ЛАБОРАТОРНАЯ РАБОТА №6

ИЗУЧЕНИЕ ОБОЛОЧКИ BASH

1. Операторы и спецсимволы оболочки. Оболочка *Bash* обеспечивает взаимодействие пользователя с операционной системой в текстовом режиме. По сути *Bash* — это командный интерпретатор. Пользователь должен ввести команду (обычно команда — это имя утилиты, которую мы хотим использовать) и ее аргументы (если они необходимы) в командную строку и нажать клавишу ENTER. По имени команды оболочка осуществляет поиск программного кода и его выполнение. По завершению выполнения команды (программы) оболочка возвращает *код завершения.* В случае корректного завершения оболочка возвращает значение 0, в случае ошибки возвращается ненулевое значение, которое интерпретируется как код ошибки.

В первых *UNIX* системах применялась оболочка *Shell*, одним из вариантов которой и является *Bash*. Основные задачи оболочки:

- обеспечение работы всех программ;
- организация ввода-вывода;
- работа с переменными окружения;
- организация конвейеров.

Bash имеет встроенный язык программирования, позволяющий реализовывать базовый набор операторов и составлять так называемые файлы сценариев (скрипты).

Оболочка *Bash* использует специальные символы, которые нельзя использовать в именах файлов. Рассмотрим некоторые из них.

Специальный символ «\» (обратный слэш). Его назначение — отмена специального значения любого из специальных символов, следующим за ним. Например, последовательность «*» отменяет действие спецсимвола «*».

Специальный символ «'» (одинарная кавычка) называют символом экранирования. Используется в паре со своей копией для обрамления какого-то выражения. **Все** спецсимволы внутри этого выражения теряют свои специальные значения.

Специальный символ «"» (двойная кавычка) называют символом экранирования. Используется в паре со своей копией для обрамления какого-то выражения. Действие аналогично действию одинарных кавычек, за исключением спецсимволов «$», «'» и «\» (знак доллара, одинарные кавычки и обратный слэш), которые не теряют своего специального значения.

Командная оболочка использует следующие **операторы**.

Оператор «;» применяется для отделения одной команды от другой, если они записаны в одной строке. Например:

```
main@main-VirtualBox:~/lab_6$ ls -l; ps -f
```

Сначала оболочка выполнит команду *ls -s*, дождется ее завершения (успешного или с ошибкой), а затем выполнит команду *ps -u*.

Оператор «&» используется для запуска выполнения команд в фоновом режиме. Сразу после запуска команды, в конце которой стоит символ «&», оболочка вернет управление пользователю независимо от того, завершилось или нет выполнение этой команды.

Оператор «&&» является управляющим оператором. Команда, следующая за этим оператором начнет свое выполнение только в том случае, если команда, стоящая перед оператором «&&», выполнится без ошибок. Например:

```
main@main-VirtualBox:~/lab_6$ ls -l abc && echo $HOME
ls: cannot access 'abc': No such file or directory
```

Поскольку выполнение команды *ls -l abc* завершилось с ошибкой, то команда *echo $HOME* не была выполнена.

Оператор «||» является управляющим оператором. Команда, следующая за этим оператором начнет свое выполнение только в том случае, если команда, стоящая перед оператором «||», вернет код ошибки. Например:

```
main@main-VirtualBox:~/lab_6$ ls -l abc || echo $HOME
ls: cannot access 'abc': No such file or directory
/home/main
```

2. Потоки ввода-вывода. Перенаправление. Практически все команды (утилиты) для своей работы требуют каких-то данных, вводимых пользователем с клавиатуры, вывод результатов своей деятельности на экран и вывод сообщений об ошибках, если такие случаются. В системах семейства *Linux* устройства ввода-вывода и все другие устройства представляются с помощью специальных файлов, которые располагаются в каталоге */dev*.

