Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Уральский федеральный университет имени первого Президента России Б.Н. Ельцина» (УрФУ)

Институт радиоэлектроники и информационных технологий – РтФ

Школа профессионального и академического образования

Отчет по дисциплине   
«Операционные системы»

Лабораторная работа №1  
«Базовые приемы работы в терминале ОС Linux»

Студенты: Клоченко И.Е., Юсупов Д.А.

Преподаватель: Коротяев А.Н.

Группа: РИ-300024

Екатеринбург

2023

**Оглавление**

[1. **Создание пользователя** 3](#_Toc129823498)

[2. **Переключение терминала** 3](#_Toc129823499)

[3. **Регистрация пользователя** 4](#_Toc129823500)

[4. **Работа с командой printenv** 4](#_Toc129823501)

[5. **Работа с файлом сценария** 9](#_Toc129823502)

[6. **Работа с командой pwd** 11](#_Toc129823503)

[7. **Работа с историей команд** 13](#_Toc129823504)

[8. **Работа с потоками** 15](#_Toc129823505)

[9. **Конвейеры в ОС Linux.** 16](#_Toc129823506)

[10. **Способы объединения команд** 17](#_Toc129823507)

[11. **Работа с переменными окружения. Присвоение переменной окружения строки, состоящей из двух имеющихся переменных и спецсимвола.** 19](#_Toc129823508)

[12. **Работа с переменным окружения. Арифметика.** 20](#_Toc129823509)

[13. **Служебные символы, используемые в шаблонах.** 21](#_Toc129823510)

**Цель работы:** знакомство с ОС Linux, получение практических навыков работы с командной строкой ОС Linux на примере изучения выводов команд *printenv*, работы с правами доступа (*chmod*), создания сценариев оболочки (.sh), работы с переменными окружения и регулярными выражениями.

**Ход работы:**

Комментарий: Лабораторная работа выполнена в CentOS 7, установленной на виртуальной машине Virtual Box. Перед выполнением лабораторной работы была произведена небольшая настройка терминала: установлена и настроена оболочка *zsh*. Также пользователем по умолчанию установлен пользователь с логином *iankie*.

# **Создание пользователя**

Для создания нового пользователя ***user01*** с паролем ***123234*** была использована следующая команда: *sudo useradd user01 -p 123234*.

******

**Рисунок 1**. Создание пользователя user01

Здесь:

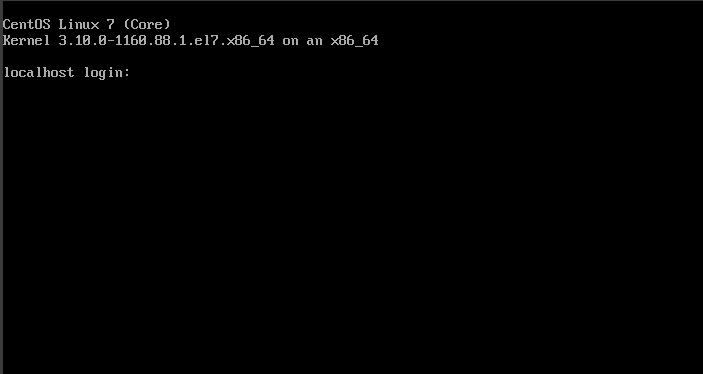
*sudo* – часть команды, которая выполняет следующую за ней команду с правами суперпользователя.

*useradd* – команда для добавления пользователя

Параметр ­*-p* используется для указания пароля для пользователя.

# **Переключение терминала**

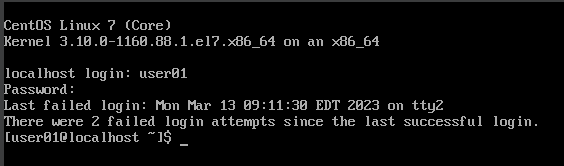
Производим переключение терминала на ***tty2*** с помощью комбинации клавиш *Host (Right CTRL) + F2*.



