

Отчёт по прохождению внешнего курса на Stepik (этап 3)

Введение в Linux

Дарья Эдуардовна Ибатулина

Содержание

1	Цель работы	6
2	Задание	7
3	Теоретическое введение	8
4	Выполнение лабораторной работы	9
4.1	Текстовый редактор vim (3.1)	9
4.2	Скрипты на bash: основы (3.2)	14
4.3	Скрипты на bash: ветвления и циклы (3.3)	18
4.4	Скрипты на bash: разное (3.4)	24
4.5	Продвинутый поиск и редактирование (3.5)	31
4.6	Строим графики в gnuplot (3.6)	36
4.7	Разное (3.7)	43
5	Выводы	47
	Список литературы	48

Список иллюстраций

4.1	Задание 3.1 (1)	9
4.2	Справка по vim	10
4.3	Задание 3.1 (2)	11
4.4	Задание 3.1 (3)	11
4.5	Задание 3.1 (3)	12
4.6	Задание 3.1 (4)	13
4.7	Задание 3.1 (4)	13
4.8	Задание 3.1 (5)	14
4.9	Проверка в терминале	15
4.10	Задание 3.2 (1)	15
4.11	Задание 3.2 (2)	16
4.12	Задание 3.2 (3)	17
4.13	Задание 3.2 (4)	18
4.14	Задание 3.3 (1)	18
4.15	Задание 3.3 (1)	19
4.16	Задание 3.3 (2)	20
4.17	Задание 3.3 (3)	21
4.18	Задание 3.3 (4)	21
4.19	Проверка работы скрипта	22
4.20	Задание 3.3 (5)	22
4.21	Задание 3.4 (1)	25
4.22	Задание 3.4 (2)	26
4.23	Задание 3.4 (4)	28
4.24	Задание 3.4 (5)	31
4.25	Задание 3.5 (1)	32
4.26	Задание 3.5 (2)	33
4.27	Задание 3.5 (3)	33
4.28	Задание 3.5 (4)	34
4.29	Задание 3.5 (5)	34
4.30	Задание 3.5 (5) - проверка на моём Linux	35
4.31	Задание 3.5 (6)	35
4.32	Задание 3.5 (6)	36
4.33	Задание 3.6 (1)	37
4.34	Задание 3.6 (1) - мануал по gnuplot	37
4.35	Задание 3.6 (2)	38
4.36	Задание 3.6 (2) - файл с данными	39
4.37	Задание 3.6 (2) - результат	40

4.38 Задание 3.6 (3)	41
4.39 Задание 3.6 (4)	41
4.40 Задание 3.7 (1)	43
4.41 Задание 3.7 (3)	44
4.42 Задание 3.7 (4)	45
4.43 Задание 3.7 (5)	45

Список таблиц

1 Цель работы

Пройти курс “Введение в Linux” на платформе [stepik.org] и получить сертификат. Для этого необходимо просмотреть видеоролики и выполнить задания, чтобы закрепить полученный материал. Процесс выполнения заданий требуется записать в виде скринкаста.

2 Задание

Пройти 3 этапа курса, записывая скринкасты, получить сертификат, сделать отчёт.

3 Теоретическое введение

Linux — это семейство операционных систем (ОС), работающих на основе одноименного ядра. Нет одной операционной системы Linux, как, например, Windows или MacOS. Есть множество дистрибутивов (набор файлов, необходимых для установки ПО), выполняющих конкретные задачи.

Линус Торвальдс — первый разработчик и создатель Linux. Именно в честь него и была названа ОС. В 1991 году Линус начал работу над собственной ОС семейства Unix. Через три года появилась первая версия, доступная для скачивания. Но тогда она имела очень низкий спрос — ей пользовались буквально несколько человек. Только через 10 лет ОС Linux получила широкое распространение. Сообщество программистов подхватило идею свободного ПО, специалисты стали помогать развивать проект. В ходе курса я познакомлюсь с основными командами терминала, научусь работать с серверами, познаю текстовый редактор *vi/vim* и рисование графиков в *gnuplot*.

4 Выполнение лабораторной работы

4.1 Текстовый редактор vim (3.1)

Какую клавишу(и) нужно нажать на клавиатуре, чтобы выйти из редактора vim? Считайте, что вы только что открыли файл и вам сразу понадобилось выйти из редактора. Чтобы это сделать, нужно написать: :q и нажать , а чтобы редактор не спрашивал, нужно ли сохранять файл или нет, нужно было бы ещё добавить ! после q. Однако, так как в задании сказано, что мы сразу выходим из редактора, ничего в нём не написав, то достаточно команды :q (рис. 4.1).

