* **:d — удалить текущий файл из списка файлов;**
* **Ctrl+G, :f — вывести основную информацию о файле;**
* **q, Q, :q, :Q, ZZ — выход.**

****

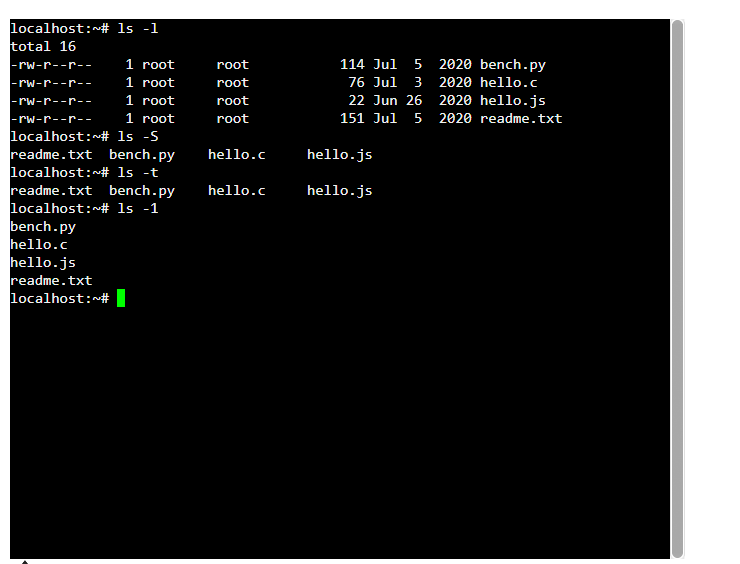
**Logout – выход из системы**

**Ls – просмотр содержимого каталога**

**$ ls опции /путь/к/папке**

**Опции команды указывают как именно и в каком виде нужно выводить информацию на экран, а путь, это та папка, которую нужно отобразить. Рассмотрим основные опции утилиты:**

* **-a - отображать все файлы, включая скрытые, это те, перед именем которых стоит точка;**
* **-A - не отображать ссылку на текущую папку и корневую папку . и ..;**
* **--author - выводить создателя файла в режиме подробного списка;**
* **-b - выводить Escape последовательности вместо непечатаемых символов;**
* **--block-size - выводить размер каталога или файла в определенной единице измерения, например, мегабайтах, гигабайтах или килобайтах;**
* **-B - не выводить резервные копии, их имена начинаются с ~;**
* **-c - сортировать файлы по времени модификации или создания, сначала будут выведены новые файлы;**
* **-C - выводить колонками;**
* **--color - включить цветной режим вывода, автоматически активирована во многих дистрибутивах;**
* **-d - выводить только директории, без их содержимого, полезно при рекурсивном выводе;**
* **-D - использовать режим вывода, совместимый с Emacs;**
* **-f - не сортировать;**
* **-F - показывать тип объекта, к каждому объекту будет добавлен один из специализированных символов \*/=>@|;**
* **--full-time - показывать подробную информацию, плюс вся информация о времени в формате ISO;**
* **-g - показывать подробную информацию, но кроме владельца файла;**
* **--group-directories-first - сначала отображать директории, а уже потом файлы;**
* **-G - не выводить имена групп;**
* **-h - выводить размеры папок в удобном для чтения формате;**
* **-H - открывать символические ссылки при рекурсивном использовании;**
* **--hide - не отображать файлы, которые начинаются с указанного символа;**
* **-i - отображать номер индекса inode, в которой хранится этот файл;**
* **-l - выводить подробный список, в котором будет отображаться владелец, группа, дата создания, размер и другие параметры;**
* **-L - для символических ссылок отображать информацию о файле, на который они ссылаются;**
* **-m - разделять элементы списка запятой;**
* **-n - выводить UID и GID вместо имени и группы пользователя;**
* **-N - выводить имена как есть, не обрабатывать контролирующие последовательности;**
* **-Q - брать имена папок и файлов в кавычки;**
* **-r - обратный порядок сортировки;**
* **-R - рекурсивно отображать содержимое поддиректорий;**
* **-s - выводить размер файла в блоках;**
* **-S - сортировать по размеру, сначала большие;**
* **-t - сортировать по времени последней модификации;**
* **-u - сортировать по времени последнего доступа;**
* **-U - не сортировать;**
* **-X - сортировать по алфавиту;**
* **-Z - отображать информацию о расширениях SELinux;**
* **-1 - отображать один файл на одну строку.**