**Рисунок 2**. Переключение терминала на tty2

# **Регистрация пользователя**

Для регистрации пользователя *user01* в системе был произведен ввод указанных ранее данных (имя пользователя и пароль):



**Рисунок 3**. Регистрация пользователя user01

# **Работа с командой printenv**

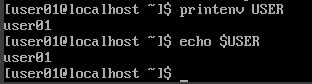
Результат выполнения команды *printenv*:



**Рисунок 4**. Выполнение команды printenv

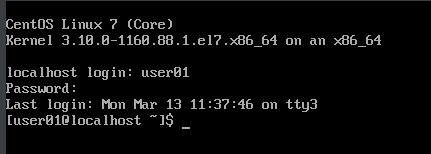
Команда выводит в терминал список переменных окружения (переменных среды).

С помощью команды *printenv* можно получить значение указанной переменной. К примеру, если ввести в терминале команду *printenv USER* (или *echo $USER*), в результате будет получено имя пользователя, зарегистрированного в текущей сессии:

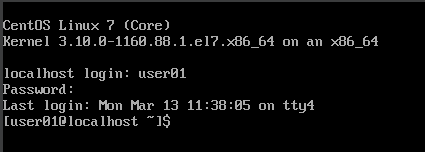


**Рисунок 5**. Пример вызова переменной окружения USER

Регистрация пользователя *user01* в консолях *tty3* и *tty4* производится аналогично:

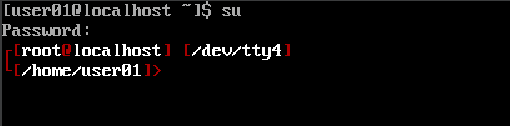


**Рисунок 6**. Регистрация пользователя user01 в консоли tty3



**Рисунок 7**. Регистрация пользователя user01 в консоли tty4

В консоли *tty3* выполним последовательно команды *su* и *printenv*:

