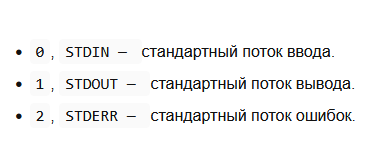
**ПЕРЕНАПРАВЛЕНИЕ ВВОДА-ВЫВОДА**

**Command > file.txt –** перенаправление стандартного потока вывода в файл. Если файл существует, то он будет перезаписан.

**Command >> file.txt –** перенаправление стандартного потока вывода, но данные дописываются в конец файла, если он существует.

**Command 2> errors.txt –** перенаправить стандартный поток ошибок в файл.

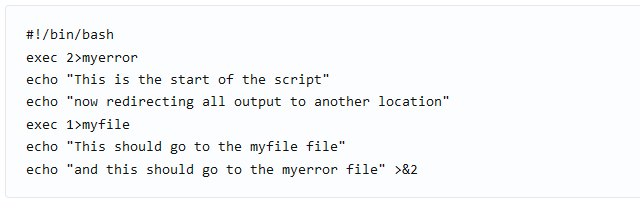
**Command &> file.txt –** перенаправить в файл и поток вывода, и поток ошибок.

**Command < file.txt –** перенаправление ввода из файла.

**Command1 | Command2 –** перенаправить вывод одной команды на вход другой.

Можно иметь до 9 открытых дескрипторов. Дескрипторы с номерами 3-8 также можно использовать для перенаправления ввода вывода.

Внутри скриптов перенаправить дескриптор можно командой **exec**.



**ОСНОВНЫЕ КОМАНДЫ**

**cd –** переход в другую директорию.

* *cd –* перемещение в домашний каталог (расположение домашнего каталога берется из переменной среды)
* *cd ~ -* переход в домашний каталог
* *cd -* - вернуться в предыдущий каталог
* *cd .*. – перейти в родительский каталог
* *cd / -* перейти в корневой каталог
* *cd folder –* перейти в каталог, относительно текущего расположения (относительный путь).
* *Cd /folder –* перейти в каталог относительно корневой директории (абсолютный путь).

**pushd и popd -** Можно сказать, что простая реализация стека для рабочих каталогов. Когда вы выполняете pushd, текущий рабочий каталог сохраняется в памяти, а на его место устанавливается указанный. popd, чтобы вернуться в предыдущий каталог.

**ls –** вывод списка файлов в каталоге.

* **-a** - отображать все файлы, включая скрытые, это те, перед именем которых стоит точка;
* **--author** - выводить создателя файла в режиме подробного списка;
* **-d** - выводить только директории, без их содержимого, полезно при рекурсивном выводе;
* **-l** - выводить подробный список, в котором будет отображаться владелец, группа, дата создания, размер и другие параметры;
* **-r** - обратный порядок сортировки;
* **-R** - рекурсивно отображать содержимое поддиректорий;
* **-1**- отображать один файл на одну строку.
* -**t –** сортировка по времени изменения. Сначала идут более новые.
* **-X –** сортировка по алфавиту.
* -**S –** сортировка по размеру документов. Сначала идут файлы и каталоги большего размера.
* -**hide -** Скрыть документы, которые полностью совпадают с указанными символами. Например нужно скрыть “kurica”, указываем –hide kurica.
* **-I** ‘шаблон’. Скрыть файлы согласно шаблону. Например нужно скрыть все документы, которые начинаются на букву f. Тогда пишем ls -I’f\*’.

**nano –** консольный текстовый редактор для unix-подобных ОС.

**man command –** справочная информация и руководство по команде.

**chown –** команда изменяет владельца файла или файлов.

* **chown USER FILE** - изменить владельца
* **chown USER:GROUP FILE** - изменить владельца и группу
* **chown :GROUP FILE** - изменить группу файла
* **chown -R USER:GROUP DIRECTORY** - рекурсивное изменение владельца всех файлов в директории

**sort –** сортирует и выводит содержимое входного потока. (**-r** для сортировки по убыванию)

**cat –** вывод содержимого файла (файлов).

**tac –** вывод содержимого фалов в обратном порядке, от последней строки к первой.

**uniq** - Удаляет повторяющиеся строки из отсортированного файла. Ключ **-c** выводит кол-во повторяющихся строк

**expand** – преобразовать символы табуляции в пробелы

**unexpand** - пробелы в табы

**cut** - Предназначена для извлечения отдельных полей из текстовых файлов. Синтаксис: **cut опции путь\_к\_файлу.**

* **-d** позволяет установить свой разделитель вместо стандартного таб
* **-f** передача полей для вырезания (номера полей)
* **-b** номер байта, набор или диапазон байтов, подлежащих вырезанию.