3.1 Текстовый редактор vim 12 из 13 шагов пройдено 7 из 10 баллов получено

Вы прошли больше 80% курса, оставьте отзыв [Оставить отзыв](#) [Нет, спасибо](#)

Какую клавишу(и) нужно нажать на клавиатуре, чтобы выйти из редактора vim? Считайте, что вы только что открыли файл и вам сразу понадобилось выйти из редактора.

Выберите один вариант из списка

✓ Здорово, всё верно.

Верно решили 32 523 учащихся
Из всех попыток 69% верных

- ☐ ":" , затем "q"
- ☐ "Ctrl", затем "x"
- ☐ "q"
- ☐ "Esc"
- ☒ ":" , затем "q", затем "Enter"

[Следующий шаг](#) [Решить снова](#)

Ваши решения Вы получили: 1 балл из 1

Рис. 4.1: Задание 3.1 (1)

Далее, нам требуется посмотреть, какие есть различия между word и WORD в vim (рис. 4.2).

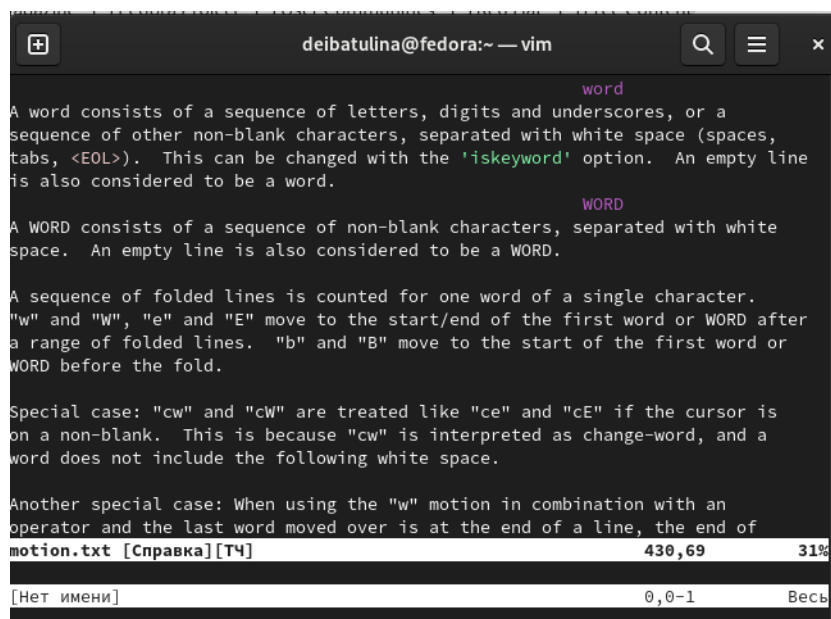


Рис. 4.2: Справка по vim

И отметит затем все верные утверждения про данную строку:

Strange_ TEXT is_here. 2=2 YES!

Сопоставив информацию в справке и данное задание, отмечаем верные утверждения (рис. 4.3):

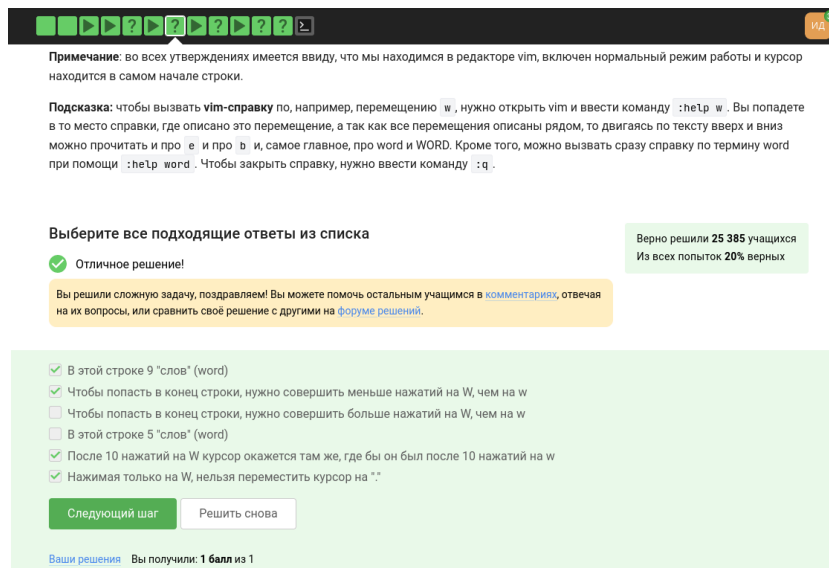


Рис. 4.3: Задание 3.1 (2)