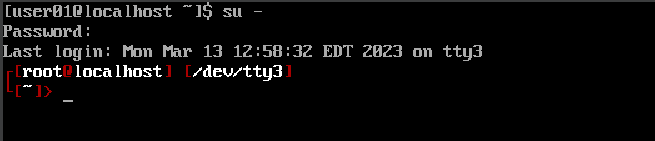


**Рисунок 8**. Выполнение команды su

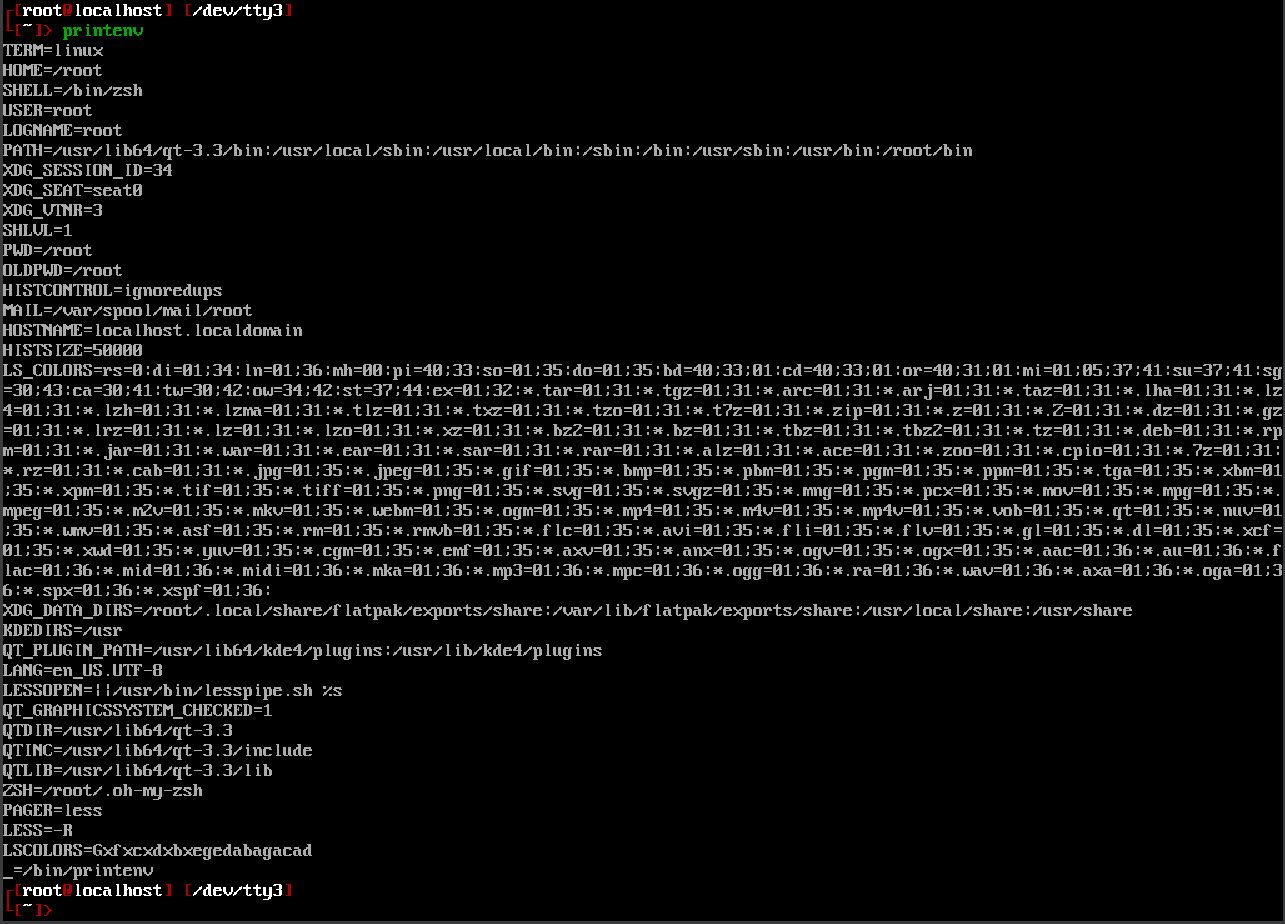


**Рисунок 9**. Выполнение команды printenv

В консоли *tty4* выполним последовательно выполним команды *su -* и *printenv:*



**Рисунок 10**. Выполнение команды su -



**Рисунок 11**. Выполнение команды printenv

Сравним результаты, полученные в п.п. 3,6 и 7. Основные отличия представлены в виде таблицы (Таблица 1).

**Таблица 1.** Сравнение результатов команд

|  |  |  |  |
| --- | --- | --- | --- |
| **Основные отличия** | **Результат п.4** | **Результат п.6** | **Результат п.7** |
| Домашняя папка | */home/user01/* | */root/* | */root/* |
| Текущая директория после выполнения команды (*su* (*su -*)) | */home/user01/* | */home/user01/* | */root/* |
| Оболочка консоли | */bin/bash/* | */bin/zsh/* | */bin/zsh/* |
| SHLVL (Уровень «погружения») | 1 | 2 | 1 |
| HISTSIZE (Количество хранимых строк) | 1000 | 50000 | 50000 |

Разница в результатах в том, что:

* Команда *su* меняет лишь текущего пользователя, оставляя переменные среды такими, какими они были при регистрации пользователя user01.
* Команда *su –* производит процесс регистрации пользователя root (суперпользователя) и, следовательно, меняет почти все переменные среды для текущей сессии.
* Когда команды выполняются без предшествующих *su* или *su –* ничего «необычного» не происходит. П.4 служит скорее для сравнения с п.6 и п.7.

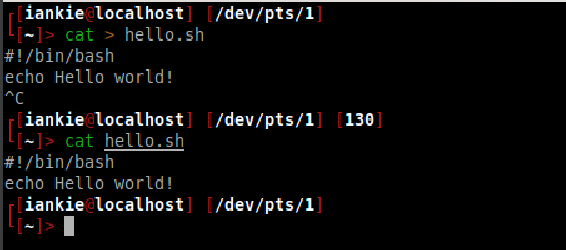
# **Работа с файлом сценария**

Создадим в домашнем каталоге тестовый сценарий *hello.sh*, используя команду *cat*:



**Рисунок 12**. Создание сценария hello.sh

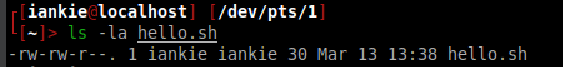
После запуска введенной команды, система будет ожидать ввода содержимого файла. Введем последовательно следующие строки и сохраним файл:



**Рисунок 13**. Завершение создания сценария hello.sh

Здесь же мы отобразили содержимое файла (команда *cat hello.sh*) и убедились, что введенные выше строки совпадают с содержимым файла.