**head** - выводит первые строки (по умолчанию 10 строк)

* **-n 1** - позволяет указать количество строк
* **-v** - перед текстом выводит название файла
* **-q**  - выводит только текст, не добавляя к нему название файла.
* **-c** - позволяет задавать количество текста не в строках, а в байтах.

**tail** - выводит последние строки (по умолчанию 10 строк)

* **-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;

**more –** постраничный вывод файла.

**grep** - Многоцелевая поисковая утилита, использующая регулярные выражения. Синтаксис: **grep pattern [file...]**

* **-i** - выполняется поиск без учета регистра символов.
* **-w**- поиск совпадений целого слова.
* **-l** - вывод только имен файлов, в которых найдены участки, совпадающие с заданным образцом/шаблоном, без вывода совпадающих строк.
* **-r** - (рекурсивный поиск) поиск выполняется в текущем каталоге и всех вложенных подкаталогах. Сами строки ищутся в файлах.
* **-n** - добавляет номер найденной строки в начало ее вывода.
* **-v** (или **--invert-match**) - выводит только строки, не содержащие совпадений.
* **-c (--count)** - выводит количество совпадений без вывода самих совпадений.
* -**e –** работает как или.  
  grep -e 'Class 1' -e Todd Students.txt – искать либо ‘Class 1’ либо Todd.
* -**E –** искать по регулярке  
  grep -E p\{2} fruits.txt – найти строки, в которых есть 2 подряд идущих буквы ‘p’
* **-C number –** показать строки сверху и снизу от искомой
* **-A number –** показать строки снизу от искомой
* **-** **B number –** показать строку сверху от искомой

ls | gre1p 'My Documents' – перенаправить вывод

grep 'Class 1' Students.txt – поиск по файлу

grep -E 'string1|string2' filename – искать в файле строки string1 или string2

grep '\<seven' test.txt – поиск по началу слова

grep 'seven\>' test.txt – поиск по концу слова

**wc** - счетчик слов в файле или в потоке

* **-w** подсчитывает только слова.
* **-l** подсчитывает только строки.
* **-c** подсчитывает только символы.
* **-L** возвращает длину наибольшей строки.

**echo**

**tr** - Замена одних символов на другие.

Синтаксис: **tr -key [набор1] [набор2]**

**history –** вывести историю недавно введенных команд.

**date –** показать текущую дату и время.

**sudo command –** выполнить операцию от имени суперпользователя. Требует ввода пароля пользователя.

**sudo su –** перейти в режим суперпользователя. (su username – обратно)

**apt-get –** пакетный менеджер (установка, обновление удаление пакетов и программ).

* **install –** установить
* **remove –** удалить
* **update –** обновить список пакетов

**whereis command** – позволяет узнать местоположения бинарных, исходных файлов и файлов справочной страницы (man). Если нет такой команды, то будет выведено только указанное имя.

* **-l** – показать каталоги, в которых ищет wheris
* -**p –** поиск только исполняемых файлов
* -**s –** поиск только исходных файлов
* **-m –** поиск только мануала

**whatis –** краткий обзор команды, параметра или флага.

**which -**

**whoami –** вывести текущего пользователя.

**ifconfig –** посмотреть список сетевых интерфейсов, их состояние, IP адрес, MAC адрес и другие параметры.

**ip –** улучшенная версия ifconfig.

**netstat** тоже умеет показывать сетевые интерфейсы и статистику по переданным данным если ей передать опцию **-i**.

**РАБОТА С ФАЙЛАМИ**

В линуксе все есть файл, с каждым файлом связан **дискриптор**, это, как правило, небольшие целые положительные числа, используемые ядром для идентификации файлов.

**mkdir –** создает новый каталог в текущем каталоге.

**touch –** команда предназначенная для установки времени последнего изменения файла или доступа в текущее время. Удобно использовать для создания пустых файлов.