****

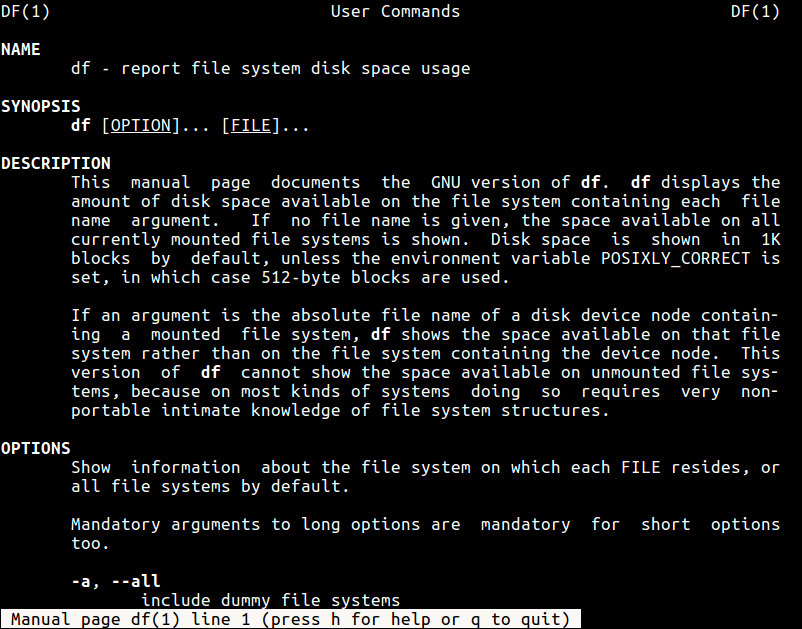
**Man**

**Команда man позволяет получить доступ к общей базе справки по команде, функции или программе. Обычно для просмотра справки программе надо передать название команды или другого объекта в системе. Синтаксис у неё такой:**

**$ man раздел название\_страницы**

**Обычно название страницы совпадает с именем команды или названием программы. Вся справка разделена на несколько разделов. По умолчанию страницы справки выводятся из всех разделов, но вы можете выводить страницы справки из определённого раздела. Для этого достаточно передать номер раздела команде. Вот список основных разделов:**

* **1 - команды оболочки или программы;**
* **2 - системные вызовы;**
* **3 - функции библиотек;**
* **4 - файлы из каталога /dev;**
* **5 - форматы файлов;**
* **6 - игры;**
* **7 - разное;**
* **8 - команды администрирования системы;**
* **9 - процедуры ядра.**



**Mc**

**"GNU Midnight Commander - это визуальный файловый менеджер, распространяемый под лицензией GPL, и поэтому он квалифицируется как Free Software. Это мощная текстовая полнофункциональная программа, которая позволяет вам копировать, перемещать и удалять файлы и директории, производить поиск файлов и запускать на выполнение команды оболочки. Также включены встроенные редактор и программа для просмотра файлов".**

**Как установить Midnight Commander в Linux**

**По умолчанию MC, как правило, не устанавливается. В Debian, Ubuntu и Linux Mint вы можете установить его с помощью команды apt-get:**