Выполним команду *ls -la hello.sh*:



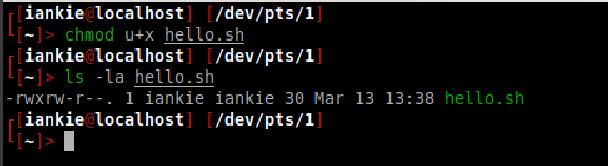
**Рисунок 14**. Результат выполнения команды ls -la hello.sh

Команда *ls* выводит список файлов и папок в текущей директории.

Флаг *-l* модифицирует вывод в строчный вывод (в одной строке один файл).

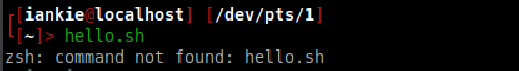
Флаг *-a* модифицирует вывод в вывод полной информации о всех файлах в текущей директории (в данном случае файла *hello.sh*).

Изменим права на файл текущему пользователю (добавим право на исполнение) командой *chmod u+x hello.sh:*



**Рисунок 15**. Изменение прав файлу hello.sh

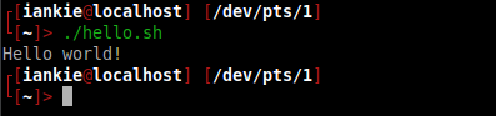
Запустим файл *hello.sh:*



**Рисунок 16**. Попытка запуска скрипта hello.sh

Во время попытки запуска файла, произошла ошибка, которая дословно значит, что не найдена указанная команда. Вывод: нужна команда для запуска скрипта.

Запустим команду, изменив ее вид:

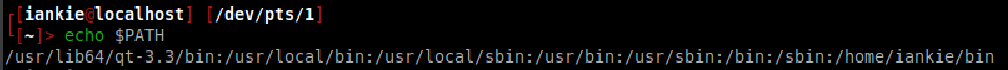


**Рисунок 17**. Запуск скрипта hello.sh

В этом случае скрипт удачно запустился и выполнил команды, которые мы сообщили ему ранее:

* *#!/bin/bash* – использовать оболочку *bash*
* *echo Hello world!*  - вывести в консоль строку «Hello world!».

Выполним команду *echo $PATH*:

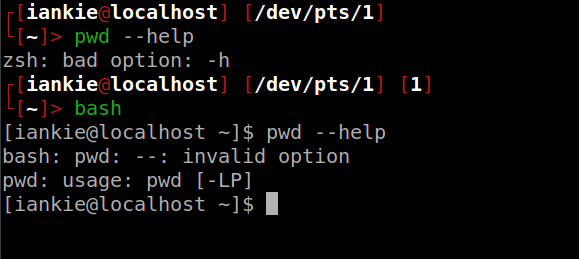


**Рисунок 18**. Вывод команды echo $PATH

Данная команда выводит значение переменной среды *PATH* - пути, где оболочке нужно находить исполняемые файлы для запуска без *./*. Из-за отсутствия пути */home/iankie* запуск сценария был невозможен в первом случае.

# **Работа с командой pwd**

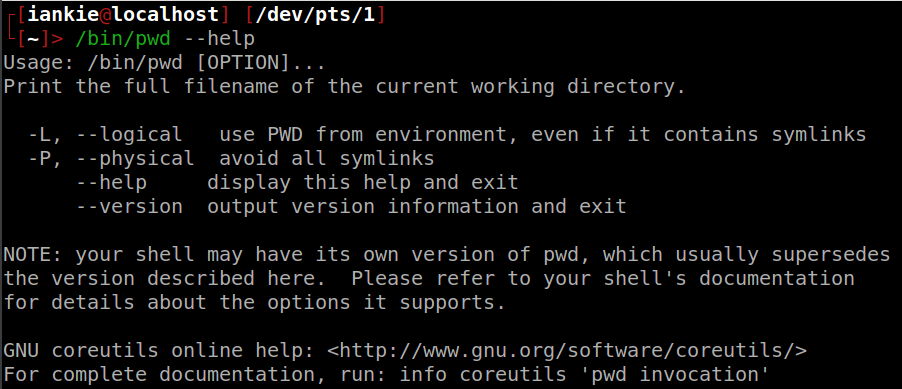
Выполним команду *pwd –help*:



**Рисунок 19**. Результат выполнения команды pwd --help

Результат выполнения команды – ошибка. Команда была проверена на выполнение в двух оболочках: основной *zsh* и *bash*. Ошибка: неверный аргумент (*bash* подсказывает как правильно использовать команду и ее параметры).