**rm –** удаление файлов или каталогов:

* **-r —** обрабатывать все вложенные подкаталоги. Без этого флага не удалить даже пустой каталог.
* **-i –** выводить запрос на подтверждение каждой операции удаления.
* **-f –** принудительно удалить файл.

Опция **-с** пропускает создание файла. Т.о. можно проверить существование файла.

**Cp file1 file2 –** копирование файлов (каталогов). Если надо скопировать несколько разных файлов в один и тот же каталог, то после команды cd сначала перечисляются все файлы-оригиналы, а последним указывается каталог, куда помещаются копии.

cp /home/irina/tux.png /mnt/D/pingvin.png

* **-r -**рекурсивное копирование каталога. Если каталога нет, будет создан.
* **-a –** копирование всех атрибутов и прав. Так же рекурсивно копирует каталоги.
* **-H –** копировать символические ссылки. По умолчанию команда переходит по символическим ссылкам и копирует сами файлы.
* **-n –** не перезаписывать существующий файл.

Мы можем дополнить файлы в директории-приёмнике, если папки называются одинаково, но если они отличаются, то папка-исходник будет помещена внутрь приёмника.

Чтобы это обойти используем специальные псевдоссылки "." и "..", на текущую и вышестоящие директории.

cp -a /source/. /target

**Mv file1 file2** – переместить (или переименовать, если file1 и file2 находятся в одном каталоге)

mv document.txt Work – переместить файл в папку Work

mv document.txt Work/doc23.txt – переместить и переименовать

**find [папка] [параметры] критерий шаблон [действие] –** команда для поиска файлов и каталогов на основе специальных условий.

* **Папка**- каталог в котором будем искать
* **Параметры**- дополнительные параметры, например, глубина поиска, и т д
* **Критерий**- по какому критерию будем искать: имя, дата создания, права, владелец и т д.
* **Шаблон**- непосредственно значение по которому будем отбирать файлы.

Основные параметры:

* **-P** никогда не открывать символические ссылки
* **-L** - получает информацию о файлах по символическим ссылкам. Важно для дальнейшей обработки, чтобы обрабатывалась не ссылка, а сам файл.
* **-maxdepth** - максимальная глубина поиска по подкаталогам, для поиска только в текущем каталоге установите 1.
* **-depth** - искать сначала в текущем каталоге, а потом в подкаталогах
* **-mount** искать файлы только в этой файловой системе.
* **-version** - показать версию утилиты find
* **-print** - выводить полные имена файлов
* **-type f** - искать только файлы
* **-type d** – искать папки.

Критерии:

* **-name** - поиск файлов по имени  
  find /path/to/search -name ".\*" – найти скрытые файлы и директории
* **-iname –** поиск по имени без учета регистра.
* **-user**- поиск файлов по владельцу  
  find /path/to/search -user root
* **-group** - поиск по группе
* **-mtime**- поиск по времени модификации файла  
  $ find /path/to/search -type f -mtime 30

$ find /path/to/search -type f -mtime +30<

$ find /path/to/search -type f -mtime -30

* **-atime** - поиск файлов по дате последнего чтения
* **-nogroup** - поиск файлов, не принадлежащих ни одной группе
* **-nouser** - поиск файлов без владельцев
* **-newer**- найти файлы новее чем указанный
* **-size**- поиск файлов в Linux по их размеру  
  *find /path/to/search -size +100M -size -1G* – найти файл размером больше 100Мб и меньше 1Гб.
* -**empty -** позволяет найти пустые файлы и каталоги
* -**perm –** поиск файлов по набору разрешений

**ПОЛЬЗОВАТЕЛИ, ГРУППЫ, ПРАВА ДОСТУПА**

Пользователи также имеют свои идентификаторы. Пользователь с идентификатором 0 называется **суперпользователем** или **root**. Cуперпользователь обладает неограниченной свободой действий в системе.

В файле паролей(/etc/passwd ) у каждого пользователя также содержится идентификатор группы. Группы позволяют организовать совместное использование ресурсов, и удобное управление правами доступа.

Пользователь имеет основную группу (обычно совпадает с именем пользователя), и дополнительные.

Идентификаторы групп и их названия хранятся в файле /etc/group.

Только root и пользователь sudo могут добавлять пользователя к роли.

**usermod -a -G –** добавить пользователя во вторичную группу

sudo usermod -a -G group1,group2 username

без опции *-a* пользователь будет удален из всех групп кроме указанных.

