```
---
## Front matter
title: "Отчет о прохождении 3 этапа внешних курсов"
subtitle: "Продвинутые темы"
author: "Гольцова Мария, НММбд-01-23"

## Generic otions
lang: ru-RU
toc-title: "Содержание"

## Bibliography
bibliography: bib/cite.bib
csl: pandoc/csl/gost-r-7-0-5-2008-numeric.csl

## Pdf output format
toc: true # Table of contents
toc-depth: 2
lof: true # List of figures
lot: true # List of tables
fontsize: 12pt
linestretch: 1.5
papersize: a4
documentclass: scrreprt
## I18n polyglossia
polyglossia-lang:
  name: russian
  options:
      - spelling=modern
      - babelshorthands=true
polyglossia-otherlangs:
  name: english
## I18n babel
babel-lang: russian
babel-otherlangs: english
## Fonts
mainfont: PT Serif
romanfont: PT Serif
sansfont: PT Sans
monofont: PT Mono
mainfontoptions: Ligatures=TeX
romanfontoptions: Ligatures=TeX
sansfontoptions: Ligatures=TeX,Scale=MatchLowercase
monofontoptions: Scale=MatchLowercase,Scale=0.9
## Biblatex
biblatex: true
biblio-style: "gost-numeric"
biblatexoptions:
  - parentracker=true
  - backend=biber
  - hyperref=auto
  - language=auto
  - autolang=other*
  - citestyle=gost-numeric
## Pandoc-crossref LaTeX customization
figureTitle: "Рис."
tableTitle: "Таблица"
listingTitle: "Листинг"
lofTitle: "Список иллюстраций"
lotTitle: "Список таблиц"
```

```
lolTitle: "Листинги"
## Misc options
indent: true
header-includes:
  - \usepackage{indentfirst}
  - \usepackage{float} # keep figures where there are in the text
  - \floatplacement{figure}{H} # keep figures where there are in the text
---
```

Цель работы

Ознакомиться с функционалом операционной системы Linux.

Задание

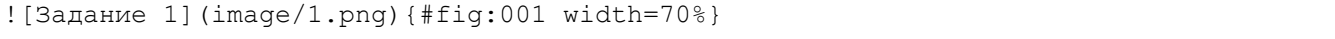
Просмотреть видео и на основе полученной информации пройти тестовые задания.

Теоретическое введение

Линукс - в части случаев GNU/Linux — семейство Unix-подобных операционных систем на базе ядра Linux, включающих тот или иной набор утилит и программ проекта GNU, и, возможно, другие компоненты. Как и ядро Linux, системы на его основе, как правило, создаются и распространяются в соответствии с моделью разработки свободного и открытого программного обеспечения. Linux-системы распространяются в основном бесплатно в виде различных дистрибутивов — в форме, готовой для установки и удобной для сопровождения и обновлений, — и имеющих свой набор системных и прикладных компонентов, как свободных, так и проприетарных.

Выполнение лабораторной работы

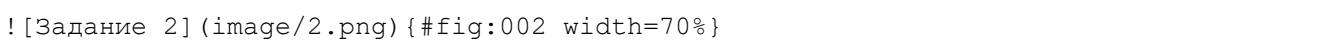
3 Этап: (рис. @fig:001, @fig:002, @fig:003, @fig:004, @fig:005, @fig:006, @fig:007, @fig:008, @fig:009, @fig:010, @fig:011, @fig:012, @fig:013, @fig:014, @fig:015, @fig:016, @fig:017, @fig:018, @fig:019, @fig:020, @fig:021, @fig:022, @fig:023, @fig:024, @fig:025, @fig:026, @fig:027, @fig:028, @fig:029, @fig:030, @fig:031, @fig:032, @fig:033, @fig:034, @fig:035, @fig:036, @fig:037, @fig:038, @fig:039, @fig:040, @fig:041).


```
![Задание 1](image/1.png){#fig:001 width=70%}
```

