

< Teach  
Me  
Skills />

# Операционные системы. Часть 3



- Язык командного интерпретатора Bash
- Настройка репозитория
- Работа с пакетными менеджерами

# Язык командного интерпретатора Bash

Безусловно, все те кто общается с ОС Linux хоть раз да имели дело(во всяком случае слышали точно) с командной оболочкой BASH. Но BASH не только командная оболочка, это еще и превосходный скриптовый язык программирования.

Цель этого урока — познакомиться поближе с bash, рассказать про синтаксис, основные приемы и фишки языка, для того чтобы даже обычный пользователь смог быстро написать простой скрипт для выполнения ежедневной(-недельной, -месячной) рутинной работы или, скажем, «на коленке» наваять скриптик для бэкапа директории.

BASH — Bourne-Again SHell (что может переводиться как «перерожденный шел», или «Снова шел Борна(создатель sh)»), самый популярный командный интерпретатор в юниксоподобных системах, в особенности в GNU/Linux. Ниже приведу ряд встроенных команд, которые мы будем использовать для создания своих скриптов.

## Bash - команды

**break** выход из цикла `for`, `while` или `until`

**continue** выполнение следующей итерации цикла `for`, `while` или `until`

**echo** вывод аргументов, разделенных пробелами, на стандартное устройство вывода

**exit** выход из оболочки

**export** отмечает 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 ее 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 ее потомками

**trap** указывает команды, которые должны выполняться при получении оболочкой сигнала

**unset** вызывает уничтожение переменных оболочки  
**wait** ждет выхода из дочернего процесса и сообщает  
выходное состояние.



Что необходимо знать с самого начала:

1. Любой bash-скрипт должен начинаться со строки:

```
#!/bin/bash
```

в этой строке после `#!` указывается путь к bash-интерпретатору, поэтому если он у вас установлен в другом месте(где, вы можете узнать набрав **whereis bash**) поменяйте её на ваш путь.

2. Комментарии начинаются с символа `#` (кроме первой строки).

3. В bash переменные не имеют типа(о них речь пойдет ниже)

# Переменные и параметры скрипта

Приведем как пример небольшой скрипт, который мы разберем:


```
#!/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)
```

## 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`



После того как мы познакомились как использовать переменные и передавать скрипту параметры, время познакомиться с зарезервированными переменными:

# Зарезервированные переменные

**\$DIRSTACK** – содержимое вершины стека каталогов

**\$EDITOR** – текстовый редактор по умолчанию

**\$EUID** – Эффективный UID. Если вы использовали программу su для выполнения команд от другого пользователя, то эта переменная содержит UID этого пользователя, в то время как...

**\$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** – ваш hostname

**\$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)

**\$@** – тоже самое, что и предыдущий, но параметры выводятся в столбик

**#!** – PID последнего запущенного в фоне процесса

**\$\$** – PID самого скрипта

Условные операторы, думаю, знакомы практически каждому, кто хоть раз пытался на чем-то писать программы. В bash условия пишутся след. образом:

```
#!/bin/bash
source=$1 #в переменную source добавляем первый
параметр скрипта
dest=$2 #в переменную dest добавляем второй
параметр скрипта
```

```
if [[ "$source" -eq "$dest" ]] # в кавычках указываем имена
переменных для сравнения. -eq - логическое сравнение
обозначающие "равны"
then # если они действительно равны, то
echo "Приемник $dest и источник $source один и тот же
файл!" #выводим сообщение об ошибке, т.к. $source и $dest у
нас равны
exit 1 # выходим с ошибкой (1 - код ошибки)
else # если же они не равны
cp $source $dest # то выполняем команду cp: копируем
источник в приемник
echo "Удачное копирование!"
fi #обозначаем окончание условия.
```



## Результат выполнения скрипта:

```
ite@ite-desktop:~$ ./primer2.sh 1 1
```

Приемник 1 и источник 1 один и тот же файл!

```
ite@ite-desktop:~$ ./primer2.sh 1 2
```

Удачное копирование!