**usermod -g** – изменить основную группу пользователя

sudo usermod -g groupname username

**gpasswd -d –** удалить пользователя из группы

sudo gpasswd -d username groupname

**groupadd –** добавить группу

sudo groupadd groupname

**groupdel –** удалить группу.

sudo groupdel groupname

**useradd –** добавить юзера, можно сразу и роли назначить

sudo useradd -g users -G wheel,developers nathan

**chmod опции права /путь/к/файлу –** изменение прав файла.

Опция **-R** для изменения прав на все файлы в папке.

**права доступа:**

* **r**- чтение;
* **w**- запись;
* **x**- выполнение;
* **s**- выполнение  от имени суперпользователя (дополнительный);

**категории пользователей:**

* **u**- владелец файла;
* **g**- группа файла;
* **o**- все остальные пользователи;
* **a** – все пользователи

**Действия: "+" - включить "-" – выключить**

* **u+x** - разрешить выполнение для владельца;
* **ugo+x** - разрешить выполнение для всех;
* **ug+w** - разрешить запись для владельца и группы;
* **o-x** - запретить выполнение для остальных пользователей;
* **ugo+rwx** - разрешить все для всех;

Права можно записывать цифрами. первая цифра - владелец файла, вторая - группа файла, третья - все остальные пользователи. Цифры это перевод из двоичной записи трехразрядного числа.

\*\*\* - Первая цифра – разрешение на чтение, вторая на запись, третья на выполнение. 1 разрешить, 0 запретить.

* 0 (000)- никаких прав;
* 1 (001) - только выполнение;
* 2 (010)- только запись;
* 3 (011) - выполнение и запись;
* 4  (100)-  только чтение;
* 5 (101) - чтение и выполнение;
* 6  (110)- чтение и запись;
* 7 (111)- чтение запись и выполнение.

Пример: chmod 764 fileName.txt

Недостаток цифровой записи в том, что вы не можете модифицировать уже существующие права доступа linux.

Возможность доступа к файлу зависит также от прав доступа к **каталогу**, в котором находится файл. Чтобы воспользоваться имеющимися у вас правами доступа к файлу, вы должны иметь право на исполнение для всех каталогов вдоль пути к файлу.

**РАБОТА С ПРОЦЕССАМИ**

Программа, находящаяся в процессе выполнения, называется **процессом**.

Каждому процессу присваивается свой идентификатор, называемый **Идентификатором процесса.**

Взаимодействие с процессами зачастую происходит посредством сигналов.

Сигнал в ос семейства Unix — асинхронное уведомление процесса о каком-либо событии, один из основных способов взаимодействия между процессами. Когда сигнал послан процессу, операционная система прерывает выполнение процесса, при этом, если процесс установил собственный обработчик сигнала, операционная система запускает этот обработчик, передав ему информацию о сигнале, если процесс не установил обработчика, то выполняется обработчик по умолчанию.

**ps –** вывод списка активных процессов (всех процессов).

-**l** – список приостановленных задач

**aux –** вывод всех активных процессов с дополнительной информацией.

USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND  
root       661  0.0  0.0   4072     8 tty1     Ss+  Jul03   0:00 /sbin/mingetty

\* где:

* **USER —** учетная запись пользователя, от которой запущен процесс.
* **PID —** идентификатор процесса.
* **%CPU —** потребление процессорного времени в процентном эквиваленте.
* **%MEM —** использование памяти в процентах.
* **VSZ —** Virtual Set Size. Виртуальный размер процесса (в килобайтах).
* **RSS —** Resident Set Size. Размер резидентного набора (количество 1K-страниц в памяти).
* **TTY —** терминал, из под которого был запущен процесс.
* **STAT —** текущее состояние процесса. Могут принимать значения:
  1. **R —** выполнимый процесс;
  2. **S —** спящий;
  3. **D —** в состоянии подкачки на диске;
  4. **T —** остановлен;
  5. **Z —** зомби.
  6. **W —** не имеет резидентных страниц;
  7. **< —** высоко-приоритетный;
  8. **N —** низко-приоритетный;
  9. **L —** имеет страницы, заблокированные в памяти.
* **START —** дата запуска процесса.
* **TIME —** время запуска процесса.
* **COMMAND —** команда, запустившая процесс.