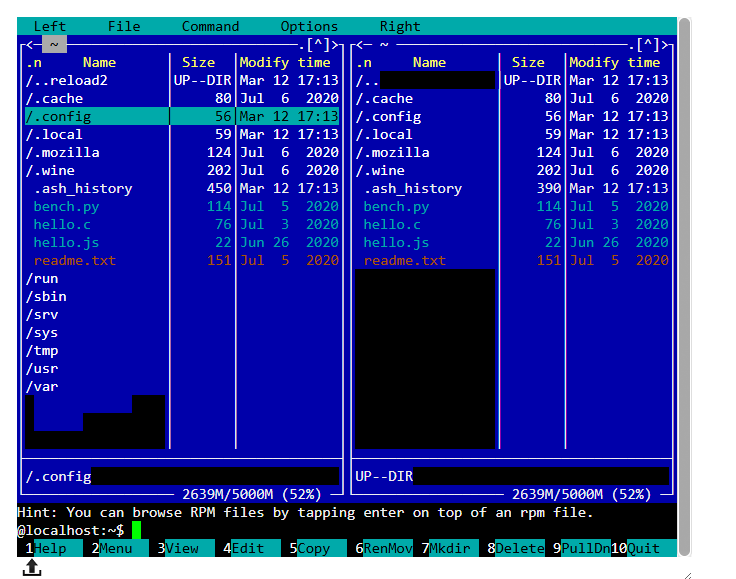
**$ sudo apt-get install mc**

**В RHEL, CentOS и Fedora вы можете использовать команду:**

**# yum install mc**

**После завершения инсталляции просто введите "mc" (без кавычек) в консоли для его запуска.**

**# mc**

****

**Mkdir – создание новой директории**

**Синтаксис**

**mkdir опции директория**

**директория — одна или несколько директорий через пробел, которые требуется создать.**

**Опции**

**-m=MODE или --mode=MODE**

**Устанавливает права доступа для создаваемой директории. Синтаксис MODE такой же как у команды chmod**

**-p или --parents**

**Создать все директории, которые указаны внутри пути. Если какая-либо директория существует, то предупреждение об этом не выводится.**

**-v или --verbose**

**Выводить сообщение о каждой создаваемой директории.**

**-Z**

**Установить контекст SELinux для создаваемой директории по умолчанию.**

**--context[=CTX]**

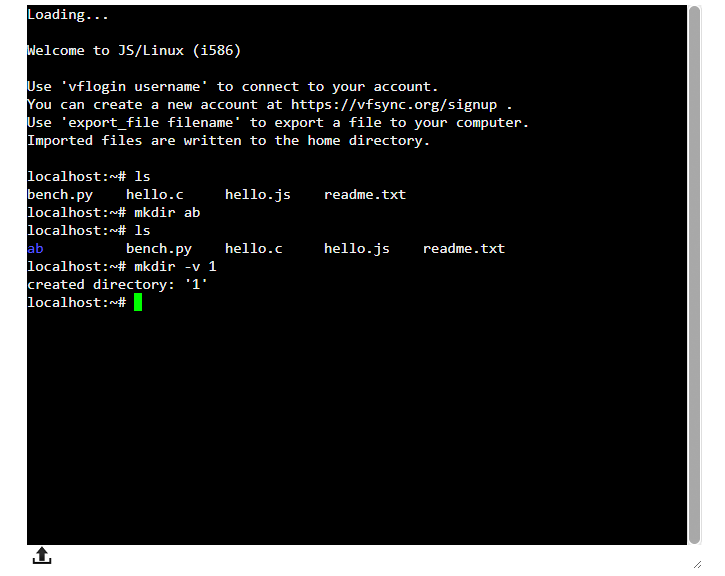
**Установить контекст SELinux для создаваемой директории в значение CTX**

**--help**

**Показать справку по команде mkdir**

**--version**

**Показать версию утилиты mkdir**

****

**Mv**

**Команда mv используется для перемещения одного или нескольких файлов (или директорий) в другую директорию, а также для переименования файлов и директорий.**

**Синтаксис**

**mv опции исходные\_файлы куда**

**исходные\_файлы — это то, что мы будем перемещать или переименовывать. То есть это либо один файл, либо несколько файлов, либо директория, либо несколько директорий.**

**куда — это название директории, в которую мы перемещаем исходные файлы или директории. Или же это новое название файла или директории, если выполняется переименование. То есть это либо директория, либо новое название файла или директории.**

**Команда mv может:**

* **Переместить один или несколько файлов в другую директорию.**
* **Переместить одну или несколько директорий в другую директорию.**
* **Переименовать файл.**
* **Переименовать директорию.**

