# Выполнение внешнего курса. Блок №3. Продвинутые темы

Отчёт

Сергеев Даниил Олегович

# Содержание

| 1  | Цель работы       |  |                            |  |  |  |  |  |  |  |  |  |  |
|----|-------------------|--|----------------------------|--|--|--|--|--|--|--|--|--|--|
| 2  | Задание           |  |                            |  |  |  |  |  |  |  |  |  |  |
| 3  | 3.1               | олнение лабораторной работы Текстовый редактор vim | 8<br>8<br>12               |  |  |  |  |  |  |  |  |  |  |
|    | 3.3<br>3.4<br>3.5 | Скрипты на bash: ветвления и циклы                 | 14<br>22<br>32<br>36<br>39 |  |  |  |  |  |  |  |  |  |  |
| 4  | Выв               | од   | 43                         |  |  |  |  |  |  |  |  |  |  |
| Сп | исок              | литературы   | 44                         |  |  |  |  |  |  |  |  |  |  |

# Список иллюстраций

| 3.1  | Задание №1. Условие и ответ    | 8  |
|------|--------------------------------|----|
| 3.2  | Задание №2. Условие и ответ    | 9  |
| 3.3  | Задание №3. Условие и ответ    | 10 |
| 3.4  | Задание №4. Условие и ответ    | 11 |
| 3.5  | Задание №5. Условие и ответ    | 11 |
| 3.6  | Задание №6. Условие и ответ    | 12 |
| 3.7  | Задание №7. Условие и ответ    | 13 |
| 3.8  | Задание №8. Условие и ответ    | 13 |
| 3.9  | Задание №9. Условие и ответ    | 14 |
| 3.10 | ) Задание №10. Условие и ответ | 15 |
| 3.11 | . Задание №11. Условие и ответ | 16 |
| 3.12 | 2 Задание №12. Условие         | 17 |
| 3.13 | 3 Задание №12. Ответ           | 17 |
| 3.14 | ∃Задание №13. Условие и ответ  | 19 |
| 3.15 | 5 Задание №14. Условие         | 20 |
| 3.16 | 5 Задание №14. Ответ           | 21 |
| 3.17 | ′ Задание №15. Условие и ответ | 23 |
| 3.18 | В Задание №16. Условие и ответ | 24 |
| 3.19 | 9 Задание №17. Условие и ответ | 24 |
| 3.20 | ) Задание №18. Условие и ответ | 25 |
| 3.21 | . Задание №19. Условие         | 26 |
| 3.22 | 2 Задание №19. Ответ           | 27 |
|      | 3 Задание №20. Условие         | 29 |
| 3.24 | ∃Задание №20. Ответ            | 30 |
| 3.25 | 5 Задание №21. Условие и ответ | 32 |
| 3.26 | . Задание №22. Условие и ответ | 32 |
| 3.27 | ′ Задание №23. Условие и ответ | 33 |
| 3.28 | В Задание №24. Условие и ответ | 34 |
| 3.29 | 9 Задание №25. Условие и ответ | 34 |
| 3.30 | ) Задание №26. Условие и ответ | 35 |
| 3.31 | . Задание №27. Условие и ответ | 36 |
| 3.32 | 2 Задание №28. Условие и ответ | 37 |
| 3.33 | 3 Задание №29. Условие и ответ | 37 |
| 3.34 | Задание №30. Условие и ответ   | 38 |
| 3.35 | . Задание №31. Условие и ответ | 39 |
| 3.36 | Задание №31. Файл              | 39 |
| 3.37 | 7 Задание №32. Условие и ответ | 40 |

| 3.38 Задание №33. Условие         |  |  | <br> | • |  |  |  |  | • |  | 40 |
|-----------------------------------|--|--|------|---|--|--|--|--|---|--|----|
| 3.39 Задание №33. Ответ           |  |  |      |   |  |  |  |  |   |  | 41 |
| 3.40 Задание №34. Условие и ответ |  |  |      |   |  |  |  |  |   |  | 41 |
| 3.41 Задание №35. Условие и ответ |  |  |      |   |  |  |  |  |   |  | 42 |
| 3.42 Задание №36. Условие и ответ |  |  | <br> |   |  |  |  |  |   |  | 42 |

# Список таблиц

# 1 Цель работы

Получить основные навыки работы с ОС Linux путем прохождения внешнего курса на образовательной платформе stepik [1].

## 2 Задание

- Пройти курс.
- Получить сертификат.
- Записать видео по каждому разделу.
- Записать итоговую презентацию по каждому этапу.
- Написать отчёт по прохождению контрольных мероприятий по каждому разделу.

# 3 Выполнение лабораторной работы

Приступим к выполнению третьего блока заданий внешнего курса – Продвинутым темам.

## 3.1 Текстовый редактор vim

1. Вопрос 1-й: Из только что открытого в vim файла можно выйти перейдя в командный режим с помощью : и написав q, затем ENTER. (если не производилось изменение файла)

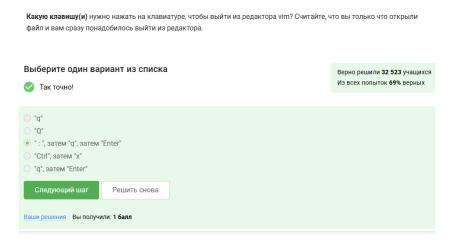


Рис. 3.1: Задание №1. Условие и ответ

2. Вопрос 2-й: Введем тестовую строку и проведем на ней указанные операции, выберем правильные ответы.