-**e –** все процессы

-**a –** запущенные в текущем терминале, кроме главных системных.

**r –** список только работающих

**f –** показать дерево процессов с родителями

**u –** показать пользователей, запустивших процесс

**x –** отсоединенные от терминала

-**p** – показать информацию только об указанном процессе.

--**sort** – сортировка процессов по указанному критерию. Для сортировки в обратном порядке указывается знак минус перед %.

ps aux --sort=%mem

**pstree**  - построить дерево процессов.

**pgrep -** поиск процессов.

**nice –** задать приоритет процессу при его запуске(от 19(min) до -20(max)). Уменьшить приоритет может любой пользователь, а увеличить только root.

nice -n 10 apt-get upgrade

**renice –** изменить приоритет для уже существующего процесса.

renice -n 10 -p 1343

**top (htop) –** статистика потребления ресурсов.

**kill pid… –** убить процесс (несколько процессов) по его pid, либо послать ему другой сигнал.

**-s –** указать посылаемый сигнал.

-**p –** указать что команда kill должна вывести идентификаторы процессов, но не посылать им сигнал.

-**l –** вывод списка имен сигналов.

Перезагрузить процесс можно сигналом SIGHUP

SIGTERM (15) – остановить процесс, используется по умолчанию. При этом процесс сохранит все необходимые данные, и выполнит безопасное завершение. Может быть проигнорирован.

SIGKILL (9) – остановить процесс принудительно, без каких-либо сохранений.

SIGINT (2) – завершение процесса, посылается при нажатии Ctrl+c

**pkill** позволяет использовать расширенные шаблоны регулярных выражений и другие критерии соответствия.

Например, для завершения работы Firefox просто введите команду:

*$ pkill firefox*

Так как оно соответствует шаблону регулярного выражения, вы можете также ввести имя только частично, например:

*$ pkill fire*

**killall** proc – убить все процессы с именем proc.

**bg** — вывод списка приостановленных или фоновых задач и запуск выполнения приостановленной задачи в фоне;

**sleep** pid – приостановить задачу

**fg** — «разбудить» последнюю приостановленную задачу;

**fg** **n** — «разбудить» приостановленную задачу под номером n в списке.

**free** – вывести свободное и используемое кол-во памяти в системе

**ПАКЕТНЫЕ ФАЙЛЫ**

Скриптовые файлы имеют расширение **.sh**.

**#!/bin/bash** – первая обязательная строка в скрипте (**Шебанг**). Указывает название оболочки скрипта.

Когда скрипт с шебангом выполняется как программа в [Unix-подобных](https://ru.wikipedia.org/wiki/UNIX-%D0%BF%D0%BE%D0%B4%D0%BE%D0%B1%D0%BD%D0%B0%D1%8F_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0" \o "UNIX-подобная операционная система) операционных системах, загрузчик программ рассматривает остаток строки после шебанга как имя файла [программы-интерпретатора](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D1%80%D0%B5%D1%82%D0%B0%D1%82%D0%BE%D1%80). Загрузчик запускает эту программу и передаёт ей в качестве параметра имя файла скрипта с шебангом.

Запустить исполняемый файл можно указав путь к нему, либо командой bash.

**./myscript** либо **bash** file.sh

Добавив символ & в конец команды можно запустить её **в фоновом режиме**.

**Комментарии** обозначаются символом решетка #.

Переход на новую строку является **разделителем** команд, но также можно использовать точку с запятой.

**set –** используется для получения списка всех переменных оболочки, переменных окружения, локальных переменных и функций оболочек.

Команда set позволяет вам управлять определенными флагами и опциями в Bash, чтобы определять поведение ваших скриптов. Так же позволяет устанавливать и перезаписывать позиционные аргументы внутри скрипта.

* **set -c** – запретить перезаписывать файлы
* **set** -**e –** прервать работу скрипта, при возникновении ошибки в одной из команд.
* **set** -**u** – прервать работу скрипта с ошибкой, если идет обращение к неинициализированной переменной.
* **set** -**x –** печатать команды и их параметры перед выводом результата их работы.
* **set** -**o** – посмотреть текущие опции оболочки

Значение переменных можно использовать в сценариях с предшествующим знаком доллара (**$user**).

В скриптах пользователи могут создавать свои собственные переменные, эти переменные относятся к переменным оболочки.