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, тот этот>
```

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

`test` - используется для логического сравнения. после выражения, необходима закрывающая скобка `"]`

`[` - синоним команды `test`

`[[` - расширенная версия `"["` (начиная с версии 2.02)(как в примере), внутри которой могут быть использованы `||` (или), `&` (и). Должна иметь закрывающую скобку `"]]"`

(( )) - математическое сравнение.

для построения многоярусных условий вида:

```
if ...  
then ....  
else  
if ....  
then....  
else ....
```

для краткости и читаемости кода, можно использовать структуру:

```
if ..  
then ...  
elif ...  
then ...  
elif ...
```

## Условия. Множественный выбор

Если необходимо сравнивать какую-то одну переменную с большим количеством параметров, то целесообразней использовать оператор `case`.

```
#!/bin/bash
echo "Выберите редактор для запуска:"
echo "1 Запуск программы nano"
echo "2 Запуск программы vi"
echo "3 Запуск программы emacs"
echo "4 Выход"
read doing #здесь мы 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.

## Результат работы:

```
ite@ite-desktop:~$ ./menu2.sh
```

Выберите редактор для запуска:

- 1 Запуск программы nano
- 2 Запуск программы vi
- 3 Запуск программы emacs
- 4 Выход



После выбор цифры и нажатия Enter запустится тот редактор, который вы выбрали(если конечно все пути указаны правильно, и у вас установлены эти редакторы :) )

Приведу список логических операторов, которые используются для конструкции if-then-else-fi:

- z # строка пуста
- n # строка не пуста
- =, (==) # строки равны
- != # строки неравны

-ne # не равно  
-lt, (< ) # меньше  
-le, (<=) # меньше или равно  
-gt, (>) # больше  
-ge, (>=) # больше или равно  
! # отрицание логического выражения  
-a, (&&) # логическое «И»  
-o, (||) # логическое «ИЛИ»

# Настройка репозитория

Установка ПО в системах Linux выполняется из репозиториев, которые по умолчанию содержат большое количество пакетов. Однако иногда необходимого софта нет в комплекте или его версия устарела. В этом случае вы можете добавить требуемый репозиторий и произвести установку из него.

Будьте осторожны при выполнении таких операций, потому что такие сборки могут содержать экспериментальные версии системного ПО или даже ядро Linux. Поэтому следует пристально изучить информацию о стороннем репозитории, а также менеджер обновлений.

Ниже мы настроим репозитории на сервере Ubuntu 20.04.

## Список репозиториев в Ubuntu

Просмотреть все репозитории:

```
nano /etc/apt/sources.list
```

Они также могут находиться в одном из файлов в папке `/etc/apt/sources.list.d/`

Чтобы отключить один репозиторий, добавьте следующий комментарий в его строку:

```
# deb http://archive.ubuntu.com/ubuntu focal  
multiverse
```

# Добавление репозитория в Ubuntu

Чтобы добавить репозиторий, необходимо узнать его адрес у разработчика ПО и использовать команду **apt-add-repository** с подобным синтаксисом:

```
apt-add-repository 'deb http://repository_address version branch'
```

Иногда требуется сначала установить ключ GPG. В качестве примера возьмем MariaDB.

```
apt-key adv --fetch-keys
```

```
'https://mariadb.org/mariadb_release_signing_key.asc'
```

Команда, используемая для добавления репозитория:

```
add-apt-repository 'deb [arch=amd64,arm64,ppc64el]
```

```
http://mirror.mephi.ru/mariadb/repo/10.5/u
```

Команда, используемая для удаления:

```
add-apt-repository --remove 'deb  
[arch=amd64,arm64,ppc64el]  
http://mirror.mephi.ru/mariadb/repo/10.5/ubuntu  
focal main'
```



Во время установки PPA-репозитория система автоматически распознает репозиторий и скачивает необходимые ключи.

```
apt-add-repository ppa:repository/ppa
```

Для удаления PPA-репозитория:

```
apt-add-repository --remove ppa:repository/ppa
```

После редактирования списка репозиториев не забудьте обновить список пакетов.

```
apt update
```

# Работа с пакетными менеджерами

Пакетными менеджерами принято называть приложения, предназначенные для управления программным обеспечением, которое установлено в системе или может быть установлено из репозиториев.

Почему в unix-подобных ОС системы управления ПО называют "пакетными"? Дело в том, что программы для Linux распространяются в архивах, содержащих множество файлов, а не один исполняемый, как это обычно происходит в случае Windows. Такие архивы, содержащие файлы самой программы, метаданные и другие файлы, принято называть пакетами.