Стоит упомянуть, что у редактора vim есть туториал, который позволяет разобраться с командами, необходимыми для стандартной работы.
За выход из редактора отвечают следующие команды:

- ZQ - выйти без сохранения
- :q! - выйти без сохранения
- ZZ - записать файл и выйти (если файл не изменяли, то записываться он не будет)
- :wq - записать файл и выйти
- :x - записать файл и выйти
- :w<CR> - записать файл
- :sav filename<CR> - "сохранить как"
- :w filename<CR> - "сохранить как"
- :w!<CR> - записать файл

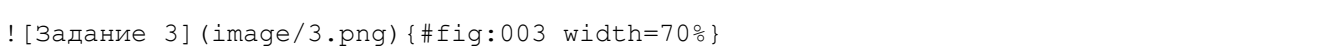
Как мы видим, вариантов много, при этом каждый сможет найти тот, который подойдёт под конкретную ситуацию.


```
![Задание 2](image/2.png){#fig:002 width=70%}
```

`Strange_  TEXT  is_here. 2=2 YES!`

Точка считается "маленьким словом", так что всего их 9:
`Strange_`, `is_here`, `.`, `2`, `=`, `2`, `!` и `два лишних пробела`.

И если посчитать нажатия на w и на W, то действительно после 10 штук попадем в одно место. 10 нажатий на W, это то же самое, что и 10 нажатий на w,

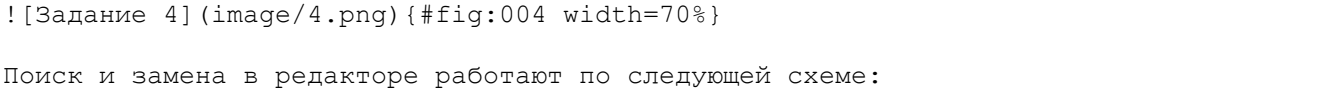


`d2wwifour four <<Esc>>`

`d2wwywPp`

`d2w$$bifour four <<Esc>>`

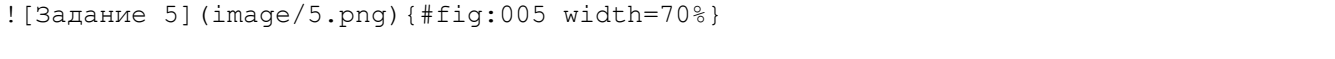
- $ — в конец текущей строки;
- w — на слово вправо;
- b — на слово влево;
- i — начать ввод перед курсором;
- p — вставка содержимого неименнованного буфера под курсором;
- P — вставка содержимого неименованного буфера перед курсором;
- yy (также Y) — копирование текущей строки в неименованный буфер;
- y<число>y — копирование числа строк начиная с текущей в неименованный буфер;



Поиск и замена в редакторе работают по следующей схеме:

`:{пределы}s/{что заменяем}/{на что заменяем}/{опции}`

Для замены во всем файле можно использовать символ %.



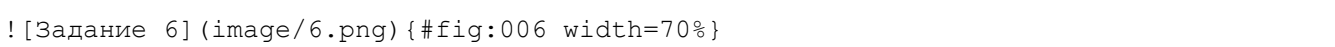
Команда $ — в конец текущей строки, W - до пробела вправо - то есть, перемещение.

Нажать Esc достаточно один раз, но да ладно.

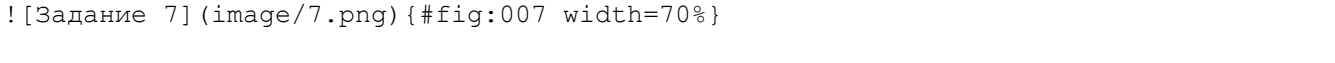
Надпись visual - горит.

d — используется совместно с командами перемещения. Удаляет символы с текущего положения курсора до положения после ввода команды перемещения.

yy (также Y) — копирование текущей строки в буфер;



Только из набора C потому что у каждой оболочки свой буфер, который при выходе из нее буде записываться в файл истории.