Теперь попытаемся выполнить эту команду с указанием полного пути до нее:



**Рисунок 20**.. Результат выполнения команды /bin/pwd –help

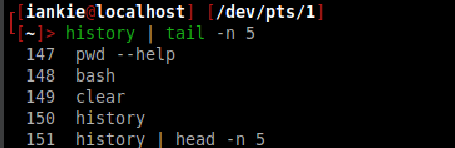
В случае выполнения этой команды, все проходит в штатном режиме: результат команды – вывод *help*-страницы команды *pwd.*

Разный вывод этих команд обусловлен тем, что в первом случае pwd — это встроенная команда оболочки, а во втором случае – производится запуск исполняемого файла напрямую. Чтобы получить желаемый результат от команды *pwd --help*, можно ввести следующие команды: *man pwd*, *info pwd*.

# **Работа с историей команд**

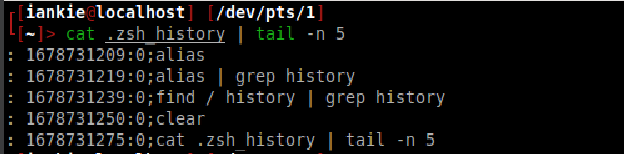
Историю команд в ОС Linux можно:

1. Выполнив команду *history*:



**Рисунок 21**. Выполнение команды history

1. Вывести содержимое файла оболочки с суффиксом *\_history* (в нашем случаем это файл *.zsh\_hictory*):

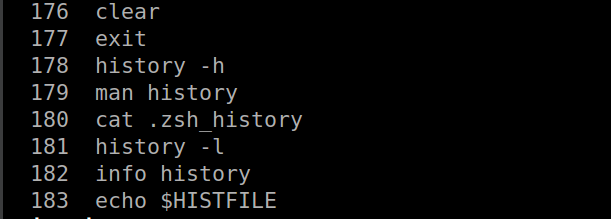


**Рисунок 22**. Содержимое файла .zsh\_history

Основные способы работы с историей в оболочке *zsh*:

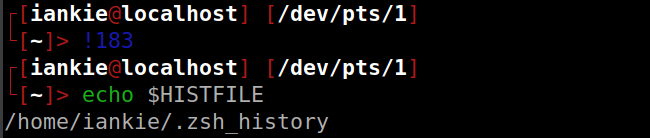
1. Использование «![номер команды]»

При вызове команды *history* можно увидеть номера команд в левом столбце. Эти номера можно использовать.



**Рисунок 23**. Несколько команд, полученных командой history

К примеру, чтобы вызвать команду *echo $HISTFILE* (команда, которая выводит название файла оболочки, в котором хранится вся история команд, в истории находится под номером 183), нужно ввести следующую команду:



**Рисунок 24**. Использование ! для работы с историей команд

После ввода *!183* оболочка в новой строке выведет предложенную команду, после чего можно нажать Enter и получить вывод.

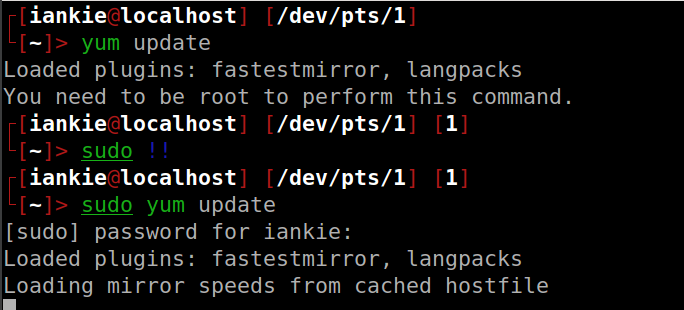
1. Использование «*!!»*

Можно использовать *!!* чтобы обратиться к предыдущей команде:



**Рисунок 25**. Использование !!

В основном это используется для подстановки приставки *sudo* к предыдущей команде после получения ошибки типа *Permission denied:*



**Рисунок 26**. Использование sudo !!

# **Работа с потоками**

В CentOS 7 существует 3 основных потока ввода/вывода:

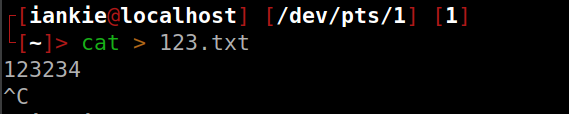
1. Поток ввода *stdin.* Ввод с клавиатуры.
2. Поток вывода *stdout*. Вывод в консоль.
3. Поток ошибок *stderr*. Вывод сообщений об ошибках в консоль. Также выводит отладочную информацию из программ в консоль.