Следующее задание: нужно выбрать, какие нажатия клавиш преобразуют одну строку в другую. Данное задание можно выполнить как аналитически, так и проверить в своём vim. Я выбрала второй вариант, попробовала ввести все варианты (рис. 4.4) и получила следующие правильные ответы (рис. 4.5):

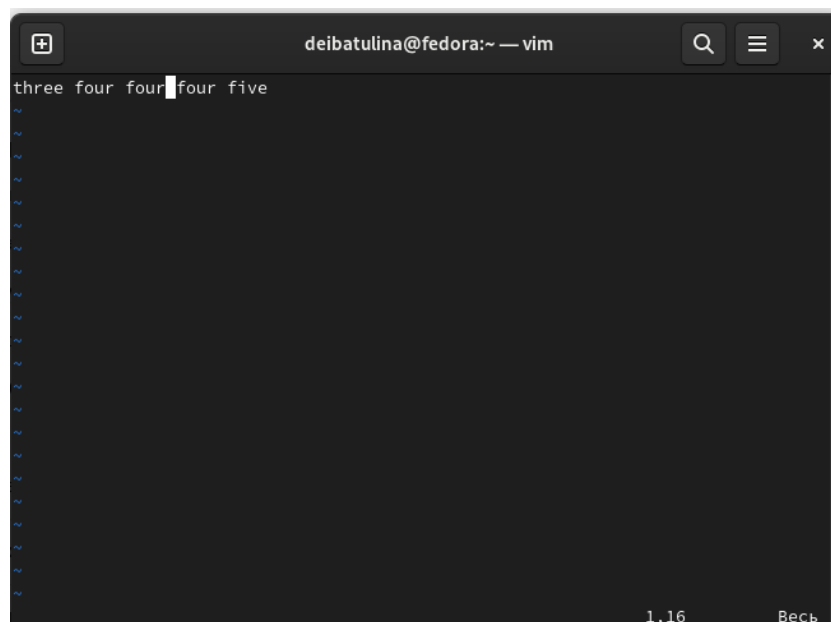


Рис. 4.4: Задание 3.1 (3)

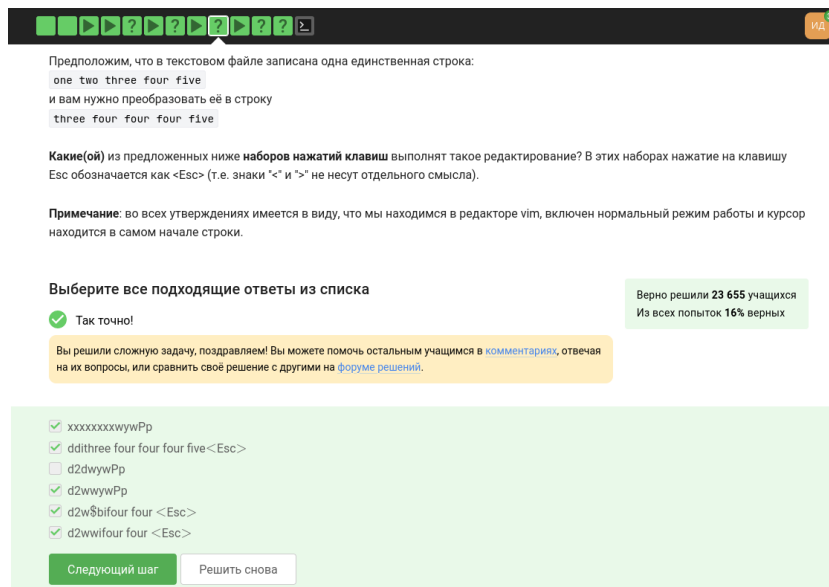


Рис. 4.5: Задание 3.1 (3)

Затем переходим к выполнению следующего задания:

Предположим, что вы открыли файл в редакторе vim и хотите заменить в этом файле все строки, содержащие слово Windows, на такие же строки, но со словом Linux. Если в какой-то строке слово Windows встречается больше, чем один раз, то заменить на Linux в этой строке нужно только самое первое из этих слов.

