нетология

Вводная информация

Командная оболочка **Unix** (англ. Unix shell) — командный интерпретатор, используемый в операционных системах семейства Unix, в котором пользователь может либо давать команды операционной системе по отдельности, либо запускать скрипты, состоящие из списка команд. В первую очередь, под shell понимаются POSIX-совместимые оболочки, восходящие к Bourne shell (шелл Борна), появившемуся в Unix Version 7.

Разновидности

- **sh** — оригинальный шелл Борна
  - **dash, bash, zsh** — другие современные клоны Bourne shell
  - **ksh** (KornShell) — клон шелла Борна, разработанный Дэвидом Корном из AT&T Labs. Синтаксис совместим, функциональность интерактивности увеличена
  - **pdksh** (public domain ksh) — открытая реализация ksh
  - **bash** (bourne again shell) (эмуляция совместимости POSIX) расширенная свободная (разработанная в рамках проекта GNU) оболочка ash, сходная с pdksh. Стандартная оболочка в Linux.
- **C shell** — (несовместима с POSIX shell) оболочка, с синтаксисом на основе Си, созданная Университетом Беркли в рамках проекта по реализации BSD Unix
  - **csh** (C-Shell) — оболочка из состава дистрибутива BSD, имеет Си-образный синтаксис и не является POSIX-совместимой. Впервые введены возможности управления заданиями и произведены другие улучшения
  - **tcsh** (csh) — реализация csh с интерактивными возможностями, не уступающими bash. Удобна для интерактивной работы. Совместима с csh.
- **ash** (Almquist shell, оболочка Альмквиста), BusyBox — современные микроверсии, предназначенные для встраиваемых систем, а также используемые в мини-дистрибутивах
- **xsh** (Xiki shell, от executable wiki) — новая командная оболочка, совмещающая командный и графический режимы работы.

Bash - самый популярный язык на данный момент, но на скриншотах затрагиваем и утилиты ОС. Поскольку у Bash синтаксис из **sh** (шелл), изучая его можно разобраться во всех остальных вариантах.

Bash (от англ. Bourne again shell) — усовершенствованная и модернизированная вариация командной оболочки Bourne shell. Одна из наиболее популярных современных разновидностей командной оболочки UNIX. Особенно популярна в среде **Linux**, где она часто используется в качестве

предустановленной командной оболочки. Также Bash доступен в **Jupyter Notebook**.

Bash представляет собой командный процессор, работающий, как правило, в интерактивном режиме в текстовом окне. Может читать команды из файла, который называется скриптом (или сценарием). Как и все Unix-оболочки, он поддерживает

- автодополнение имён файлов и каталогов,
- подстановку вывода результата команд,
- переменные, контроль над порядком выполнения,
- операторы ветвления и цикла.

Ключевые слова, синтаксис и другие основные особенности языка были заимствованы из sh. Другие функции, например, история, были скопированы из csh и ksh. Bash в основном соответствует стандарту POSIX, но с рядом расширений.

Bash-скрипт — это набор обычных команд, которые пользователь ввел с клавиатуры, собранные в файл. Говоря коротко - сценарий действий.
POSIX — набор стандартов, созданный для обеспечения совместимости Unix-подобных операционных систем.

Основные команды

Интерпретатор **bash** имеет множество встроенных команд, часть из которых имеет аналогичные исполняемые файлы в операционной системе. Следует обратить внимание, что чаще всего для встроенных команд отсутствуют man-страницы, а при попытке просмотра справки по встроенной команде будет выдаваться справка по исполняемому файлу. Исполняемый файл и встроенная команда могут различаться параметрами. Информация по встроенным командам расписана в справочной странице bash: **man bash**.

Примечание - **sudo** позволяет выполнить следующую команду в режиме администратора (linux).

Список основных команд

- **alias** - создание псевдонимов\имен для команд. В слово можно вставить команду и пользоваться alias, чтобы не прописывать что-то длинное

```
sweety@sweetappletooth:~/netology_test$ alias inst="sudo apt-get install"
sweety@sweetappletooth:~/netology_test$ inst bionic
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

- **bg** - переводит в фоновый режим приостановленные процессы
- **bind** - привязка команд к комбинациям клавиш
- **cd** - позволяет сменить текущую папку
- **declare** - объявление переменных
- **dirs** - история смены текущей папки
- **echo** - выводит указанные строковые значения на экран (можно сказать, print python'a)

```
sweety@sweetappletooth:~/netology_test$ echo 123
123
sweety@sweetappletooth:~/netology_test$ _
```

- **enable** - включение, либо отключение, встроенных команд
- **exit** - позволяет закрыть сеанс оболочки, выйти из терминала или завершить текущую сессию
- **export** - объявление переменных
- **fg** - позволяет восстановить команду из фона. В параметрах указываем id нужного процесса

```
                                                [ Wrote 1
^G Get Help   ^O Write Out   ^W Where Is   ^K Cut Te
^X Exit       ^R Read File   ^\ Replace    ^U Paste

Use "fg" to return to nano.