**Опции**

**-b или --backup или --backup=МЕТОД**

**Создать резервную копию существующих файлов, которые будут перезаписаны или удалены.  
Имя для резервной копии может создаваться по определенным правилам. Для этого используется опция --backup=МЕТОД В качестве значения МЕТОД можно указать:**

* **none или off — не делать резервных копий, даже если включена опция --backup**
* **numbered или t — имя резервной копии получит числовой индекс (пример: myfile.txt~2~).**
* **existing или nil — если в директории уже есть резервные копии с числовыми индексами, то использовать числовые индексы для новых резервных копий, во всех остальных случаях использовать метод simple.**
* **simple или never — делать обычные резервные копии (пример: myfile.txt~).**

**-f или --force**

**При перезаписи существующих файлов не спрашивать разрешения пользователя.**

**-i или --interactive**

**При перезаписи существующих файлов спрашивать разрешение пользователя на подтверждение операции.**

**-n или --no-clobber**

**Не перезаписывать существующие файлы.**

**--strip-trailing-slashes**

**Удалить завершающий символ слеш / (если он там есть) из имени файла-источника.**

**-S СУФФИКС или --suffix=СУФФИКС**

**Изменить символ суффикса, который добавляется к именам резервных копий (при использовании опции --backup). По умолчанию СУФФИКС равен значку тильды ~**

**-t ДИРЕКТОРИЯ или --target-directory=ДИРЕКТОРИЯ**

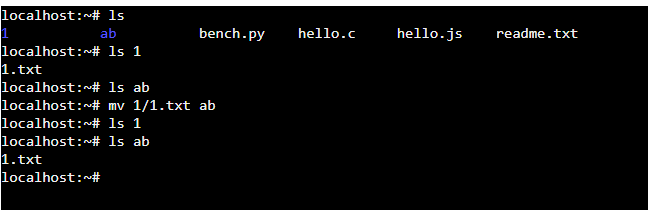
**Переместить все исходные файлы в директорию, которая указана в аргументе опции.**

**-u или --update**

**Перемещать только если исходный файл новее, чем файл назначения или если файл-назначения отсутствует.**

**-v или --verbose**

**Выводить информацию о каждом файле, который обрабатывает команда *mv*.**

****

**Nano – консольный текстовый редактор**

**синтаксис:**

**$ nano опции /адрес/файла/файл**

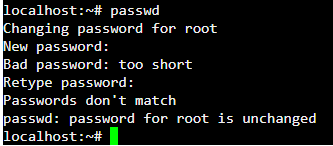
**Использовать опции командной строки не обязательно достаточно только указать имя файла, но в некоторых случаях опции могут быть очень полезны. Рассмотрим их подробнее:**

* **+номер - указать строку в файле, с которой нужно начинать редактирование;**
* **-B - создавать резервную копию для файла при сохранении;**
* **-С - папка для резервной копии;**
* **-D - выводить текст жирным;**
* **-E - конвертировать табуляции в пробелы;**
* **-F - включить поддержку редактирования нескольких файлов;**
* **-H - сохранять историю поиска и замены;**
* **-I - не читать конфигурацию из файлов nanorc;**
* **-P - запоминать и восстанавливать позицию курсора во время последнего редактирования;**
* **-T - размер табуляции в пробелах;**
* **-m - включить поддержку мыши;**
* **-v - режим только для чтения, вы не можете сохранить файл.**

****

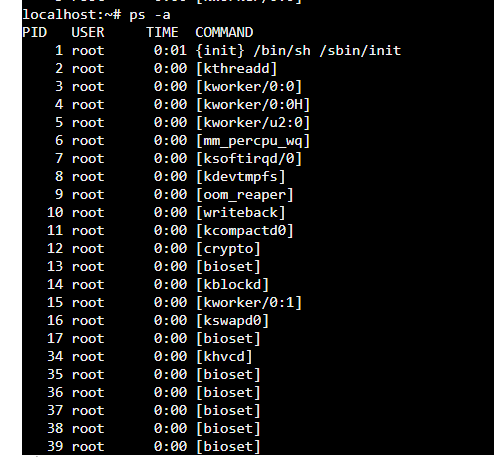
**Passwd – изменение паролей учётных записей**

