**Командная оболочка BASH**

Оболочка GNU/Linux (GNU/Linux Shell) — это специальная интерактивная утилита.

Ядром оболочки является командная строка.

Командная строка — это интерактивная часть оболочки. Она позволяет вам вводить текстовые команды, а затем интерпретирует команды и выполняет их в ядре.

Оболочкой по умолчанию, используемой во всех дистрибутивах Linux, является оболочка bash.

|  |
| --- |
| Самый старый и самый простой из пакетов эмуляции терминала — это xterm.  Пакет xterm существует еще до появления X Window, популярного сервера отображения, и он часто включается в дистрибутивы по умолчанию. |

Таблица 1. Оболочки Linux

|  |  |
| --- | --- |
| Оболочка (shell) | Описание |
| ash | простая легковесная оболочка, которая работает в средах с малым объемом памяти, но обладает полной совместимостью с оболочкой bash |
| korn | оболочка программирования, совместимая с оболочкой Bourne, но поддерживающая расширенные функции программирования, такие как ассоциативные массивы и арифметика с плавающей точкой |
| tcsh | оболочка, которая включает элементы языка программирования C в сценарии оболочки |
| zsh | усовершенствованная оболочка, включающая функции bash, tcsh и korn, предоставляющая расширенные функции программирования, общие файлы истории и тематические подсказки. |

Таблица 2. Описание основных встроенных команд

|  |  |
| --- | --- |
| **Встроенная команда** | **Описание** |
| break | выход из цикла for, while или until |
| continue | выполнение следующей итерации цикла for, while или until |
| echo | вывод аргументов, разделённых пробелами, на стандартное устройство вывода |
| exit | выход из оболочки |
| export | отмачает аргументы как переменные для передачи в дочерние процессы в среде |
| hash | запоминает полные имена путей команд, указанных в качестве аргументов, чтобы не искать их при следующем обращении |
| kill | посылает сигнал завершения процессу |
| pwd | выводит текущий рабочий каталог |
| read | читает строку из ввода оболочки и использует её для присвоения значений указанным переменным |
| return | заставляет функцию оболочки выйти с указанным значением |
| shift | перемещает позиционные параметры налево |
| test | вычисляет условное выражение |
| times | выводит имя пользователя и системное время, использованное оболочкой и её потоками |
| trap | указывает команды, которые должны выполняться при получении оболочкой сигнала |
| unset | вызывает уничтожение переменных оболочки |
| wait | ждёт выхода из дочернего процесса и сообщает выходное состояние |

1. Любой bash-скрипт должен начинаться со строки:

#!/bin/bash

в этой строке после #! указывается путь к bash-интерпретатору.

Узнать, где находится bash-интерпретатор: $ whereis bash

1. Комментарии начинаются с символа #
2. В bash переменные не имеют типа

Таблица . Простой скрипт с комментариями

|  |  |
| --- | --- |
| Скрипт | # Комментарии |
| #!/bin/bash | указываем расположение bash-интерпретатора |
| 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" | вывод строки с подстановкой значений переменных (обращение к переменным через $имя\_переменной) |
| echo 'Вы запустили скрипт с именем $script\_name и параметром $parametr1' | в одинарных кавычках не происходит подстановки переменных |
| exit 0 | Выход с кодом 0 (удачное завершение работы скрипта) |
| Результат выполнения скрипта: | |
| ite@ite-desktop:~$ ./test.sh qwerty | |
| Вы запустили скрипт с именем ./test.sh и параметром qwerty | |
| Вы запустили скрипт с именем $script\_name и параметром $parametr1 | |

Таблица . Зарезервированные переменные

|  |  |
| --- | --- |
| **Переменная** | **Значение** |
| $DIRSTACK | содержимое вершины стека каталогов |
| $EDITOR | текстовый редактор по умолчанию |
| $EUID | Эффективный UID. Если команда выполняется от лица другого пользователя (например, superuser), то эта переменная содержит UID этого пользователя |
| $UID | содержит реальный идентификатор, который устанавливается только при логине. |
| $FUNCNAME | имя текущей функции в скрипте |
| $GROUPS | массив групп к которым принадлежит текущий пользователь |
| $HOME | домашний каталог пользователя |
| $HOSTNAME | имя хоста |
| $HOSTTYPE | архитектура машины |
| $LC\_CTYPE | внутренняя переменная, которая определяет кодировку символов |
| $OLDPWD | прежний рабочий каталог |
| $OSTYPE | тип ОС |
| $PATH | путь поиска программ |
| $PPID | идентификатор родительского процесса |
| $SECONDS | время работы скрипта(в сек.) |
| $# | общее количество параметров переданных скрипту |
| $\* | все аргументы, переданные скрипту (выводятся в строку) |
| $@ | все аргументы, переданные скрипту, (параметры выводятся в столбик) |
| $! | PID последнего запущенного в фоне процесса |
| $$ | PID самого скрипта |

Логическая структура оператора ветвления (if-then-else):

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, тот этот>

В качестве команд возвращающих код возврата могут выступать структуры:

* [ (равносильно test) - используется для логического сравнения. Должна иметь ”]”
* [[ - расширенная версия “[ ]”; внутри структуры могут быть использованы || («или»), & («и»).  
  Должна иметь закрывающую скобку "]]"
* (( )) – математическое сравнение
* …любая другая команда или несколько команд

Пример:

**if [[ "$source" -eq "$dest" ]]** # -eq - логическое сравнение (эквивалент ==)  
**then**  # если true ("$source" == "$dest"), то  
**echo "Применик $dest и источник $source один и тот же файл!"**   
**exit 1** # выходим с кодом ошибки 1  
**else** # если false ("$source" != "$dest")  
**cp $source $dest** # то копируем источник в приемник  
**echo "Удачное копирование!"**  
**fi**  #обозначаем окончание оператора ветвления

End

https://linux.die.net/Bash-Beginners-Guide/chap\_01.html