Для работы процесса могут потребоваться файлы, которые перед использованием необходимо открыть. С каждым **открытым** файлом связывается *дескриптор файла* — целое неотрицательное число, по которому система его (файл) идентифицирует. Первые три дескриптора (0,1,2) зарезервированы системой и привязаны к файлам */dev/stdin, /dev/stdout, /dev/stderr*, которые назначаются каждому процессу при его запуске. Они являются символическими ссылками, приводящими в итоге к файл-устройству, с которым работает интерпретатор:

```
main@main-VirtualBox:~/lab_6$ ls -l /dev | grep std
lrwxrwxrwx  1 root    root          15 окт 15 07:46 stderr -> /proc/self/fd/2
lrwxrwxrwx  1 root    root          15 окт 15 07:46 stdin -> /proc/self/fd/0
lrwxrwxrwx  1 root    root          15 окт 15 07:46 stdout -> /proc/self/fd/1
main@main-VirtualBox:~/lab_6$ ls -l /proc/self/fd
total 0
lrwx------ 1 main main 64 ноя  1 16:20 0 -> /dev/pts/7
lrwx------ 1 main main 64 ноя  1 16:20 1 -> /dev/pts/7
lrwx------ 1 main main 64 ноя  1 16:20 2 -> /dev/pts/7
```

Файл *stdin* называют *стандартным потоком ввода* оболочки. Ему назначен дескриптор с номером **0**, связан он с клавиатурой и **открыт на чтение**. Поэтому любая

команда, ожидающая ввод данных, по умолчанию ожидает его с клавиатуры.

Файл *stdout* называют *стандартным потоком вывода*. Ему назначен дескриптор с номером **1**, связан он с экраном и **открыт на запись**. По умолчанию любая команда, если она должна вывести данные, выводит их на экран.

Файл *stderr* называют *стандартным потоком ошибок*. Ему назначен дескриптор с номером **2**, связан он с экраном и **открыт на запись**. По умолчанию любая команда, выводящая сообщения об ошибках, выводит их на экран.

Стандартные потоки ввода-вывода можно переключать на файл или устройство. Это называется *перенаправлением потоков*. Оболочка *Bash* поддерживает следующие операции перенаправления: «>», «<», «>>».

Операция	**Действие**
команда [-опции] <аргументы> 1> <имя_файла>	Перенаправление стандартного потока вывода (дескриптор 1) в файл с именем *<имя_файла>*. Если указанного файла нет, то создается новый.
команда [-опции] <аргументы> 2> <имя_файла>	Перенаправление стандартного потока ошибок (дескриптор 2) в файл с именем *<имя_файла>*
команда [-опции] 0< <имя_файла>	Перенаправление стандартного потока ввода (дескриптор 0) из файла с именем *<имя_файла>*
команда [-опции] <аргументы> 1>> <имя_файла>	То же, что и операция «>», только данные добавляются в конец существующего файла

Когда происходит перенаправление вывода (операция «>») в некоторый файл, стандартный поток вывода закрывается, а его дескриптор освобождается и передается этому файлу. Например, команда *cat file 1> cat_out* производит считывание файла *file*, но результаты выводит не на экран, а сохраняет их в файл *cat_out*. По сути, файл *cat_out* является копией файла *file*.

Оболочка *Bash* позволяет перенаправлять несколько потоков данных в один поток. Конструкция *2 > &1* перенаправляет стандартный поток ошибок в стандартный поток вывода. Конструкция *1 > &2* перенаправляет стандартный поток вывода в поток ошибок. **Последовательность операций перенаправления потоков имеет значение!** Например, команды *1>&2 2>out_1* и *2>out_2 1>&2* дадут разные результаты.

Конструкция &> объединяет стандартные потоки вывода и ошибок в один поток и направляет его в файл.

3. Организация конвейеров. Стандартные потоки можно перенаправлять в программу, т. е. создавать *конвейер*. Рассмотрим простейший пример конвейера. Утилита *grep* предназначена для фильтрации полученной из стандартного ввода информации и вывода ее на экран:

```
main@main-VirtualBox:~/lab_6$ cat passwd_info | grep passwd
passwd(1)                  Пользовательские команды                  passwd(1)
       passwd - изменяет пароль пользователя
       passwd [параметры] [УЧЁТНАЯ_ЗАПИСЬ]
       Программа passwd изменяет пароли пользовательских учётных записей.
       passwd также изменяет информацию об учётной записи или срок действия
       момент. Если нет, то passwd не производит изменение пароля и завершает
       Не включайте системные символы стирания и удаления. Программа passwd не
       Параметры команды passwd:
       Команда passwd для аутентификации пользователей и для смены паролей
       /etc/passwd
       /etc/pam.d/passwd
           настройки PAM для passwd
       Программа passwd завершая работу, возвращает следующие значения:
           неожиданная ошибка при работе, отсутствует файл passwd
           файл passwd занят другой программой, попробуйте ещё раз
       chpasswd(8), passwd(5), shadow(5), usermod(8).
shadow-utils 4.2               _    05/16/2017                  passwd(1)
```