[1]+  Stopped                 nano file1
sweety@sweetappletooth:~$ _
```

- **history** - просмотр истории команд, которые выполнялись ранее
- **kill** - завершение процессов. Может быть как сис утилитой, так и командой Рекомендуем дополнительно почитать по теме

```
sweety@sweetappletooth:~$ ps xf
    PID TTY      STAT   TIME COMMAND
  10315 pts/0    S      0:00 -bash
  10528 pts/0    R+     0:00  \_ ps xf
   1695 ?        S      0:00 sshd: sweety@pts/0
   1696 pts/0    Ss     0:00  \_ -bash
   1582 tty1     S      0:00 -bash
   1611 tty1     S+     0:01  \_ ssh -p 22 sweety@localhost
   1576 ?        Ss     0:00 /lib/systemd/systemd --user
   1577 ?        S      0:00  \_ (sd-pam)
sweety@sweetappletooth:~$ sudo kill -9 1695
[sudo] password for sweety:
Connection to localhost closed by remote host.
Connection to localhost closed.
sweety@sweetappletooth:~$ _
```

- **let** - калькулятор
- **popd** - удаляет последнюю папку из стека
- **printf** - аналог echo, но с возможностью форматирования строк
- **pushd** - устанавливает указанную папку на верхушку стека директорий, тем самым меняет текущую рабочую директорию на указанную вами
- **pwd** - отображает текущую рабочую папку

```
sweety@sweetappletooth:~$ pwd
/home/sweety
sweety@sweetappletooth:~$ cd netology_test
sweety@sweetappletooth:~/netology_test$ pwd
/home/sweety/netology_test
sweety@sweetappletooth:~/netology_test$ _
```

- **read** - “читает” строку в переменную
- **sigcont** - снимает паузу
- **source** - читает и выполняет команды из файла, указанного в качестве аргумента, в текущем процессе оболочки
- **suspend** - ставит выполнение оболочки на паузу
- **time** - замеряет время выполнения скрипта или программы
- **umask** - изменение правил для прав
- **wait** - ожидание завершения определенного процесса
- **jobs** - выводит список фоновых задач текущей оболочки

Отличие **declare** и **export** в том, что второй метод экспортирует объявленные переменные во внешние среды. Таким образом, данные переменные будут доступны всем программам и скриптам.

Для работы с папками

- **cd** — переход с текущего каталога на домашний пользовательский
- **cd /** — переход по директориям относительно корневого каталога
- **cd название_папки** — перейти в определенную папку
- **ls** — просмотреть список файлов в каталоге
- **ls -a** - просмотреть список скрытых файлов
- **ls -d */** — просмотреть список папок в текущем каталоге\папке
- **ls -l** - проверить владельцев и права на файлы
- **ls -la** - просмотреть список скрытых файлов, с правами и владельцами
- **ls название_папки** — вывод содержимого каталога определенной папки на экран
- **mkdir название_папки** — создать папку с нужным вам наименованием
- **pwd** — вывод полного пути к текущему каталогу
- **rm -rf название_папки** — удалить папку «dirname» с её содержимым (опция -r) без предупреждения пользователя (опция -f)
- r**mdir название_папки** — удалить определенную папку

Помимо параметров скрипт может принимать опции — значения, состоящие из одной буквы, перед которыми пишется дефис. Они задаются после имени запускаемой программы, как и параметры. При необходимости можно сделать обработку опций командной строки по такому же принципу, как это делается с параметрами.

Как вы видите на примере **ls -l\ls-la**, некоторые опции можно совмещать.