Перенаправление потоков ввода/вывода:

Перенаправление в ОС Linux производится с помощью операторов перенаправления (< - перенаправление ввода, > - перенаправление вывода (использовалось в связке с командой *cat* для создания сценария оболочки hello.sh) и >> - добавление вывода в конец файла).

Примеры:

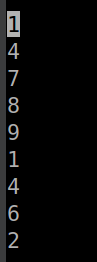
Создадим файл *123.txt* c текстом *123234* с помощью перенаправления вывода:



**Рисунок 27**. Пример перенаправления вывода

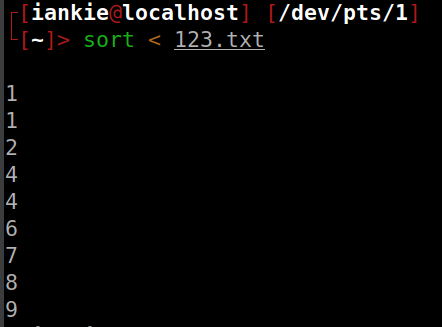
Таким же образом можно записать в файл выполнение любой команды.

Немного изменим файл 123.txt для демонстрации следующего примера:



**Рисунок 28**. Измененное содержимое файла 123.txt

Теперь воспользуемся оператором перенаправления ввода, чтобы отсортировать содержимое файла *123.txt*:



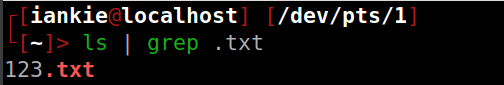
**Рисунок 29**. Использование оператора перенаправления ввода

# **Конвейеры в ОС Linux.**

Конвейеры предназначены для передачи вывода одной команды на вход другой команды. Для этого служит символ *|* (прямая черта).

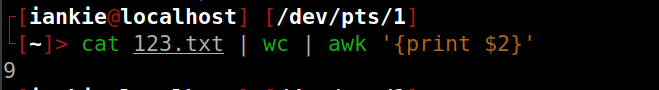
Примеры использования:

Найдем все файлы с расширением .txt в текущей директории:



**Рисунок 30**. Использование конвейеров

Выведем количество слов в файле 123.txt используя конвейеры:



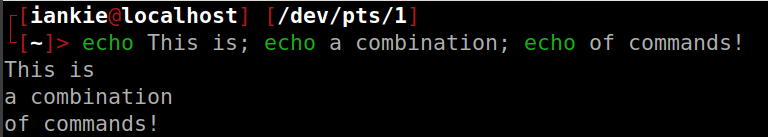
**Рисунок 31**. Использование конвейеров

Вывод команды cat *123.txt* передается на вход команды *wc*, которая считает количества строк, слов и байт в файле. Далее второй прямой чертой мы передаем вывод этой команды команде *awk*, которая обработает вывод предыдущей команды и выведет в консоль только количество слов в файле *123.txt.*

# **Способы объединения команд**

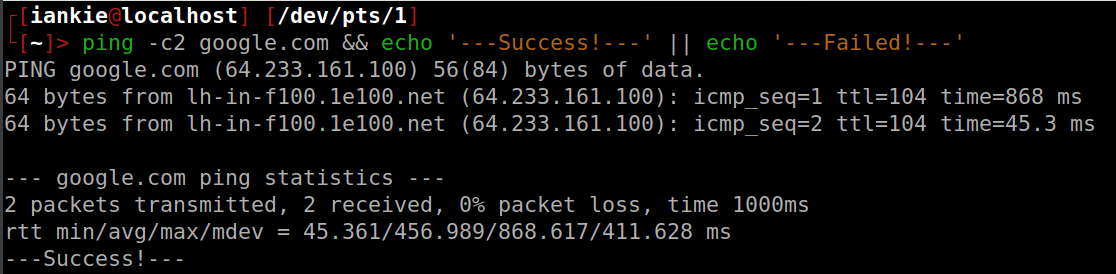
## Последовательное объединение команд.