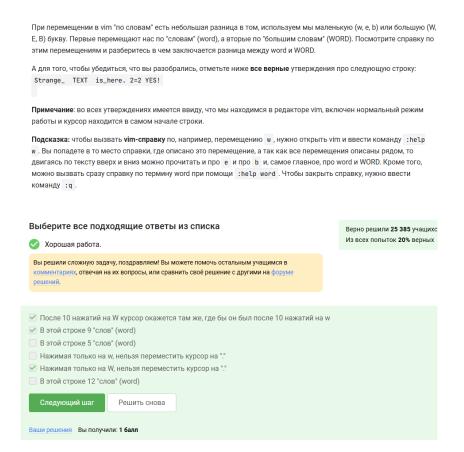


Рис. 3.2: Задание №2. Условие и ответ

3. Вопрос 3-й: Проверим каждую комбинацию в редакторе vim. Укажем правильные ответы.

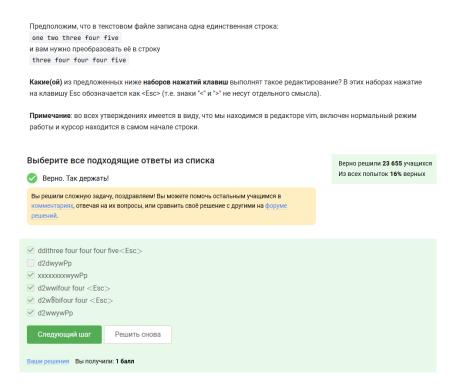


Рис. 3.3: Задание №3. Условие и ответ

- 4. Вопрос 4-й: В командном режиме можно совершить замену первого вхождения слова в строке. Для этого укажем строку :%s/Windows/Linux, где:
  - % диапазон (всего файла)
  - s команда замены
  - Windows то что заменяем
  - Linux то на что заменяем

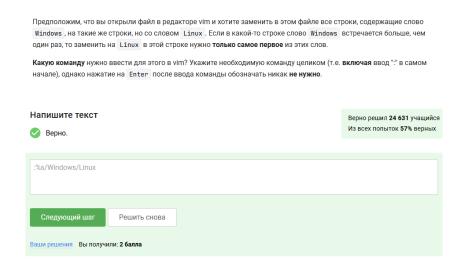


Рис. 3.4: Задание №4. Условие и ответ

5. Вопрос 5-й: Проверим каждую опцию из предложенного списка, выберем правильные ответы.

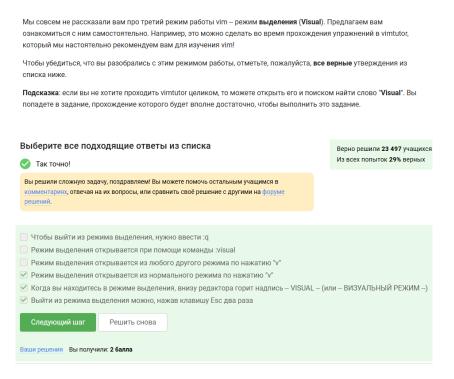


Рис. 3.5: Задание №5. Условие и ответ

## 3.2 Скрипты на bash: основы

1. Вопрос 6-й: Каждая оболочка хранит свойфайл независимо, поэтому в последней оболочке будут лишь команды С1, С2, С3.



Рис. 3.6: Задание №6. Условие и ответ

2. Вопрос 7-й: Исходя из скрипта, мы перейдем в домашнюю директорию, после в ней создастся файл file1.txt и команда отправит нас в каталог /home/bi/Desktop. В итоге выбираем второй вариант.

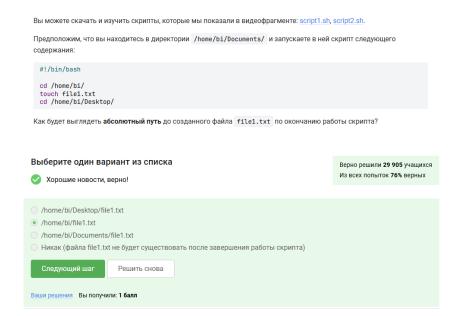


Рис. 3.7: Задание №7. Условие и ответ

3. Вопрос 8-й: В языке программирования bash в названия переменных допускаются нижние подчёркивания, числа и буквы, однако имена не должны начинаться с цифры. Выберем подходящие варианты ответа.

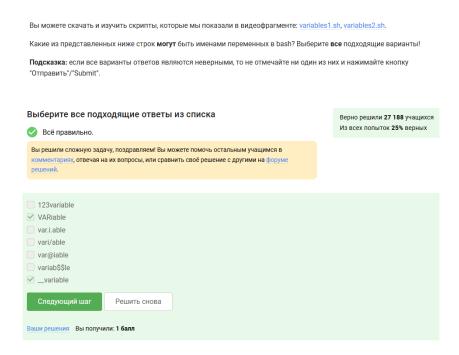


Рис. 3.8: Задание №8. Условие и ответ

4. Вопрос 9-й: Используем команду echo для вывода сообщения. Чтобы символ доллара корректно отображался, укажем обратный слеш перед ним: ...\\$1....

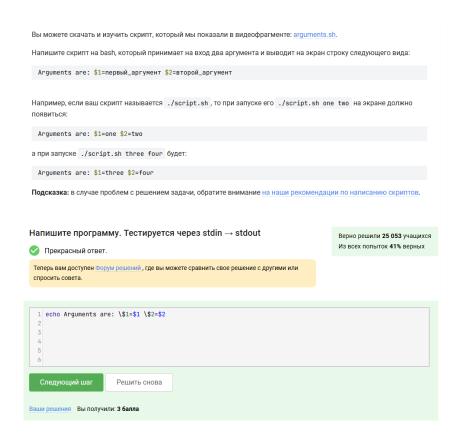


Рис. 3.9: Задание №9. Условие и ответ

Листинг №3.2.1. Команда вывода сообщения.

echo Arguments are: \\$1=\$1 \\$2=\$2

## 3.3 Скрипты на bash: ветвления и циклы

 Вопрос 10-й: Двойные квадратные скобки – усовершенствованная версия тестовой конструкции test, позволяющая совершать сравнения и тесты.
 Изучим возможные опции для этой конструкции и выберем подходящие варианты.