Какую команду нужно ввести для этого в vim? Укажите необходимую команду целиком (т.е. включая ввод ":" в самом начале), однако нажатие на Enter после ввода команды обозначать никак не нужно.

Для этого я проделала множество попыток в своём vim, и нашла соответствующую команду: :%s/Windows/Linux (рис. 4.6).

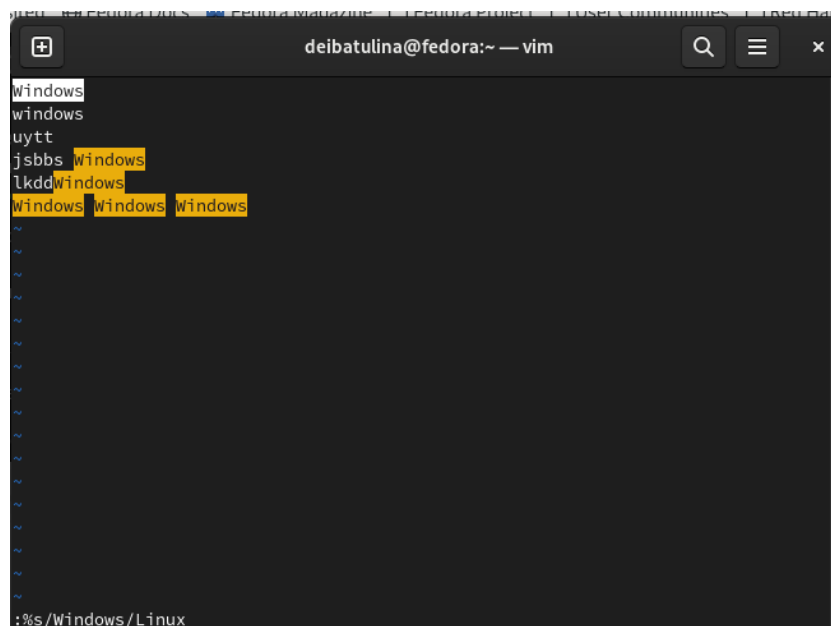


Рис. 4.6: Задание 3.1 (4)

Ответ верный (рис. 4.7).

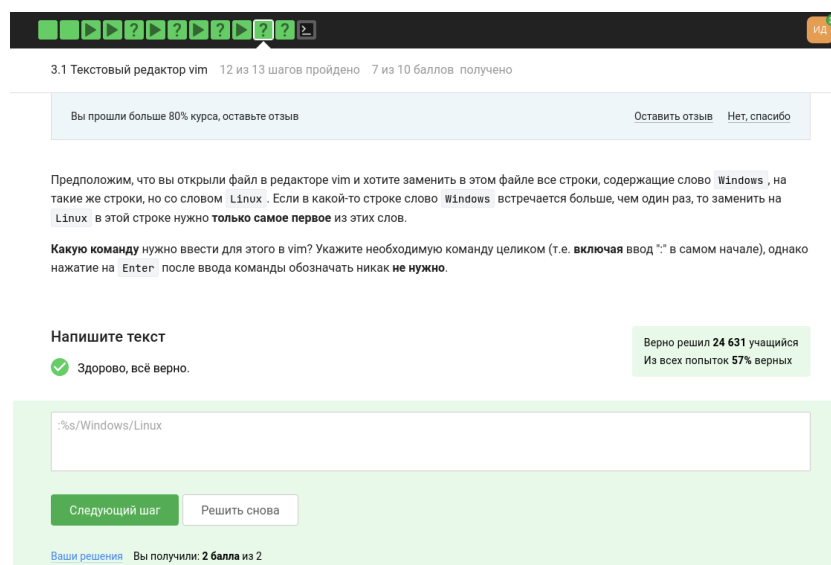


Рис. 4.7: Задание 3.1 (4)

Итак, следующее задание: необходимо самостоятельно ознакомиться с режимом выделения в vim, почитав документацию в Интернете, и отметить все

верные утверждения (рис. 4.8):

Мы совсем не рассказали вам про третий режим работы vim – режим **выделения (Visual)**. Предлагаем вам ознакомиться с ним самостоятельно. Например, это можно сделать во время прохождения упражнений в vimtutor, который мы настоятельно рекомендуем вам для изучения vim!