Объединить команды последовательно можно с помощью «;», например:



**Рисунок 32**. Пример последовательного объединения команд

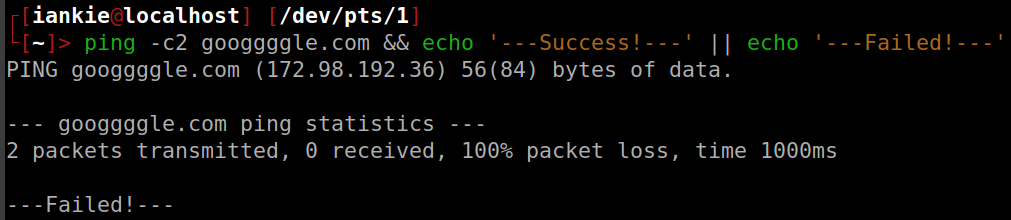
Также последовательное объединение можно выполнить с помощью логических операторов && и/или ||:



**Рисунок 33**. Использование логических операторов && и ||

Логический оператор && требует успешного выполнения предыдущей команды и только при выполнении этого условия запускает следующую после него команду (в нашем случае это *echo*).

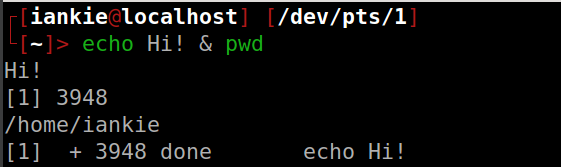
Логический оператор || запускает следующую после него команду только в том случае, если основная команда выдала код завершения отличный от успешного:



**Рисунок 34**. Пример работы логического оператора ||

## Параллельное объединение команд

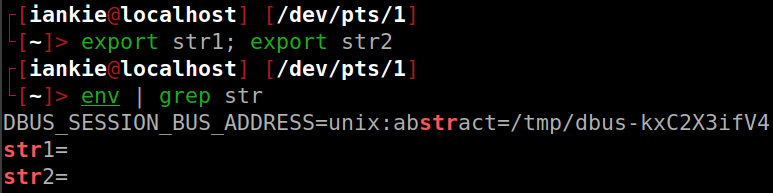
Объединить команды параллельно – значит заставить работать их одновременно, не зависимо друг от друга. Эта независимая работа команд может быть достигнута с помощью амперсанда (&):



**Рисунок 35**. Пример использования & для параллельного объединения команд

# **Работа с переменными окружения. Присвоение переменной окружения строки, состоящей из двух имеющихся переменных и спецсимвола.**

Создадим переменные окружения *str1* и *str2*:



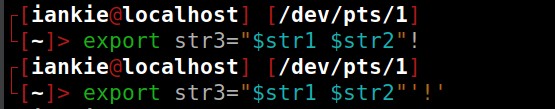
**Рисунок 36**. Создание переменных окружения str1 и str2

Присвоим им значения «*Hello*» и «*world*» соответственно:



**Рисунок 37**. Присвоение значений переменным str1 и str2

Создадим переменную *str3* таким же образом, что и *str1-2* и построим выражение таким образом, чтобы str3 приняло значение «Hello world!»:

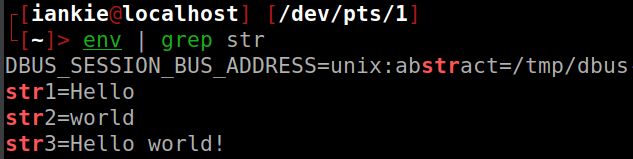


**Рисунок 38**. Присвоение значения переменной str3

Здесь важно обратить внимание на то, что символ «!» — это специальный символ, который в оболочке имеет свое значение (обращение к истории команд), поэтому его нужно экранировать (на рисунке 38 продемонстрированы 2 способа экранирования символа «!»).

Также важно, что обращение к переменным происходит путем указания перед ними символа «$».