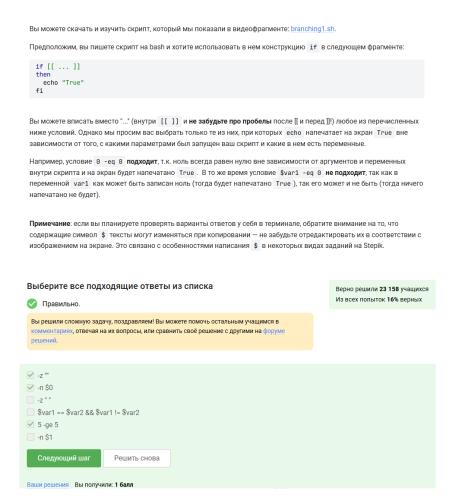


Рис. 3.10: Задание №10. Условие и ответ

2. Вопрос 11-й: Задание проверяет знание опций для [[]]. Пройдемся по условиям с заданными параметрами и выберем правильные варианты из списка.

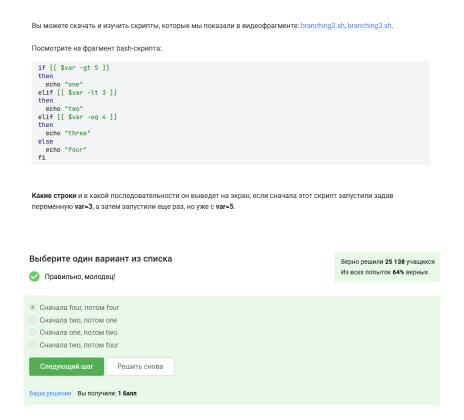


Рис. 3.11: Задание №11. Условие и ответ

3. Вопрос 12-й: Будем считывать первый аргумент с помощью \$1. Для проходки по возможным вариантам используем оператор case.

```
Напишите скрипт на bash, который принимает на вход один аргумент (целое число от 0 до бесконечности), который
будет обозначать число студентов в аудитории. В зависимости от значения числа нужно вывести разные сообщения.
Соответствие входа и выхода должно быть таким:
  0 --> No students
 1 --> 1 student
2 --> 2 students
3 --> 3 students
4 --> 4 students
  5 и больше --> A lot of students
Примечание а): выводить нужно только строку справа, т.е. "->" выводить не нужно.
Примечание 6): в последней строке слово "lot" с маленькой буквы!
Примечание 2: в этой и всех последующих задачах на написание скриптов, если не указано явно, что нужно проверять
вход (например, что он будет именно числом и именно от 0 до бесконечности), то этого делать не нужно!
Пример №1: если ваш скрипт называется ./script.sh , то при запуске его как ./script.sh 1 на экране должно
появиться:
1 student
Пример №2: если ваш скрипт называется ./script.sh , то при запуске его как ./script.sh 5 на экране должно
A lot of students
Подсказка: в случае проблем с решением задачи, обратите внимание на наши рекомендации по написанию скриптов.
```

Рис. 3.12: Задание №12. Условие

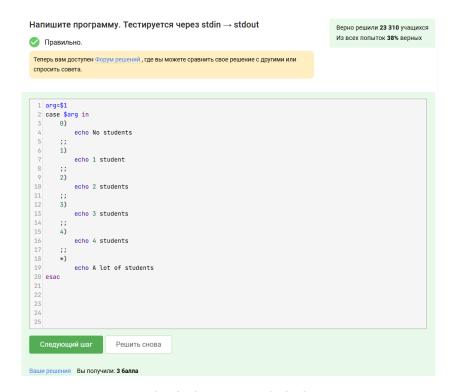


Рис. 3.13: Задание №12. Ответ

### Листинг №3.3.1. Скрипт проверки количества студентов.

arg=\$1

```
case $arg in
    0)
        echo No students
    ;;
    1)
        echo 1 student
    ;;
    2)
        echo 2 students
    ;;
    3)
        echo 3 students
    ;;
    4)
        echo 4 students
    ;;
    *)
        echo A lot of students
```

### esac

4. Вопрос 13-й: Команда запуститься 5 раз, так как знак запятой тоже считается отдельным значением переменной str. Условие [[ str > "c" ]] проверяет по длине строк, поэтому команда закончится на последнем аргументе, не выводя сообщение finish.

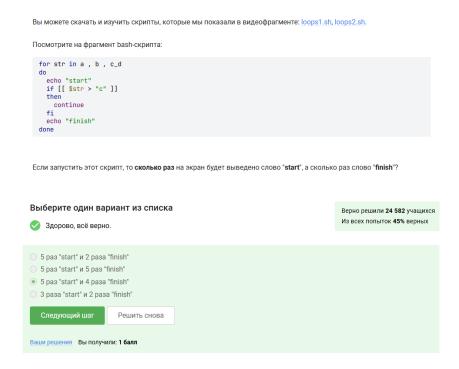


Рис. 3.14: Задание №13. Условие и ответ

5. Вопрос 14-й: Программу запишем в цикл while true. Сделаем отдельные переменные для имени и возраста, считаем значения через команду read, проверяя на пустую строку либо нулевой возраст.