Пакетный менеджер определенным образом распаковывает архив, организует выполнение ряда команд таким образом, чтобы программное обеспечение правильно установилось, были соблюдены все зависимости, не допускает конфликтов с другим ПО.

В мире Linux существует целый ряд пакетных менеджеров. Их распространенность зависит от степени популярности дистрибутивов, в которых они используются. Так часто встречается пакетный менеджер APT (advanced package tool), так как он используется семейством Debian, к которому относятся Ubuntu, Linux Mint и др. Среди других менеджеров можно отметить YUM для дистрибутивов, пакеты которых распространяются в формате RPM (Fedora, CentOS и др).

Программы **apt** и **apt-get** (аналог apt, появился раньше, более низкоуровневый) запускаются из командной строки. Однако для них бывают различные надстроенные графические интерфейсы. Таким образом, обычный пользователь может устанавливать и удалять программы в привычной для него среде. Однако вся гибкость управления программами доступна в основном через Bash.



# Пакетный менеджер APT

```
pl@comp:~$ apt --help
```

```
apt 1.6.6 (amd64)
```

```
Использование: apt [параметры] команда
```

apt – менеджер пакетов с интерфейсом командной строки, предоставляет команды для поиска и управления, а также запросов информации о пакетах.

Он выполняет те же задачи, что и специализированные инструменты APT, например apt-get и apt-cache, но содержит параметры, которые больше подходят для интерактивного использования по умолчанию.

Основные команды:

- list - показать список пакетов из указанных имён пакетов

- search - искать в описаниях пакетов

- show - показать дополнительные данные о пакете

- install - установить пакеты

- remove - удалить пакеты

- autoremove - автоматически удалить все неиспользуемые пакеты

- update - обновить список доступных пакетов

- upgrade - обновить систему, устанавливая/обновляя пакеты

- full-upgrade - обновить систему, удаляя/устанавливая/обновляя пакеты

- edit-sources - редактировать файл с источниками пакетов

Дополнительную информацию о доступных командах смотрите в apt(8).

Параметры настройки и синтаксис описаны в apt.conf(5).

Информацию о том, как настроить источники, можно найти в sources.list(5).

Выбор пакетов и версий описывается через apt\_preferences(5).

Информация о безопасности доступна в apt-secure(8).

В APT есть коровья СУПЕРСИЛА.

```
pl@comp:~$ █
```

С помощью первого аргумента (он же "команда" на скрине), передаваемого программе apt, мы указываем, что хотим сделать: установить программу, удалить, обновить. Вторым аргументом идет имя пакета или его часть, если выполняется поиск. Как для команды, так и для имен пакетов работает автодополнение. Программы берутся из списка источников ПО – репозиториев, список которых можно редактировать.

Поскольку изменение установленного на компьютере программного обеспечения относится к административным задачам, то для части команд apt необходимы права администратора. Таким образом, полный синтаксис установки и удаления ПО таков:

```
sudo apt install имя_пакета  
sudo apt remove имя_пакета
```



Пусть надо установить консольный файловый менеджер Midnight Commander. Имени пакета мы не знаем, поэтому воспользуемся командой `search` утилиты `apt`. На экране появится список пакетов с кратким описанием. Очевидно нам нужен пакет `mc`.

```
pl@comp:~$ apt search "Midnight Commander"
Сортировка... Готово
Полнотекстовый поиск... Готово
avfs/bionic 1.0.5-2 amd64
  virtual filesystem to access archives, disk images, remote locations

gnome-commander/bionic 1.4.8-1.1 amd64
  Удобный и быстрый файловый менеджер для рабочего стола GNOME

gnome-commander-data/bionic,bionic 1.4.8-1.1 all
  Файлы данных для GNOME Commander

gnome-commander-dbg/bionic 1.4.8-1.1 amd64
  Отладочные символы для gnome-commander

junior-system/bionic,bionic 1.26ubuntu2 all
  Утилиты для Debian Jr. System

krusader/bionic 2:2.6.0-1 amd64
  двухпанельный (в стиле Commander) файловый менеджер

lfm/bionic,bionic 3.1-1 all
  простой, но мощный менеджер файлов для консоли UNIX

mc/bionic 3:4.8.19-1 amd64
  Midnight Commander - многофункциональный диспетчер файлов ✓

mc-data/bionic,bionic 3:4.8.19-1 all
  Midnight Commander - a powerful file manager -- data files

pilot/bionic 2.21+dfsg1-1build1 amd64
  Simple file browser from Alpine, a text-based email client

ranger/bionic,bionic 1.8.1-0.2 all
  File manager with an ncurses frontend written in Python

pl@comp:~$ sudo apt install mc
```