Переменные оболочки не наследуются дочерними оболочками, Используются для информации, специфической для конкретного экземпляра оболочки, например текущего рабочего каталога.

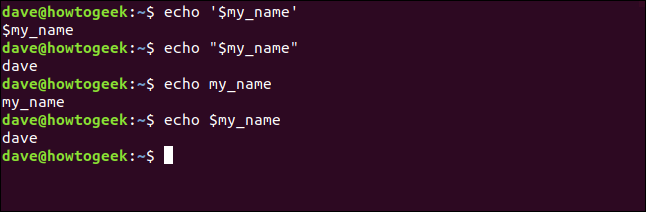
Переменные окружения наследуются. Переменные окружения используются для передачи информации в дочерние процессы.

Чтобы создать переменную, вы просто указываете ее имя и значение.

*var1=50*

*var2=-120  
var3=test  
var4="текстовая строка"*

* Переменная в одинарных кавычках ' обрабатывается как буквенная строка, а не как переменная.
* Переменные в кавычках " рассматриваются как переменные.
* Чтобы получить значение, хранящееся в переменной, необходимо указать знак доллара $ .



Чтобы присвоить переменной результат выполнения команды используется конструкция **$( ).**

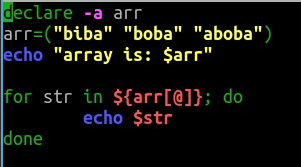
*file\_count = $ (ls $ folder\_to\_count | wc -l)*

Оператор += **объединяет две строки**.

**declare и typeset –** предназначены для наложения ограничений на переменные (эти две команды абсолютно идентичны). Можно передавать флаги:

* -r readonly
* -i integer
* -a array
* -f function
* -x export

*declare -i var*



**env –** запустить команду или программу с измененным окружением. При вызове без параметров выводит список всех переменных окружения.

**-i** – запуск команды с пустым окружением.

-**u=name –** удалить параметр из окружения

-**С=DIR –** изменить рабочую директорию на DIR

*env -i ./my\_script*

*env VAR1="value" command\_to\_run –* определить

Установить переменную оболочки можно так же командой **export.** Такая переменная будет доступна как в родительском, так и в дочернем скрипте.

export VAR="value"

**$PATH** — это переменная среды, используемая для указания оболочке, где искать исполняемые файлы. Содержит список каталогов, разделенных двоеточиями. Чтобы добавить каталог в path:

export PATH=$PATH:/opt/local/bin

Но это только для текущего сеанса терминала.

Чтобы обновить path(и любую переменную) на постоянке, нужно изменить файл **/etc/environment**. Значение переменной будет доступно всем пользователям в системе.

Можно изменить файл **.bashrc**, это скрипт, выполняемый всякий раз, когда вы инициализируете сеанс интерактивной оболочки. В него нужно дописать export необходимой переменной. *export TZ="America/New\_York"*

**read var\_name** – чтение ввода в переменную.

Доступ к параметрам командной строки осуществляется с помощью знака “**$”** и номера переменной. Нулевой параметр всегда содержит **имя скрипта**.

отличия set от declare.

| **Специальная переменная** | **Описание** |
| --- | --- |
| $0 | Имя сценария bash. |
| $1, $2 … $n | Аргументы сценария bash. |
| $$ | Идентификатор процесса текущей оболочки. |
| $# | Общее количество аргументов, переданных скрипту. |
| $@ | Значение всех аргументов, переданных скрипту. |
| $? | Статус выхода последней выполненной команды. |
| $! | Идентификатор процесса последней выполненной команды. |

Чтобы проверить наличие переданных параметров:

#!/bin/bash

if [ -n "$1" ]

then

echo Hello $1.

else

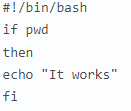
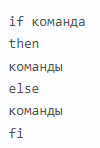
echo "No parameters found. "

fi

Для сдвига параметров на одну позицию используется команда **shift**. Это позволяет обрабатывать неизвестное количество параметров. Нулевой параметр не смещается, он всегда содержит название программы.

#!/bin/bash  
count=1  
while [ -n "$1" ]  
do  
echo "Параметр №$count = $1"  
count=$[ $count + 1 ]  
shift  
done

Конструкции **if then** и **if then else**:

**Сравнение –** *if[$val1 > $val2]*

Как вырубить комп: sudo shutdown -r now