Напишите скрипт на bash, который будет определять в какую возрастную группу попадают пользователи. При запуске скрипт должен вывести сообщение "enter your name:" и ждать от пользователя ввода имени (используйте read, чтобы прочитать его). Когда имя введено, то скрипт должен написать "enter your age:" и ждать ввода возраста (опять нужен read). Когда возраст введен, скрипт пишет на экран "// MMs>, your group is <rpynna>", где <rpynna>" определяется на основе возраста по следующим правилам:

- младше либо равно 16: "**child**",
- от 17 до 25 (включительно): "**youth**",
- старше 25: "adult".

После этого скрипт опять выводит сообщение **"enter your name:"** и всё начинается по новой (бесконечный цикл!). Если в какой-то момент работы скрипта будет введено **пустое имя** или **возраст 0**, то скрипт должен написать на экран **"bye"** и закончить свою работу (выход из цикла!).

Примеры корректной работы скрипта:

Nº1

```
./script.sh
enter your name:
Egor
enter your age:
16
Egor, your group is child
enter your name:
Elena
enter your age:
0
bye
```

#### Nº2:

```
./script.sh
enter your name:
ELena Petrovna
enter your age:
25
ELena Petrovna, your group is youth
enter your name:
bye
```

Рис. 3.15: Задание №14. Условие

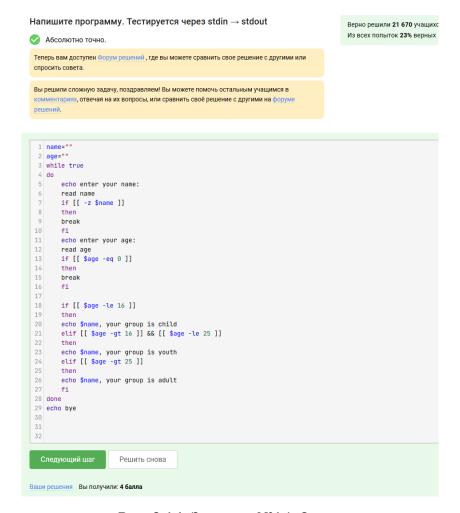


Рис. 3.16: Задание №14. Ответ

### Листинг №3.3.2. Скрипт проверки возраста.

```
name=""
age=""
while true
do
    echo enter your name:
    read name
    if [[ -z $name ]]
    then
    break
```

```
fi
    echo enter your age:
    read age
    if [[ $age -eq 0 ]]
    then
    break
    fi
   if [[ $age -le 16 ]]
    then
    echo $name, your group is child
    elif [[ $age -gt 16 ]] && [[ $age -le 25 ]]
    then
    echo $name, your group is youth
   elif [[ $age -gt 25 ]]
    then
    echo $name, your group is adult
    fi
done
echo bye
```

## 3.4 Скрипты на bash: разное

1. Вопрос 15-й: Выражения с командой let можно писать как в кавычках, так и без них. Во втором варианте происходит сложение (объединение) строк, а первый вариант не сработает из-за неправильного операнда.

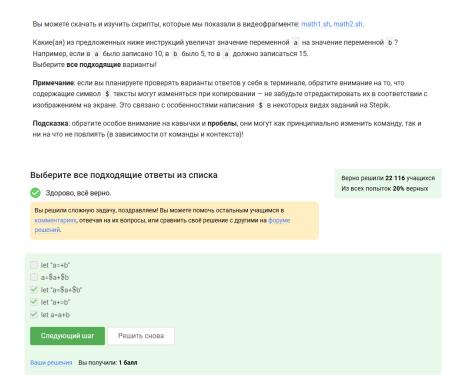


Рис. 3.17: Задание №15. Условие и ответ

2. Вопрос 16-й: Обратные одинарные ковычки позволяют выполнять команды внутри сценария оболочки, поэтому вывод будет равен пути pwd, указанном в файле.

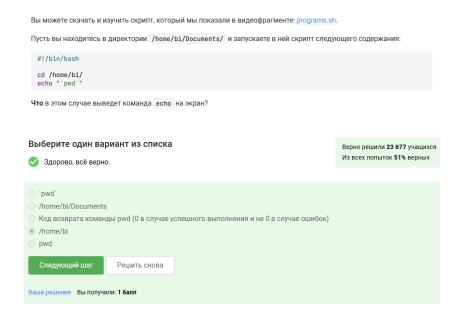


Рис. 3.18: Задание №16. Условие и ответ

3. Вопрос 17-й: Достать вывод потока ошибок можно через встроенную переменную \$? либо через запись результата в какой-нибудь файл с последующим считыванием кода выхода уже этого условия.

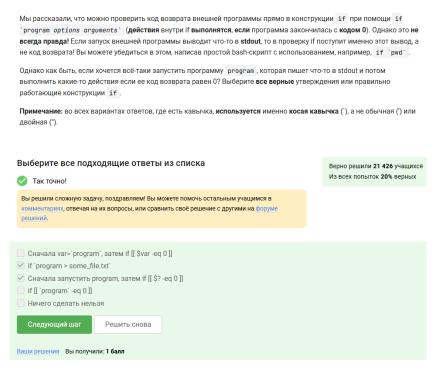


Рис. 3.19: Задание №17. Условие и ответ

4. Вопрос 18-й: Подвох этой задачи заключается в том, что мы пытаемся вывести значение переменной \$c1, хотя она является локальной в теле функции counter. На выходе её значение будет равно пустой строке.