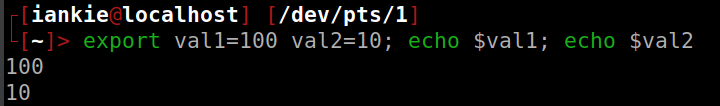
Результат:



**Рисунок 39**. Результат работы с переменной str3

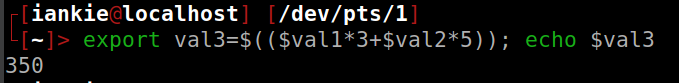
# **Работа с переменным окружения. Арифметика.**

Определим две переменные окружения *val1* и *val2*,и присвоим значения 100 и 10 соответственно:



**Рисунок 40**. Инициализация переменных val1=100 и val2=10

Инициализируем переменную val3, значение которой будет результат выполнения арифметического выражения *val1*\*3+*val2*\*5:



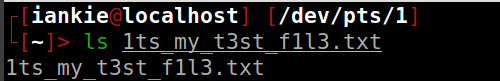
**Рисунок 41**. Инициализация переменной val3

Чтобы решить арифметическое выражение в консоли, нужно использовать конструкцию *$((выражение))*. Тогда спецсимволы, используемые для написания этого выражения будут интерпретированы оболочкой верно.

# **Служебные символы, используемые в шаблонах.**

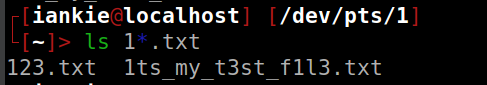
Часто, чтобы облегчить работу в консоли используются шаблоны (наиболее часто с этим можно столкнуться при работе с файлами), составленные из специальных символов:

Для примеров был создан следующий пустой файл:



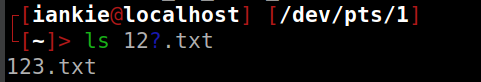
**Рисунок 42**. Файл для примеров

«\*» - соответствует любому количеству произвольных символов (также символов может и не быть совсем).



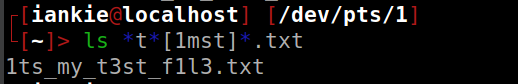
**Рисунок 43**. Пример использования символа «\*»

«?» - соответствует одному любому символу.



**Рисунок 44**. Пример использования символа «?»

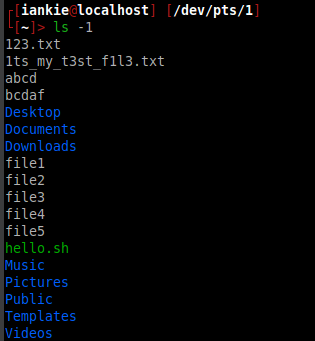
«[» и «]» - (используются именно в такой последовательности и соответствуют одному символу) – необходимы для указания диапазона или набора символов, которые могут находиться на указанной позиции.



**Рисунок 45**. Пример использования символов «[» и «]»

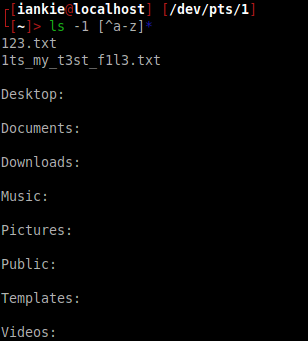
«^» используется в регулярных выражениях для определения начала строки или для отрицания.

В домашней директории есть следующие файлы:



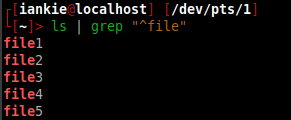
**Рисунок 46.** Файлы в домашней директории

Выполним команду, которая выведет в домашней директории файлы, которые не начинаются с букв в диапазоне *[a-z]:*



**Рисунок 47.** Использование символа «^» в шаблоне для отрицания

Теперь выполним команду, которая выдаст все файлы, которые начинаются с «file»:



**Рисунок 48.** Использование символа «^» в шаблоне для задания начала строки

# **Вывод:**

В ходе выполнения лабораторной работы поставленные цели были успешно выполнены. Произведена успешная установка и настройка виртуальной машины в среде VirtualBox, а также установка дистрибутива CentOS 7. Были освоены методы работы с терминалами в ОС Linux, получены навыки работы с основными командами, утилитами, переменными окружения и регулярными выражениями. Также были освоены основы работы со сценариями оболочки bash, работа с файловой системой (а именно правами доступа для разрешения запуска исполняемых файлов .sh), а также получены навыки работы с суперпользователем, конвейерами и потоками ввода/вывода: (stdin - ввод с клавиатуры, stdout - вывод в консоль, stderr - вывод сообщений об ошибках в консоль). Были разобраны варианты последовательного и параллельного объединения команд.