Чтобы убедиться, что вы разобрались с этим режимом работы, отметьте, пожалуйста, **все верные** утверждения из списка ниже.

Подсказка: если вы не хотите проходить vimtutor целиком, то можете открыть его и поиском найти слово "Visual". Вы попадете в задание, прохождение которого будет вполне достаточно, чтобы выполнить это задание.

Выберите все подходящие ответы из списка

Верно решили **23 497** учащихся
Из всех попыток **29%** верных

☒ Правильно.

Вы решили сложную задачу, поздравляем! Вы можете помочь остальным учащимся в [комментариях](#), отвечая на их вопросы, или сравнить свое решение с другими на [форуме решений](#).

- ☒ Выйти из режима выделения можно, нажав клавишу Esc два раза
- ☒ Режим выделения открывается из нормального режима по нажатию "v"
- ☐ Чтобы выйти из режима выделения, нужно ввести :q
- ☐ Режим выделения открывается при помощи команды :visual
- ☒ Когда вы находитесь в режиме выделения, внизу редактора горит надпись – VISUAL – (или – ВИЗУАЛЬНЫЙ РЕЖИМ –)
- ☒ В режиме выделения можно использовать команды d (удалить) и у (скопировать)

Следующий шаг Решить снова

Ваши решения Вы получили: **2 балла** из 2

Рис. 4.8: Задание 3.1 (5)

4.2 Скрипты на bash: основы (3.2)

Первое задание звучит следующим образом: Предположим, что вы открыли терминал и у вас в нем запущена оболочка bash. Вы набираете в ней команды A1, A2, A3, а затем запускаете оболочку sh. В этой оболочке вы набираете команды B1, B2, B3 и запускаете оболочку bash. И, наконец, в этой последней оболочке вы набираете команды C1, C2, C3. Если теперь вы попытаете при помощи стрелочек вверх/вниз перемещаться по истории набранных команд, то команды из какого набора(ов) будут появляться? На своём компьютере я провела эксперимент (рис. 4.9), и выяснила, что будут появляться команды из последнего набора, т.е. набора C (рис. 4.10).

```
deibatulina@fedora:~  
[deibatulina@fedora ~]$ pwd  
/home/deibatulina  
[deibatulina@fedora ~]$ ls  
backup      interacter.py  Документы      Общедоступные  
bash1       lab07.sh~     Загрузки       'Рабочий стол'  
bash.sh     names.txt     'Задание 1.odt'  
bin         text.txt      Изображения    Шаблоны  
bowtie.log  work         Курс_Степик_задание_1.xml  
file.txt    Видео        Музыка  
[deibatulina@fedora ~]$ cat names.txt  
Darya  
Svetlana  
anastasia  
regina  
Michael  
Ilya  
andrey  
[deibatulina@fedora ~]$ sh  
sh-5.2$ ls -l  
итого 52  
drwxr-xr-x. 1 deibatulina deibatulina 20 апр 8 19:43 backup  
-rwxr-xr-x. 1 deibatulina deibatulina 111 мая 27 17:51 bash1  
-rwxr-xr-x. 1 deibatulina deibatulina 36 мая 27 17:34 bash.sh  
drwxr-xr-x. 1 deibatulina deibatulina 8 фев 22 14:59 bin  
-rw-r--r--. 1 deibatulina deibatulina 206 мая 27 14:52 bowtie.log  
-r--rw-r--. 1 deibatulina deibatulina 0 мая 27 18:33 file.txt  
-rwxr-xr-x. 1 deibatulina deibatulina 361 мая 12 10:54 interacter.py  
-rw-r--r--. 1 deibatulina deibatulina 155 апр 1 16:35 lab07.sh~  
-rw-r--r--. 1 deibatulina deibatulina 52 мая 12 10:57 names.txt  
-rw-r--r--. 1 deibatulina deibatulina 148 мая 27 18:14 text.txt  
drwxr-xr-x. 1 deibatulina deibatulina 64 мая 6 16:39 work  
drwxr-xr-x. 1 deibatulina deibatulina 0 фев 11 16:23 Видео
```

Рис. 4.9: Проверка в терминале

Надеемся, что вы разобрались, что одну оболочку (например, `sh`) можно запустить из другой оболочки (например, из `bash`).