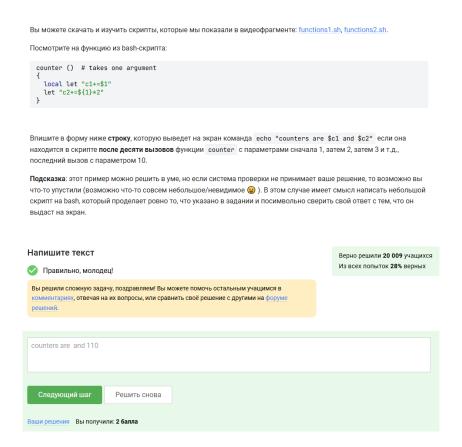


Рис. 3.20: Задание №18. Условие и ответ

5. Вопрос 19-й: Напишем программу для вычисления НОД, используя алгоритм Евклида и рекурсивный вызов функции в bash.

Напишите скрипт на bash, который будет искать наибольший общий делитель (НОД, greatest common divisor, GCD) двух чисел. При запуске ваш скрипт не должен ничего писать на экран, а просто ждет ввода двух натуральных чисел через пробел (для этого можно использовать read и указать ему две переменные – см. пример в видеофрагменте). После ввода чисел скрипт считает их НОД и выводит на экран сообщение "GCD is «посчитанное значение»", например, для чисел 15 и 25 это будет "GCD is 5". После этого скрипт опять входит в режим ожидания двух натуральных чисел. Если в какой-то момент работы пользователь ввел вместо этого пустую строку, то нужно написать на экран "bye" и закончить свою работу.

Вычисление НОД несложно реализовать с помощью алгоритма Евклида. Вам нужно написать функцию gcd , которая принимает на вход два аргумента (назовем их M и N). Если аргументы равны, то мы нашли НОД – он равен M (или N), нужно выводить соответствующее сообщение на экран (см. выше). Иначе нужно сравнить аргументы между собой. Если М больше N, то запускаем ту же функцию gcd , но в качестве первого аргумента передаем (M-N), а в качестве второго N. Если же наоборот, М меньше N, то запускаем функцию gcd с первым аргументом M, а вторым (N-M).

Пример корректной работы скрипта:

```
./script.sh
10 15
6CD is 5
7 3
6CD is 1
bye
```

Примечание: в вызове функции из себя самой нет ничего страшного или неправильного, т.ч. смело вызывайте gcd прямо внутри gcd !

Примечание 2: для завершения работы функции в произвольном месте, можно использовать инструкцию return (все инструкции функции после return выполняться не будут). В отличии от exit эта команда завершит только функцию, а не выполнение всего скрипта целиком. Однако в данной задаче можно обойтись и без использования return!

Подсказка: в случае проблем с решением задачи, обратите внимание на наши рекомендации по написанию скриптов.

Рис. 3.21: Задание №19. Условие

26

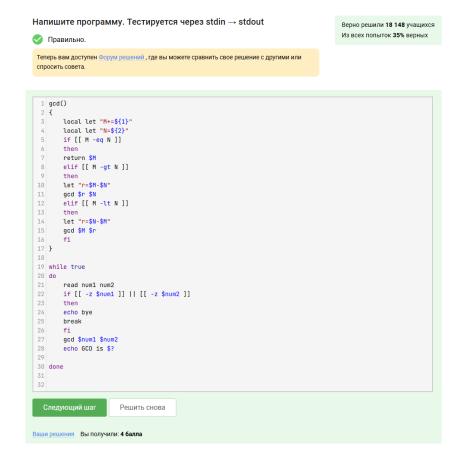


Рис. 3.22: Задание №19. Ответ

### Листинг №3.4.1. Скрипт нахождения НОД.

```
gcd()
{
    local let "M+=${1}"
    local let "N=${2}"
    if [[ M -eq N ]]
    then
    return $M
    elif [[ M -gt N ]]
    then
    let "r=$M-$N"
    gcd $r $N
```

```
elif [[ M -lt N ]]
    then
    let "r=$N-$M"
    gcd $M $r
    fi
3
while true
do
    read num1 num2
    if [[ -z $num1 ]] || [[ -z $num2 ]]
    then
    echo bye
    break
    fi
    gcd $num1 $num2
    echo GCD is $?
done
```

6. Вопрос 20-й: Реализуем калькулятор. Будем считывать уже 3 аргумента командой read. Чтобы оператор case не путал умножение "\*") и оставшиеся варианты \*), поместим знак умножения в кавычки.

Напишите **калькулятор** на bash. При запуске ваш скрипт должен ожидать ввода пользователем команды (при этом на экран выводить ничего не нужно). Команды могут быть трех типов:

- 1. Слово **"exit"**. В этом случае скрипт должен вывести на экран слово "bye" и завершить работу. 2. **Три аргумента через пробел** первый операнд (целое число), операция (одна из "+", "-", "\*", "\*", "%", "\*\*") и второй операнд (целое число). В этом случае нужно произвести указанную операцию над заданными числами и вывести результат на экран. После этого переходим в режим ожидания новой команды.
- 3. **Любая другая команда** из одного аргумента или из трех аргументов, но с операцией не из списка. В этом случае нужно вывести на экран слово "error" и завершить работу.

Чтобы проверить работу скрипта, вы можете записать сразу несколько команд в файл и передать его скрипту на stdin (т.е. выполнить ./script.sh < input.txt ). В этом случае он должен вывести сразу все ответы на экран.

Например, если входной файл будет следующего содержания:

```
10 + 1
2 ** 10
exit
```

то на экране будет:

```
1024
```

Если же на вход поступит следующий файл:

```
3 - 5
2/10
exit
```

то на экране будет:



т.к. вторая команда была **некорректной** (в ней всего один аргумент, т.к. нет пробелов между числами и операцией, а единственная допустимая команда из одного аргумента это "exit").