`/home/bi/file1.txt` - потому что именно в этой директории мы создаем новый файл, а уже после его создания мы переходим в другую папку.

![Задание 8](image/8.png){#fig:008 width=70%}

Имя не может начинаться с цифры, содержать специальные символы или пробелы.

![Задание 9](image/9.png){#fig:009 width=70%}

`$ echo опции строка`
Эта команда печатает строки, которые передаются в качестве аргументов в стандартный вывод и обычно используется в сценариях оболочки для отображения сообщения или вывода результатов других команд.

`var1=$1` - обозначение переменных

`var2=$2`

`echo "Arguments are: \$1=$var1 \$2=$var2"` - строка печати.

![Задание 10](image/10.png){#fig:010 width=70%}

- $0 - имя скрипта
- $# - вернет количество аргументов
- -ge - больше или равно
- -n <string> - не пустая строка.

Имя скрипта - это не пустая строка.

$# Это число аргументов без учета имени скрипта, который всегда $0. И число аргументов всегда будет или равно нулю, или больше него, тк просто не может скатиться в отрицательную сторону.

![Задание 11](image/11.png){#fig:011 width=70%}

- -lt, (<) - меньше
- -gt - больше
- -eq - равно

3 не больше 5, 3 не меньше 3, 3 не равно 4.

5 не больше 5, 5 не меньше 3, 5 не равно 4.

Оба раза выведет four.

![Задание 12](image/11_1.png){#fig:012 width=70%}

1. Задаю общую часть в каждом выводе - слово "student": v=student
2. Выполняем команды для разных аргументов.
3. res - это результат для вывода
4. echo "$res" - вывести результат

![Задание 13](image/12.png){#fig:013 width=70%}

- (Start)
- a > c нет (Finish)
- (Start)
- , > c нет (Finish)
- (Start)

- b > c нет (Finish)
- (Start)
- , > c нет (Finish)
- (Start)
- c_d > c да


```
child=16
adult=25
stdout=0

while [[ $stdout != 1 ]] #конструкция типа while-True
    do
        echo "enter your name: " #Пользователь вводит имя
        read name
    if [[ (-z $name) || ($name = 0) ]] ;then #Если имя не по параметрам, простимся
        echo "bye"
        stdout=1
    elif [[ -n $name ]]; then #А вот если имя нормальное
        while [[ $stdout != 1 ]] ;do
            echo "enter your age: " #То пусть вводит возраст
            read age #Считываем возраст
            if [[ ($age -eq 0) || (-z $age) ]] ;then #Если возраст 0 или строка
пуста - прощаемся
                echo "bye"
                stdout=1
            elif [[ $age -le $child ]] ;then #Если меньше или равен ребенку, то
ребенок
                echo "$name, your group is child"
            elif [[ $age -gt $adult ]] ; then #Больше взрослого - то взрослый
                echo "$name, your group is adult" ;else
                if [[ ($age -ge 17) && ($age -le 25) ]] ;then #Если от 17 до 25, то
подросток.
                    echo "$name, your group is youth" ;fi
            fi ;break
        done ;fi
done
```

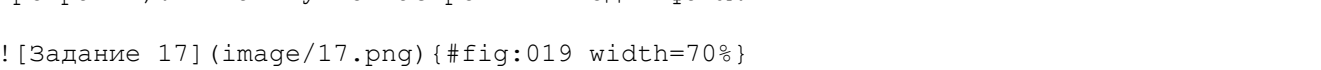

1. a = $a

2. a += b это то же самое, что и a = a + b, но с символами "+=" != "=+"

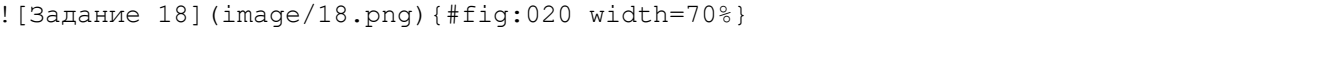
3. если выражение не в скобках, но с пробелами - работать не будет. (let a=a+b - сработает; let a = a + b - нет)