Предположим, что вы открыли терминал и у вас в нем запущена оболочка `bash`. Вы набираете в ней команды `A1`, `A2`, `A3`, а затем запускаете оболочку `sh`. В этой оболочке вы набираете команды `B1`, `B2`, `B3` и запускаете оболочку `bash`. И, наконец, в этой последней оболочке вы набираете команды `C1`, `C2`, `C3`. Если теперь вы попытаете при помощи стрелочек вверх/вниз перемещаться по истории набранных команд, то команды из какого набора(ов) будут появляться?

Выберите один вариант из списка

☒ Всё получилось!

Верно решили 30 266 учащихся
Из всех попыток 65% верных

- ☐ Только из набора A
- ☐ Из наборов B и C
- ☐ Из наборов A и C
- ☒ Только из набора C
- ☐ Никакие команды появляться не будут

Следующий шаг

Решить снова

Ваши решения Вы получили: 1 балл из 1

Рис. 4.10: Задание 3.2 (1)

Далее, следующее задание.

Предположим, что вы находитесь в директории `/home/bi/Documents/` и запускаете в ней скрипт следующего содержания:

```
cd /home/bi/ # сейчас мы в папке bi
touch file1.txt # создаём в ней файл file1.txt
cd /home/bi/Desktop/ # а теперь мы в директории bi/Desktop/
```

Как будет выглядеть абсолютный путь до созданного файла file1.txt по окончании работы скрипта? Ответ: /home/bi/file1.txt (рис. 4.11).

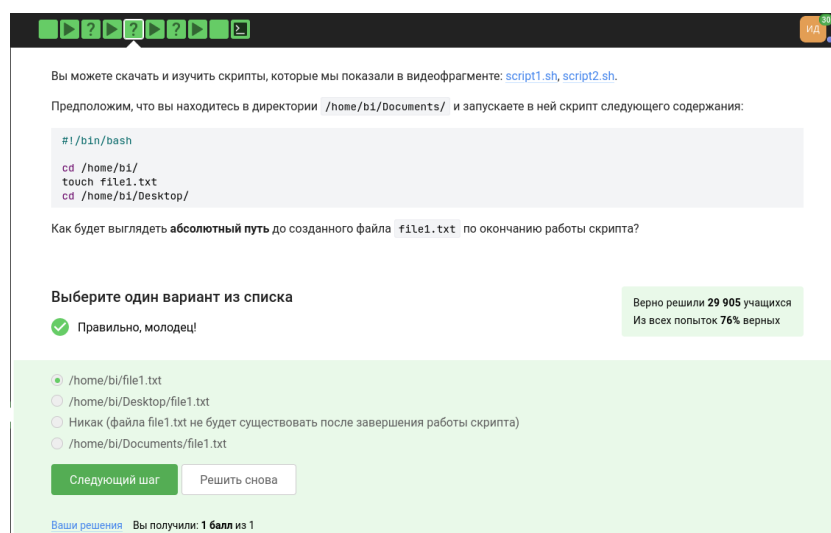


Рис. 4.11: Задание 3.2 (2)

Из видеоурока мы узнали, что имена переменных в `bash` могут включать цифры и символы - и `_`, однако **не могут начинаться с цифр** и содержать любые другие символы, кроме разрешённых. Соответственно, отмечаем все подходящие варианты (рис. 4.12).

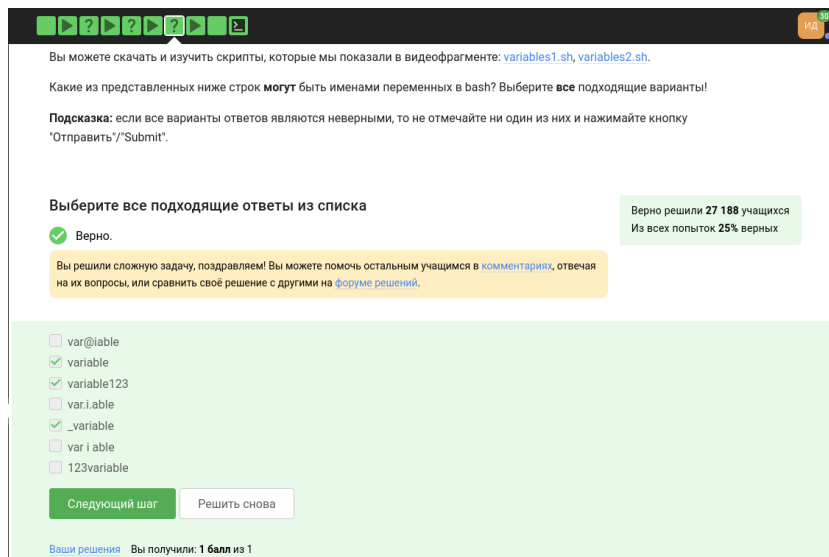


Рис. 4.12: Задание 3.2 (3)