Подсказка: в случае проблем с решением задачи, обратите внимание на наши рекомендации по написанию скриптов.

Рис. 3.23: Задание №20. Условие

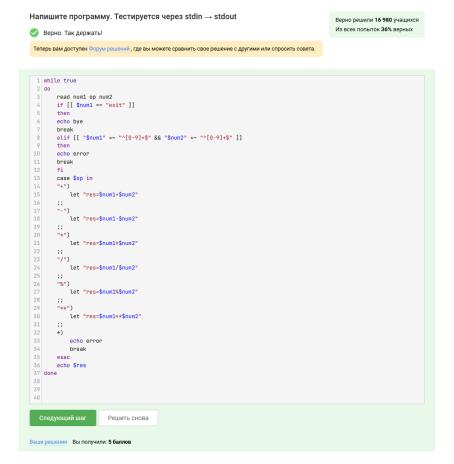


Рис. 3.24: Задание №20. Ответ

### Листинг №3.4.2. Скрипт калькулятора.

```
while true
do

    read num1 op num2
    if [[ $num1 == "exit" ]]
    then
    echo bye
    break
    elif [[ "$num1" =~ "^[0-9]+$" && "$num2" =~ "^[0-9]+$" ]]
    then
    echo error
```

```
break
    fi
    case $op in
    "+")
       let "res=$num1+$num2"
    ;;
    "-")
        let "res=$num1-$num2"
    ;;
    "*")
       let "res=$num1*$num2"
    ;;
    "/")
       let "res=$num1/$num2"
    ;;
    " 0/ " )
       let "res=$num1%$num2"
    ;;
    "**")
       let "res=$num1**$num2"
    ;;
    *)
       echo error
        break
    esac
    echo $res
done
```

## 3.5 Продвинутый поиск и редактирование

1. Вопрос 21-й: Команда - iname не учитывает регистр, поэтому найдутся те варианты, которые начинаются с star и заканчиваются любым набором в любом регистре.

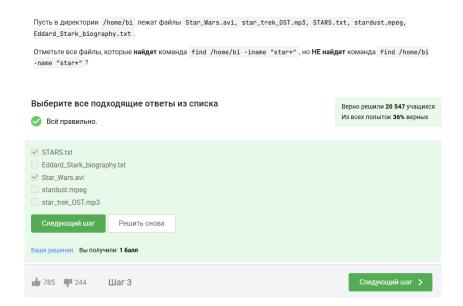


Рис. 3.25: Задание №21. Условие и ответ

2. Вопрос 22-й: Изучим каждый вариант ответа и выберем верные.

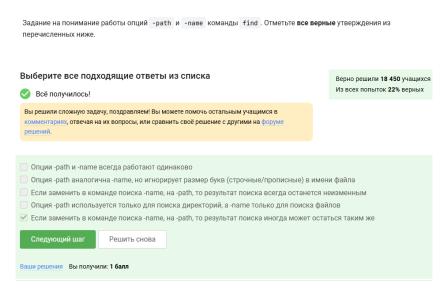


Рис. 3.26: Задание №22. Условие и ответ

3. Вопрос 23-й: Глубина поиска начинается с текущей директории, поэтому он будет проходить в области dir1 <-> dir2, не заходя в каталог dir3.

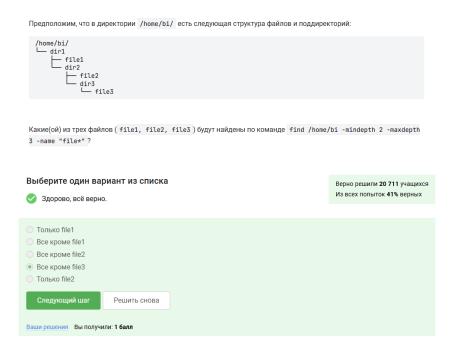


Рис. 3.27: Задание №23. Условие и ответ

- 4. Вопрос 24-й: Рассмотрим каждую из опций:
  - -А n Вывод n строк контекста после группы найденных вхождений;
  - -B n Вывод n строк контекста перез группой найденных вхождений;
  - -C n Вывод n строк контекста до и после группы найденных вхождений;

Так как до и после 10 строк слов word нет текста, то контекста не будет, а размер всех файлов будет одинаковый.

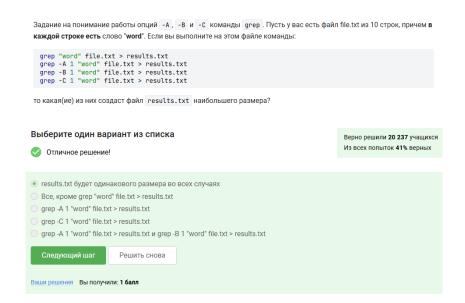


Рис. 3.28: Задание №24. Условие и ответ

5. Вопрос 25-й: Команда выведет только строки, которые: заканчиваются на слове Ubuntu или ubuntu, перед которым слитно находится не более одного вхождения символов из набора xklXKL.

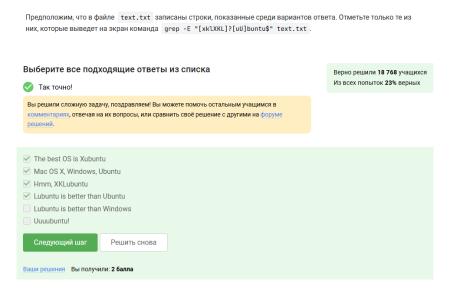


Рис. 3.29: Задание №25. Условие и ответ

6. Вопрос 26-й: Без опции - n происходит автоматический вывод всех строк, при этом р в конце выражения указывает на то, что дополнительно должны

быть выведены строки, удовлетворяющие шаблону. Произойдет задвоение строк.