Стандартный поток вывода утилиты *cat* был перенаправлен командой «|» на стандартный поток ввода утилиты *grep*, которая вывела на экран только строки, содержащие слово «*passwd*». Более сложный пример:

```
main@main-VirtualBox:~/lab_6$ cat passwd_info | grep passwd 1>passwd_filter
```

Стандартный выходной поток утилиты *grep* перенаправлен в файл *passwd_filter*.

Еще один пример — конвейер из трех утилит:

```
main@main-VirtualBox:~/lab_6$ cat passwd_info | grep passwd | grep /etc
       /etc/passwd
       /etc/pam.d/passwd
```

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ №6

0. Изучить теоретический материал.
1. Авторизоваться в консоли №1.
2. Создать каталог *~/lab_6*, в котором создать пустой файл, в имени которого присутствуют два не следующие друг за другом пробела.
3. Запустить две любые команды (например, *ls -l, ps -u, time, date, id, pwd*) в формате *команда_1 <оператор> команда_2*. При этом следует рассмотреть следующие случаи:

Код завершения команды_1	Код завершения команды_2	Оператор
0	0	«;»
Число, отличное от 0	0	«||»

Число, отличное от 0	0	«&&»
Число, отличное от 0	Число, отличное от 0	«||»

3.1 Результаты пояснить.

3.2 **Примечания**. Для получения кода завершения команды «число, отличное от 0» достаточно ввести неверные опции или аргументы для данной команды. Код завершения 0 соответствует правильно введенным опциям и аргументам.

4. Выполнить команду *ls -li ~/lab_6,* перенаправив стандартный поток вывода в файл *~/lab6/ls_out*. Результат просмотреть.
5. Перенаправить стандартный поток ошибок команды *pwd -h* в файл *~/lab_6/pwd_err*. Результат просмотреть и объяснить.
6. Сохранить руководство по утилите *passwd* в файл *~/lab_6/passwd_info*.
7. Добавить в конец файла *passwd_info* распечатанный на текущий месяц календарь (команда *cal*).
8. Перенаправить стандартный вывод команды *man cat* в файл, специально предназначенный для уничтожения данных (*/dev/null*).
9. Перенаправить стандартный поток ошибок любой команды в стандартный поток вывода, который в свою очередь перенаправить в файл *~/lab_6/<имя_команды>_out*.
10. *Находясь в каталоге *~/lab_6* выполнить команды:

10.1 *ls -l abc 1>ls_abc_1 2>&1;*

10.2 *ls -l 1>ls_1 2>&1;*

10.3 *ls -l abc 2>&1 1>ls_abc_2*;

10.4 *ls -l 1>ls_2 2>&1;*

10.5 Просмотреть содержимое файлов *ls_abc_1, ls_abc_2, ls_1, ls_2*. Результаты объяснить.

11. Для любой утилиты объединить стандартные потоки вывода и ошибок в один поток и перенаправить его в файл *~/lab_6/<имя_утилиты>_in_out*.
12. *Находясь в терминале №1 (*tty1*) выполнить команду *man groups,* выходной поток которой перенаправить на терминал №2 (*tty2*).
13. Разобрать работу следующих конвейеров:

13.1 *who | wc -l*

13.2 *cut -d: -f1,3 /etc/passwd | tail -8*

13.3 *cut -c2-7 /etc/passwd | tail -8*

13.4 **Примечание**. С утилитами *cut, tail, who, wc* можно ознакомиться на соответствующих страницах справочника.

14. *Вывести на экран список всех имен файлов из каталога */etc,* в которых содержится строка *conf*.
15. Ответить на контрольные вопросы:
    15.1 Что называют оболочкой Bash?
    15.2 Какие операторы оболочки вы знаете?
    15.3 Что называю дескриптором открытого файла?
    15.4 Какие файлы по умолчанию всегда связаны с процессом?
    15.5 Что такое стандартные потоки ввода, вывода и ошибок?
    15.6 Опишите действие операторов перенаправления стандартных потоков.
    15.7 Что называют конвейером?