```
sweety@sweetappletooth:~$ echo "blablabla" > failek.txt
sweety@sweetappletooth:~$ ls -ls
total 4
4 -rw-rw-r-- 1 sweety sweety 10 Mar 20 14:29 failek.txt
sweety@sweetappletooth:~$ ls -la
total 36
drwxr-xr-x 4 sweety sweety 4096 Mar 20 14:29 .
drwxr-xr-x 5 root   root   4096 Mar 20 13:14 ..
-rw-r--r-- 1 sweety sweety  220 Feb 25  2020 .bash_logout
-rw-r--r-- 1 sweety sweety 3771 Feb 25  2020 .bashrc
drwx------ 2 sweety sweety 4096 Mar 17 10:53 .cache
-rw-r--r-- 1 sweety sweety  807 Feb 25  2020 .profile
drwx------ 2 sweety sweety 4096 Mar 17 10:57 .ssh
-rw-r--r-- 1 sweety sweety    0 Mar 17 12:12 .sudo_as_admin_successful
-rw------- 1 sweety sweety 1430 Mar 20 13:09 .viminfo
-rw-rw-r-- 1 sweety sweety   10 Mar 20 14:29 failek.txt
sweety@sweetappletooth:~$ cat failek.txt
blablabla
sweety@sweetappletooth:~$
```

Для работы с файлами

- **./имя_файла** — запустить исполняемый файл
- **cat > имя_файла** — запись в файл
- **cat имя_файла** — чтение файла
- **cat file1 file2 > file3** — объединение файлов
- **cp file1 file2** — копировать файл «file1» с переименованием на «file2». При этом, если файл уже есть, то он просто перезапишется, с потерей
- **diff file1 file2** — сравнение файлов.
- **echo текст >> имя_файла** — дописать в файл текст
- **file имя_файла** — определение типа файла
- **find имя_файла** — поиск файла
- **grep текст имя_файла** — поиск и вывод строк из файла, содержащих «текст»
- **less имя_файла** — открыть файл в окне терминала
- **mcedit имя_файла** — редактирование файла
- **mv file1 file2** — переименовать файл «file1» в «file2»
- **mv имя_файла имя_папки** — переместить определенный файл в нужный вам каталог\папку
- **rm имя_файла** — удалить файл
- **sh имя_файла** — запустить файл со сценарием Bash
- **touch имя_файла** — создать файл