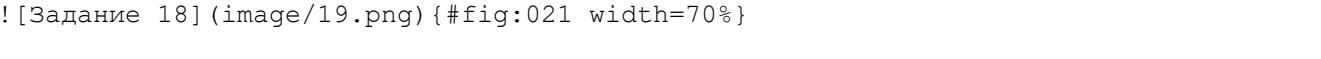
Выведет путь до директории, в которую мы перешли, так как "```pwd```" - это команда

`programm` выполняет стандартный вывод в терминал (если это принцип работы программы). И нам нужно настроить вывод в файл.



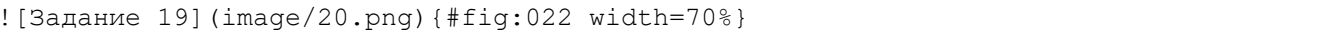
Первая переменная локальная, и это просто пустая строка, вторая переменная - это сумма арифметической прогрессии от 1 до 10, равна 55, но при умножении на 2 даст 110.

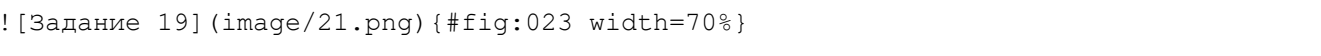




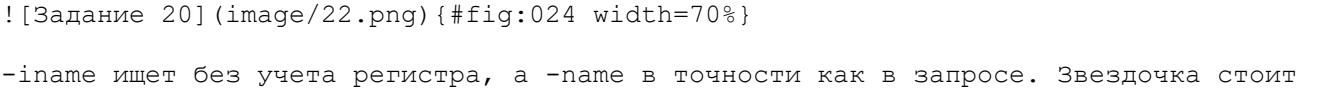
Алгоритм нахождения НОД делением

1. Большее число делим на меньшее.
2. Если делится без остатка, то меньшее число и есть НОД (следует выйти из цикла).
3. Если есть остаток, то большее число заменяем на остаток от деления.
4. Переходим к пункту 1.

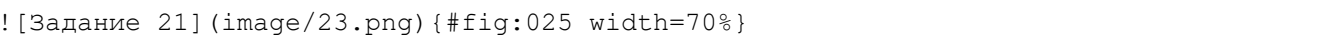




Калькулятор выглядит обычно - мы вводим два числа, пишем, что с ними надо сделать, и потом, учитывая случаи ошибок, выводим результат.



-iname ищет без учета регистра, а -name в точности как в запросе. Звездочка стоит после слова - это значит после слова может быть сколько угодно символов.

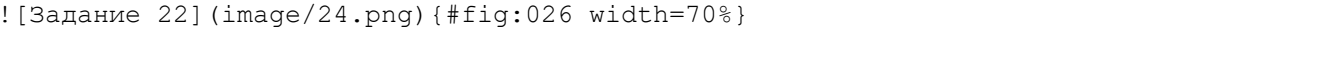


find [path] [expression]

где: path - это путь к директории, в которой нужно выполнить поиск файлов (по умолчанию, поиск производится в текущей директории и всех ее поддиректориях);

expression - это выражение, которое определяет критерии поиска файлов.

-name: поиск файлов по имени. Например: find /home/user -name myfile.txt



Текущий каталог - это depth=1, а остальное считается просто:

/home/bi   -> depth=1

/home/bi/dir1   -> depth=2

/home/bi/dir1/dir2   -> depth=3

![Задание 23](image/25.png){#fig:027 width=70%}

Из описания man:
Print NUM lines of trailing context after/before matching lines
"matching lines" - множественное число, строки в которых нашлось совпадение

Т.е. если идут 2...10...100 строк подряд, в которых обнаружилось совпадение, контекст будет выведен до и после этой ГРУППЫ строк, а не до и после каждой строки в этой группе

![Задание 24](image/26.png){#fig:028 width=70%}

![Задание 24](image/26_2.png){#fig:029 width=70%}

Объяснение на втором скриншоте.