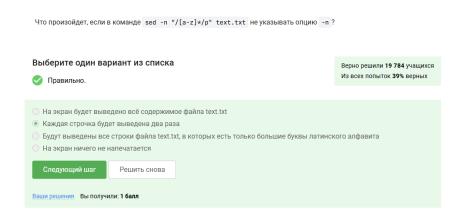


Рис. 3.30: Задание №26. Условие и ответ

- 7. Вопрос 27-й: Используем команду sed "s/ [A-Z]\{2,} / abbreviation /g" input.txt > edited.txt, в которой:
  - s указание на копирование;
  - [А-Z] любые символы из набора от А до Z;
  - \{2,} хотя бы две буквы из набора;
  - abbreviation то на что заменяем;
  - g каждое вхождение в строке;

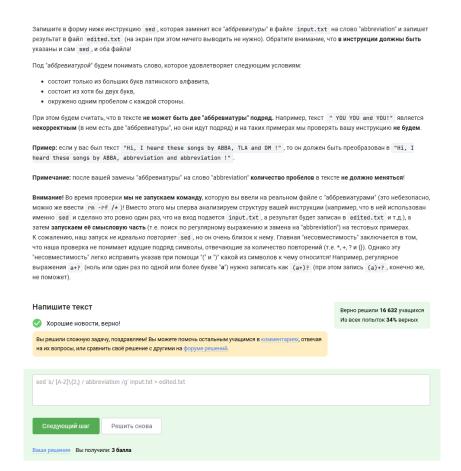


Рис. 3.31: Задание №27. Условие и ответ

### 3.6 Строим графики в gnuplot

1. Вопрос 28-й: Изучим справку man gnuplot. Из неё получим, что существует ключ -р или же --persist, который позволяет оставить окно графика после закрытия терминала.



Рис. 3.32: Задание №28. Условие и ответ

2. Вопрос 29-й: Первая строка указывает gnuplot использовать первую строку файла как название столбцов, так как файлы названий столбцов отсутствуют, gnuplot присвоит ряду стандартное название во второй строке – 'data.csv' using 1:2.

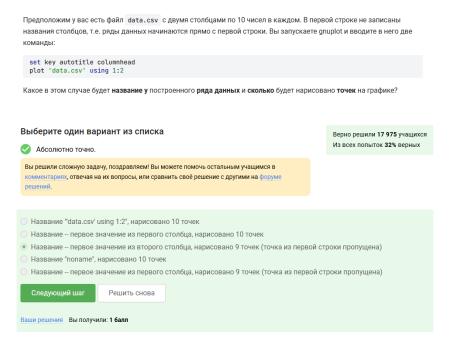


Рис. 3.33: Задание №29. Условие и ответ

3. Вопрос 30-й: Установим деления командой set, укажем ось Ох с помощью хісs и введем через запятую точки в формате ("название", значение).

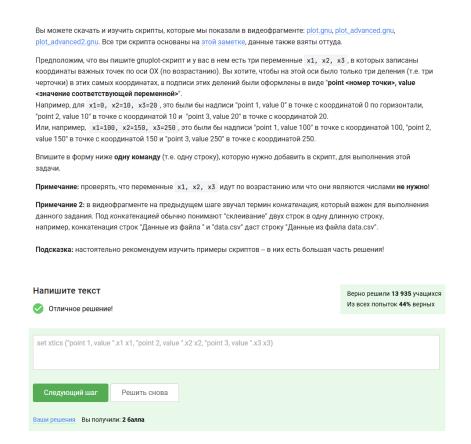


Рис. 3.34: Задание №30. Условие и ответ

### 4. Вопрос 31-й: В данном случае график:

- отразится зеркально, если поставить отрицание перед функцией и раскрыть скобки: splot -x\*\*2-y\*\*2
- станет вращаться в обратную сторону, если поменять с 10 до 350 градусов: zrot=(zrot+350)%360
- станет вращаться в два раза быстрее, если уменьшить задержку в два раза: pause 0.1

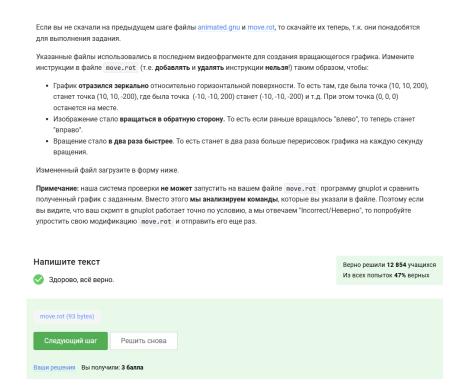


Рис. 3.35: Задание №31. Условие и ответ



Рис. 3.36: Задание №31. Файл

### 3.7 Разное

1. Вопрос 32-й: Проверим каждую строку на правильность изменения прав файла и выберем нужные варианты ответа.

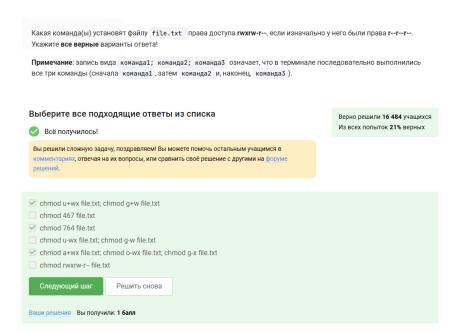


Рис. 3.37: Задание №32. Условие и ответ

2. Вопрос 33-й: Так как изначальные права у root, то владелец с помощью sudo должен дать права пользователю. Без sudo права не поменяются. Выберем верные варианты ответа.