В следующем задании требуется написать скрипт на bash, который принимает на вход два аргумента и выводит на экран строку следующего вида:

Arguments are: \$1=первый_аргумент \$2=второй_аргумент Ответ на задачу будет выглядеть так (рис. 4.13):

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

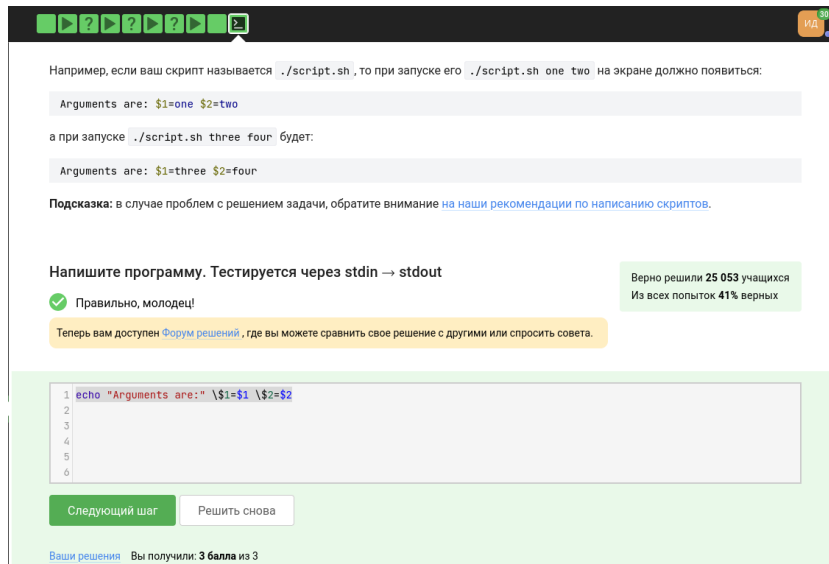


Рис. 4.13: Задание 3.2 (4)

4.3 Скрипты на bash: ветвления и циклы (3.3)

Задание 1 звучит так (рис. 4.14):

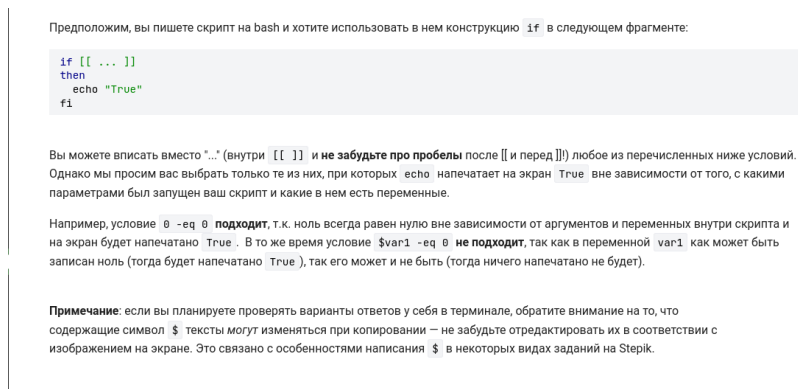


Рис. 4.14: Задание 3.3 (1)

Разберём все варианты ответа:

- `!(4 -le 3) # !(4 <= 3)`, т.е. `4 > 3`
- `$# -ge 0 # $#` считает общее число аргументов, `-ge 0` означает, что оно `>= 0`

- -s \$0 # размер скрипта больше 0
- 5 -ge 5 # 5 >= 5
- -n \$0 # имя скрипта - не пустая строка
- -e \$0 # путь к скрипту существует

Все они подходят под условие задачи, поэтому отмечаем их все (рис. 4.15).