![Задание 25](image/27.png){#fig:030 width=70%}

The -n option disables the automatic printing, which means the lines you don't specifically tell it to print do not get printed, and lines you do explicitly tell it to print (e.g. with p) get printed only once.

![Задание 26](image/28.png){#fig:031 width=70%}

аббревиатура ABBA отличается от двух других аббревиатур тем, что справа он неё стоит запятая без пробела: "ABBA,".

При этом по условию аббревиатура должна выглядеть как [XX] или [XXX] (и ещё больше X). Следовательно, для этой проверки надо добавить пробел квадратными скобками [ ] слева и, соответственно, с права.

![Задание 27](image/29.png){#fig:032 width=70%}

-persist lets plot windows survive after main gnuplot program exits.

![Задание 28](image/30.png){#fig:033 width=70%}

`plot 'data.csv' using 1:2` даст ошибку:

`warning: Skipping data file with no valid points ^ x range is invalid`

Скорее всего причиной такого поведения является тот факт, что формат CSV содержит строки, где столбцы разделены запятой? Содержимое файла:

```
1,21
2,22
3,23
4,24
5,25
6,26
7,27
8,28
9,29
10,30
```

![Задание 29](image/31.png){#fig:034 width=70%}

Сначала идет команда установки подписей, а потом в скобках:

подпись - пробел - переменная с координатой - запятая

Повторяется это количество раз соответствующее числу переменных, и без запятой (в случае с последней переменной)

А подпись в свою очередь получается конкатенацией текста из задания и переменной с координатой.

![Задание 30](image/32.png){#fig:035 width=70%}

1. График строится строкой "splot x**2+y**2".
2. Вращение задается строкой "zrot=(zrot+10)%360". Значит, смещение вперед (которое было изначально) можно также задать строкой "zrot=(zrot+360+10)%360" или иначе говоря "zrot=(zrot+370)%360". А теперь посмотрим на наше требование - чтоб вращалось в другую сторону, значит, по аналогии, необходимо вместо перебора на 10 сделать недобор.

"zrot=(zrot+350)%360"

3. Строка "pause 0.2" ставит выполнение на паузу на определенный промежуток времени. В задании сказали перерисовывать чаще, значит пауза должна быть меньше.

![Задание 31](image/33.png){#fig:036 width=70%}

- r - чтение;
- w - запись;
- x - выполнение;
- s - выполнение от имени суперпользователя (дополнительный);

- u - владелец файла;
- g - группа файла;
- o - все остальные пользователи;

- 0 - никаких прав;
- 1 - только выполнение;
- 2 - только запись;
- 3 - выполнение и запись;
- 4 - только чтение;
- 5 - чтение и выполнение;
- 6 - чтение и запись;
- 7 - чтение запись и выполнение.

![Задание 32](image/34.png){#fig:037 width=70%}

Решений два типа:

- Сменить права гостей, добавив W
- Сделать владельцем нужную группу или пользователя, в зависимости от того, у кого из них уже есть права на W
- Помнить, что root - владелец и остальные для него - others.

![Задание 33](image/35.png){#fig:038 width=70%}

- wc -l <filename> вывести количество строк
- wc -c <filename> вывести количество байт
- wc -m <filename> вывести количество символов
- wc -L <filename> вывести длину самой длинной строки
- wc -w <filename> вывести количество слов

-h, --human-readable
    print sizes in human readable format (e.g., 1K 234M 2G)

-s, --summarize
    display only a total for each argument

Команда создаст три директории от dir1 до dir3.

Сертификат

Выводы

Я просмотрел курс и освежил в памяти навыки работы с более сложными командами в Линукс.

Список литературы{.unnumbered}

1. Введение в Linux

::: {#refs}
:::