Далее устанавливаем программу, передав менеджеру пакетов команду `install`. Сначала `apt` соберет данные о пакете и его зависимостях. После этого попросит подтвердить ваше намерение установить пакет, и начнется процесс установки.

```
pl@comp:~$ sudo apt install mc
[sudo] пароль для pl:
Чтение списков пакетов... Готово
Построение дерева зависимостей
Чтение информации о состоянии... Готово
Следующие пакеты устанавливались автоматически и больше не требуются:
blueman cinnamon-common cjs gir1.2-appindicator3-0.1 gir1.2-caribou-1.0
gir1.2-clutter-1.0 gir1.2-cmenu-3.0 gir1.2-cogl-1.0 gir1.2-coglpango-1.0
gir1.2-gtkclutter-1.0 gir1.2-keybinder-3.0 gir1.2-meta-muffin-0.0
gnome-backgrounds libcaribou-common libcaribou0 libcjs0 libfm-data
libfm-extra4 libfm-gtk-data libfm-gtk4 libfm4 libkeybinder-3.0-0
libmenu-cache-bin libmenu-cache3 libmozjs-38-0 libnemo-extension1
linux-headers-4.15.0-39 linux-headers-4.15.0-39-generic
linux-image-4.15.0-39-generic linux-modules-4.15.0-39-generic
linux-modules-extra-4.15.0-39-generic lxmenu-data metacity-common nemo
nemo-data nemo-fileroller python-bs4 python-chardet python-html5lib
python-lxml python-pam python-pkg-resources python-pyinotify
python-webencodings sgml-base
Для их удаления используйте «sudo apt autoremove».
Будут установлены следующие дополнительные пакеты:
mc-data ← Зависимость
Предлагаемые пакеты:
arj catdvi | texlive-binaries dbview djvulibre-bin gv libaspell-dev links
| w3m | lynx odt2txt python-boto python-tz
НОВЫЕ пакеты, которые будут установлены:
mc mc-data
Обновлено 0 пакетов, установлено 2 новых пакетов, для удаления отмечено 0 п
Необходимо скачать 1 712 kB архивов.
После данной операции, объём занятого дискового пространства возрастёт на 7
Хотите продолжить? [Д/н] у ← Подтвердить
Пол:1 http://ru.archive.ubuntu.com/ubuntu bionic/universe amd64 mc-data all
Пол:2 http://ru.archive.ubuntu.com/ubuntu bionic/universe amd64 mc amd64 3:
Получено 1 712 kB за 0с (4 027 kB/s)
Выбор ранее не выбранного пакета mc-data.
(Чтение базы данных ... на данный момент установлено 249455 файлов и каталоги)
Подготовка к распаковке .../mc-data_3%3a4.8.19-1_all.deb ...
Распаковывается mc-data (3:4.8.19-1) ...
Выбор ранее не выбранного пакета mc.
Подготовка к распаковке .../mc_3%3a4.8.19-1_amd64.deb ...
Распаковывается mc (3:4.8.19-1) ...
Обрабатываются триггеры для mime-support (3.60ubuntu1) ...
Обрабатываются триггеры для desktop-file-utils (0.23-1ubuntu3.18.04.2) ...
Настраивается пакет mc-data (3:4.8.19-1) ...
Настраивается пакет mc (3:4.8.19-1) ...
update-alternatives: используется /usr/bin/mcview для предоставления /usr/b
Обрабатываются триггеры для man-db (2.8.3-2ubuntu0.1) ...
Обрабатываются триггеры для gnome-menus (3.13.3-11ubuntu1.1) ...
Обрабатываются триггеры для hicolor-icon-theme (0.17-2) ...
pl@comp:~$
```



С помощью `sudo apt autoremove` можно удалить неиспользуемые пакеты. Скорее всего они требовались для установки ПО. Если надо удалить саму программу, то используется команда `remove` утилиты `apt`.