Предположим вы использовали команду sudo для создания директории dir. По умодчанию для dir были выставлены права доступа **гwxг-xr-x** (владелец **root**, группа **root**). Таким образом никто кроме пользователя root не может ничего записывать в эту директорию, например, не может создавать файлы в ней. После выполнения какой команды user из группы group всё-таки сможет создать файл внутри dir ? Укажите все верные варианты ответов! Примечание: считаем, что все команды выполняются от имени **user**, если явно не указано, что команда выполнена с sudo. Примечание 2: мы выбрали пример с директорией, а не с файлом не случайно. Дело в том, что если создать при помощи sudo файл с правами **rw-r--r-** в директории, которая принадлежит пользователю, то возникнет любопытная ситуация. С одной стороны пользователь может удалить этот файл (т.к. ему разрешено удалять **все** файлы внутри его директории) и может прочитать его содержимое (т.к. право " $\mathbf{r}$ " у файла установлено для всех), с другой стороны он не может этот файл редактировать (т.к. право "**w**" у файла есть только для root). При этом некоторые "умные" редакторы, например, vim позволят даже редактировать этот файл, но сделают они это своеобразно: через удаление оригинала и создание копии уже с нужными правами (удалять мы можем, а раз можем читать, то и копию создать не сложно). Итого получается, что несмотря на права rw-r--r-, пользователь может сделать с этим файлом почти всё что угодно! В случае же, когда речь идет о директории созданной **root**, ситуация будет проще: пользователь сможет смотреть её содержимое (у него есть право "r"), но удалять и создавать файлы в ней не сможет (права "w" у него нет). Важно отметить, что директории в Linux это в каком-то смысле файлы. Содержимое такого "файла" — это записи о файлах и поддиректориях этой директории (грубо говоря их *названия*). Таким образом, право "**r**" у директории дает возможность просматривать "записи", т.е. просматривать её состав. Право "**w**" у директории дает возможность удалять/добавлять новые "записи", т.е. удалять/создавать файлы/поддиректории в ней. На самом деле и это еще не всё. Существует так называемый <u>sticky bit</u> (атрибут файла или директории), выставление которого меняет описанное выше поведение. Файлы (или директории) с таким атрибутом сможет удалить только их владелец вне зависимости от прав, установленных у директории, в которой эти файлы (или директории) лежат! Отдельное спасибо слушателю курса Alexey Antipovsky за помощь в оформлении Примечания 2!

Рис. 3.38: Задание №33. Условие

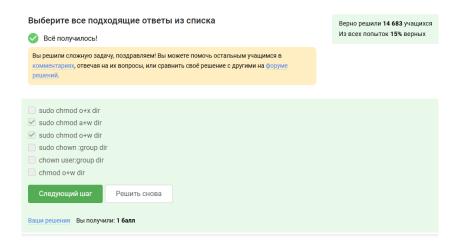


Рис. 3.39: Задание №33. Ответ

- 3. Вопрос 34-й: Откроем справку по команде wc и просмотрим ключи:
- -1 имя\_файла вывести количество строк;
- -с имя\_файла вывести количество байт;
- - т имя\_файла вывести количество символов;
- - L имя\_файла вывести длину самой длинной строки;
- - w имя\_файла вывести количество слов;

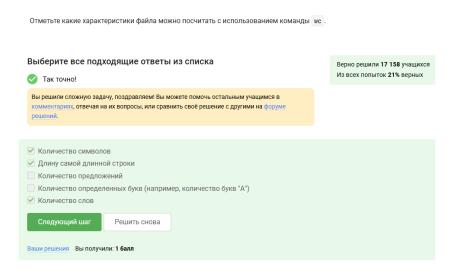


Рис. 3.40: Задание №34. Условие и ответ

4. Вопрос 35-й: Выведем команду du, анализирующую занимаемое файло-

вое пространство, с двумя ключами -h или --human-readable и -s или -summarize.

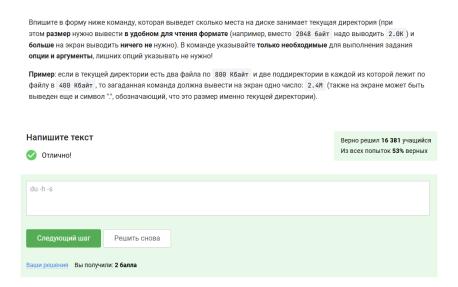


Рис. 3.41: Задание №35. Условие и ответ

5. Вопрос 36-й: Создадим директорию с помощью, а номера укажем с помощью команды {1..3}, которая проходит по значениям от 1 до 3.

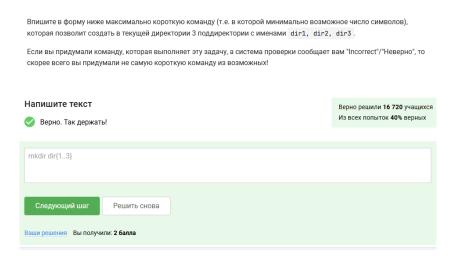


Рис. 3.42: Задание №36. Условие и ответ

## 4 Вывод

В результате выполнения блока внешнего курса  $N^{\circ}3$  "Продвинутые темы" я ознакомился с редактором vim и написанием скриптов на bash, изучил продвинутый поиск файлов и их продвинутое редактирование и научился строить графики в gnuplot.

# Список литературы

1. Gurevich A., Przhibelskiy A. Введение в Linux [Электронный ресурс]. Stepik. URL: https://stepik.org/course/73/syllabus.