```
sweety@sweetappletooth:~$ pwd
/home/sweety
sweety@sweetappletooth:~$ mkdir netology_test
sweety@sweetappletooth:~$ cd netology_test
sweety@sweetappletooth:~/netology_test$ pwd
/home/sweety/netology_test
sweety@sweetappletooth:~/netology_test$ touch file1
sweety@sweetappletooth:~/netology_test$ ls -la
total 8
drwxrwxr-x 2 sweety sweety 4096 Sep  4 17:27 .
drwxr-xr-x 8 sweety sweety 4096 Sep  4 17:26 ..
-rw-rw-r-- 1 sweety sweety    0 Sep  4 17:27 file1
sweety@sweetappletooth:~/netology_test$ cp file1 file2
sweety@sweetappletooth:~/netology_test$ ls -la
total 8
drwxrwxr-x 2 sweety sweety 4096 Sep  4 17:28 .
drwxr-xr-x 8 sweety sweety 4096 Sep  4 17:26 ..
-rw-rw-r-- 1 sweety sweety    0 Sep  4 17:27 file1
-rw-rw-r-- 1 sweety sweety    0 Sep  4 17:28 file2
sweety@sweetappletooth:~/netology_test$ mv file1 file3
sweety@sweetappletooth:~/netology_test$ ls -la
total 8
drwxrwxr-x 2 sweety sweety 4096 Sep  4 17:28 .
drwxr-xr-x 8 sweety sweety 4096 Sep  4 17:26 ..
-rw-rw-r-- 1 sweety sweety    0 Sep  4 17:28 file2
-rw-rw-r-- 1 sweety sweety    0 Sep  4 17:27 file3
sweety@sweetappletooth:~/netology_test$ _
```

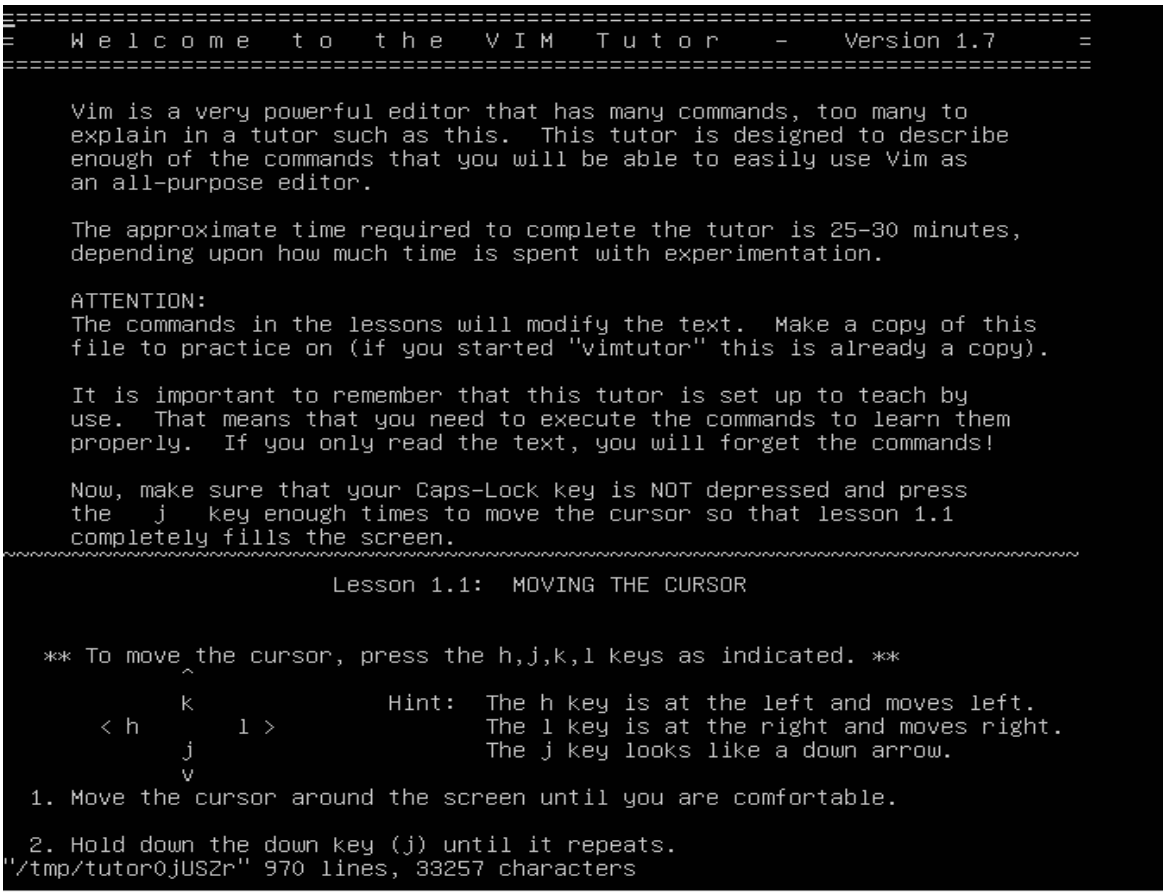
Также, можно пользоваться редакторами **Nano** и **Vim**.

Текстовый редактор **Nano** является стандартным консольным редактором для многих дистрибутивов Linux, рассмотрим некоторые его возможности:

- **nano имя_файла** - откроет определенный файл в редакторе nano. Если такого файла не существует, то создаст пустой файл с таким именем и откроет его
- **Alt+U** - отменить последнее действие
- **Alt+E** - повторить последнее действие еще раз
- **Alt+R** - поиск и замена
- **Ctrl+R** - вставляет данные из определенного файла в открытый
- **Ctrl+O** - сохраняет изменения в файле
- **Ctrl+X** - выход из редактора. Если заранее файл не сохранили, то nano предложит его сохранит

Для ознакомления с редактором **VIM**, можно пройти встроенный в него курс. Рекомендуем сразу же попрактиковаться, чтобы запомнить команды.

- **vimtutor** - запустить обучение редактору VIM



Символы

- # - комментарии
- * - всё (по аналогии с sql)

Операторы

- + - сложение
- - - вычитание
- / деление
- % остаток от деления
- ** возведение в степень

Синтаксис

Синтаксис команд **bash** — это расширенный синтаксис команд **Bourne shell**. Окончательная спецификация синтаксиса команд **bash** есть в **Bash Reference Manual**, который распространяется проектом GNU (свободная Unix-подобная операционная система, разрабатываемая Проектом GNU). Подробнее можно почитать тут.

Горячие клавиши для Bash

Управление запущенными процессами

- **Ctrl+C** - закрывает текущий процесс, запущенный в терминале
- **Ctrl+D** - закрывает bash оболочку вместе с процессами
- **Ctrl+Z** - сворачивает текущий процесс. Его можно возобновить с помощью команды fs имя+процесса

Управление экраном

- **Ctrl+L** - производит быструю очистку экрана
- **Ctrl+S** - останавливает весь вывод на экране
- **Ctrl+Q** - продолжает работу и отображение вывода после ее остановки

Управление курсором

- **Ctrl+A** или **Home** - курсор идет в начало строки
- **Ctrl+E** или **End** - курсор идет в конец строки
- **Alt+B** - курсор двигается к началу на одно слово
- **Ctrl+B** - курсор двигается к началу на один символ
- **Alt+F** - курсор двигается в конец на одно слово

- **Ctrl+F** - курсор двигается в конец на один символ
- **Ctrl+XX** - при повторном исполнении команды, курсор будет двигаться между началом линии и текущей позиции курсора