|  |  |
| --- | --- |
| **-d,- -delete** | **Удалить пароль для указанной записи.** |
| **-f,- -force** | **Форсировать операцию.** |
| **-k,- -keep-tokens** | **Сохранить не устаревшие пароли.** |
| **-l,- -lock** | **Блокировать указанную запись.** |
| **--stdin** | **Прочитать новые пароли из стандартного ввода.** |
| **-S,- -status** | **Дать отчет о статусе пароля в указанной записи.** |
| **-u,- -unlock** | **Разблокировать указанную запись.** |
| **-?,- -help** | **Показать справку и выйти.** |
| **--usage** | **Дать короткую справку по использованию.** |
| **-V,- -version** | **Показать версию программы и выйти.** |

****

**Ps – отображение списка активных процессов**

* **-A, -e, (a) - выбрать все процессы;**
* **-a - выбрать все процессы, кроме фоновых;**
* **-d, (g) - выбрать все процессы, даже фоновые, кроме процессов сессий;**
* **-N - выбрать все процессы кроме указанных;**
* **-С - выбирать процессы по имени команды;**
* **-G - выбрать процессы по ID группы;**
* **-p, (p) - выбрать процессы PID;**
* **--ppid - выбрать процессы по PID родительского процесса;**
* **-s - выбрать процессы по ID сессии;**
* **-t, (t) - выбрать процессы по tty;**
* **-u, (U) - выбрать процессы пользователя.**

****

**Pwd – вывод в терминал путь к текущей папке**

* **-L, --logical - брать директорию из переменной окружения, даже если она содержит символические ссылки;**
* **-P - отбрасывать все символические ссылки;**
* **--help - отобразить справку по утилите;**
* **--version - отобразить версию утилиты.**

****

**Rm – удаление файлов и деректорий**

**-f, --force**

**Игнорировать несуществующие файлы и аргументы, ни о чём не спрашивать**

**-i**

**Запрашивать подтверждение перед каждым удалением**

**-I**

**Запрашивать подтверждение один раз перед удалением более трёх файлов или перед рекурсивным удалением; не так назойливо, как с -i, но всё же защищает от большинства ошибок**

**--interactive[=КОГДА]**

**Запрашивать подтверждение в соответствии с КОГДА: never (никогда), once (-I, один раз) или always (-i, всегда); если КОГДА не задано — запрашивать всегда**

**--one-file-system**

**При рекурсивном удалении дерева, пропускать все каталоги, находящиеся не на той же файловой системе, что и соответствующий аргумент командной строки**

**--no-preserve-root**

**Не обрабатывать / особым образом**

**--preserve-root[=all]**

**Не удалять / (по умолчанию); при указании all отвергать любой аргумент командной строки на отдельном устройстве от его родителя**

**-r, -R, --recursive**

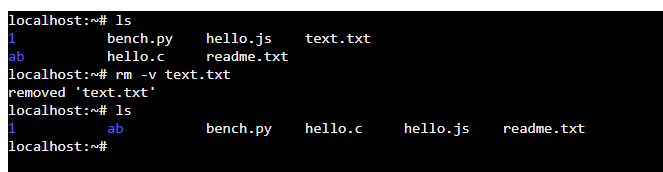
**Рекурсивно удалять каталоги и их содержимое**

**-d, --dir**

**Удалять пустые каталоги**

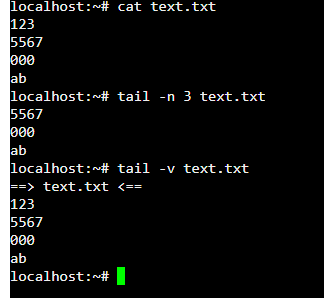
**-v, --verbose**

**Пояснять производимые действия**

****

**Tail -  позволяет выводить заданное количество строк с конца файла, а также выводить новые строки в интерактивном режиме.**