Удаление текста

- **Ctrl+D** или **Delete** - удаляет символ находящийся на курсоре
- **Alt+D** - удаляет все символы находящиеся после курсора на текущей линии
- **Ctrl+H** или **Backspace** - удаляет символ, находящийся перед курсором

Переставление

- **Alt+T** - переставляет текущее слово и предыдущее
- **Ctrl+T** - переставляет два предыдущих символа перед курсором
- **Ctrl+_** - делает отмену последней нажатой клавиши

Вырезка и вставка текста

- **Ctrl+W** - делает вырезку слова перед курсором и добавляет его в буфер
- **Ctrl+K** - вырезает весь текст после курсора и добавляет его в буфер
- **Ctrl+U** - вырезает весь текст перед курсором и добавляет его в буфер
- **Ctrl+Y** - вставляет последний контент из буфера

Регистр

- **Alt+U** - после курсора слово переводится в верхний регистр
- **Alt+L** - после курсора слово переводится в нижний регистр
- **Alt+C** - после курсора первый символ слова переводится в верхний регистр

Перебор команд из истории

- **Ctrl+P** или **Up Arrow** - показывает предыдущую введенную команду
- **Ctrl+N** или **Down Arrow** - показывает следующую введенную команду
- **Ctrl+R** - вызывает последнюю команду, которая соответствует введенной ключевой фразе
- **Ctrl+O** - запускает найденную команду в этом режиме
- **Ctrl+G** - выход из режима

Безопасность

Bashdoor и прочее

При определенных условиях, обнаруженная уязвимость позволяет запускать на атакуемом устройстве произвольный код, предварительно «пристегнув» его к любому безобидному скрипту. Учитывая то, что исполнение кода произойдет незаметно, теоретически злоумышленник может получить полный контроль над системой.

Что собой представляет уязвимость #Shellshock в #Bash и чем она опасна

На данный момент наиболее емко критичность бага описали специалисты из американской Компьютерной группы реагирования на чрезвычайные ситуации (CERT): «Эта уязвимость позволяет злоумышленникам передавать специально созданные переменные окружения, содержащие произвольные команды, которые могут выполняться на уязвимых системах. Это особенно опасно по причине того, что оболочка Bash очень распространена и может быть вызвана приложениями разными способами».

Кроме того, в CERT назвали степень проникновения уязвимости высокой (10 баллов), а сложность низкой. Последнее означает, что для эксплуатации бага не требуется сложных умений и большого опыта. В отличие, кстати, от пресловутой Heartbleed, для применения которой требовались весьма специфические навыки.

Как себя защитить?

Единственное, что вы можете сделать для защиты себя и своих устройств, помимо полного отказа от пользования Интернетом, — это регулярно обновлять все имеющееся у вас программное обеспечение.

Однако есть нюансы. Если в случае с операционными системами апдейт решает проблемы на всех компьютерах сразу, вне зависимости от их конфигурации, то с обновлением роутеров, модемов и прочих сетевых устройств ситуация иная. Очевидно, что универсального патча, который поправил бы ошибку во всех девайсах сразу, существовать не может, так что придется смиренно ждать, пока каждый производитель выпустит для своих устройств соответствующие “заплатки”, после чего устанавливать их вручную.

Еще одна проблема заключается в том, что ***nix-систем**, на которых установлена Bash, великое множество, и среди устройств, которые работают под их управлением, много тех, которые в принципе не могут быть обновлены.