* **-c - выводить указанное количество байт с конца файла;**
* **-f - обновлять информацию по мере появления новых строк в файле;**
* **-n - выводить указанное количество строк из конца файла;**
* **--pid - используется с опцией -f, позволяет завершить работу утилиты, когда завершится указанный процесс;**
* **-q - не выводить имена файлов;**
* **--retry - повторять попытки открыть файл, если он недоступен;**
* **-v - выводить подробную информацию о файле;**

****

**Touch**

**Утилита touch устанавливает время доступа и время модификации указанных файлов. Если файл не существует, он создается.**

**Опции:**

**-a**

**Изменять время доступа к файлу. Не изменять время модификации, если одновременно не указана опция -m.**

**-c**

**Не создавать указанный файл, если он не существует. Не выдавать никаких диагностических сообщений в этом случае.**

**-m**

**Изменять время модификации файла. Не изменять время доступа, если одновременно не указана опция -a.**

**-r базовый\_файл**

**Использовать соответствующие атрибуты времени базового\_файла, а не текущее время.**

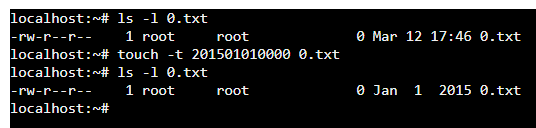
**-t время**

**Использовать вместо текущего указанное время. Время задается десятичным числов вида:**

**[[CC]YY]MMDDhhmm[.SS]**

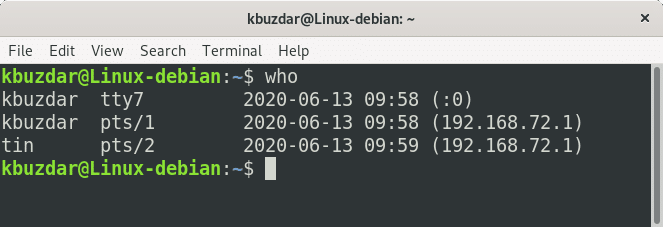
**где каждая пара цифр представляет следующее:**

|  |  |
| --- | --- |
| **MM** | **Месяц года [01-12].** |
| **DD** | **День месяца [01-31].** |
| **hh** | **Час дня [00-23].** |
| **mm** | **Минута часа [00-59].** |
| **CC** | **Первые две цифры года.** |
| **YY** | **Последние две цифры года.** |
| **SS** | **Секунда минуты [00-61].** |



**who**

* **-a (--all)** — включает в себя все основные опции.
* **- b (--boot)** — показывает время загрузки операционной системы.
* **-d (--dead)** — выводит перечень зомби-процессов.
* **- H (--heading)** — никак не влияет на получаемую информацию, зато добавляет колонкам заголовки и помогает понять что где находится.
* **-m** — показать пользователя, который сейчас работает в терминале.
* **-r —**вывести текущий уровень запуска (runinit);
* **-t —**показать последнее изменение системных часов;
* **-s** **—**вывести только имя, терминальную сессию и время.
* **-q** **—**вывести количество авторизованных пользователей.
* **-T —**данные о терминальной сессии.
* **-u —**показать активных пользователей.
* **--ips** — вместо названия хостов показывает ips.
* **--lookup** — используется в сочетании с --ips, выводит данные, которые основываются на сохраненном IP, если он доступен, а не на названии хоста.



**Задание 2.**

* 1. Написать скрипт с использованием цикла for, выводящий на консоль размеры и права доступа для всех файлов в заданном каталоге и всех его подкаталогах (имя каталога задается пользователем в качестве первого аргумента командной строки).

#!/bin/bash

mkdir a

mkdir a/b

> a/1.txt

> a/b/2.txt

> a/xyz.txt

mkdir a/b/c

> a/b/c/3.txt

> a/b/c/4.txt

> a/b/5.txt

> a/6.txt

read -p "input catalog: " dir

cd $dir

pwd

#arr = ($(find .-type f))

arr=($(find . -type f))

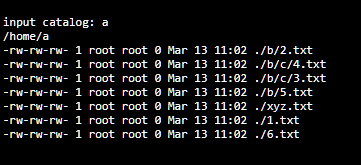
index=0

length=${#arr[@]}

for element in "${arr[@]}"; do

ls -lh "$element"

done

****