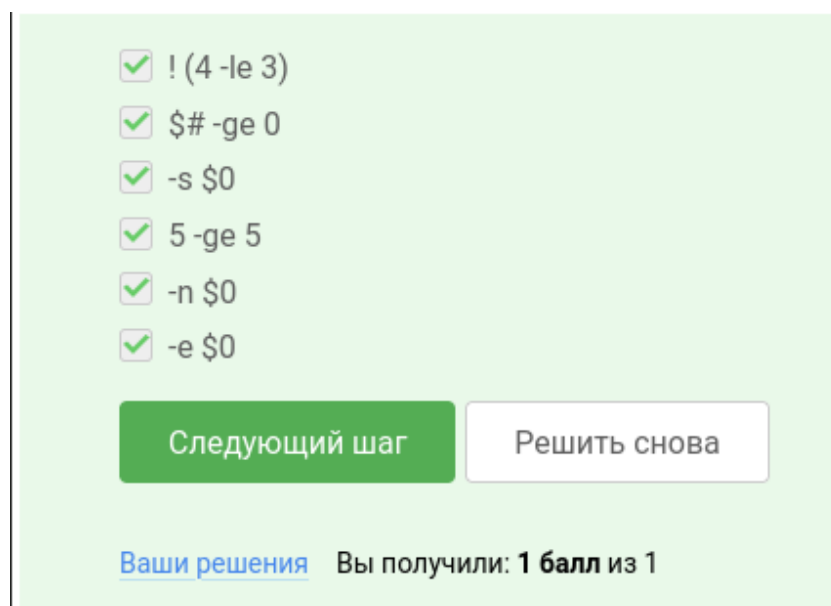


Рис. 4.15: Задание 3.3 (1)


Далее, следующее задание.

Изучим скрипт:

```
if [[ $var -gt 5 ]] # если var > 5 then    echo "one" # то печатаем
"one" elif [[ $var -lt 3 ]] # если var < 3 then    echo "two" # то
печатаем "two" elif [[ $var -eq 4 ]] # если var = 4 then    echo
"three" # то печатаем "three" else    echo "four" # иначе печатаем
"four" fi
```

Какие строки и в какой последовательности он выведет на экран, если сначала этот скрипт запустили задав переменную var=3, а затем запустили еще раз, но

уже с `var=5`? Если `var = 3`, то напечатается следующее: *four*, если же `var=5`, то напечатается *four*. Ответ: *four, four* (рис. 4.16).



The screenshot shows a quiz interface. At the top, it says "Выберите один вариант из списка" (Choose one option from the list). Below this, there is a green checkmark icon followed by the text "Хорошие новости, верно!" (Good news, correct!). Below this, there is a light green box containing four radio button options: "Сначала one, потом two", "Сначала two, потом one", "Сначала four, потом one", and "Сначала four, потом four". The fourth option is selected. Below the options are two buttons: "Следующий шаг" (Next step) and "Решить снова" (Solve again). At the bottom, it says "Ваши решения" (Your solutions) and "Вы получили: 1 балл из 1" (You received: 1 point out of 1).

Рис. 4.16: Задание 3.3 (2)

В следующем задании нужно написать 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
```

Вот какой скрипт получился у меня, и система засчитала его как верный. Также я проверила работу данного скрипта в своём терминале и убедилась, что он работает корректно (рис. 4.17).

```
if [[ $1 -eq 1 ]]; then
    echo "$1 student"
elif [[ $1 -gt 1 && $1 -le 4 ]]; then
```

```

    echo "$1 students"
elif [[ $1 -ge 5 ]]; then
    echo "A lot of students"
else
    echo "No students"
fi

```

```

[deibatulina@fedora ~]$ ./bash3.sh 3
3 students
[deibatulina@fedora ~]$ bash3.sh 5
bash: bash3.sh: команда не найдена...
[deibatulina@fedora ~]$ ./bash3.sh 1
1 student
[deibatulina@fedora ~]$ ./bash3.sh 0
No students
[deibatulina@fedora ~]$ ./bash3.sh 3
3 students
[deibatulina@fedora ~]$ ./bash3.sh 10
A lot of students
[deibatulina@fedora ~]$

```

Рис. 4.17: Задание 3.3 (3)

В следующем задании требуется проанализировать скрипт на bash и ответить на вопрос по нему (рис. 4.18):

Посмотрите на фрагмент bash-скрипта:

```

for str in a , b , c_d
do
    echo "start"
    if [[ $str > "c" ]]
    then
        continue
    fi
    echo "finish"
done

```

Если запустить этот скрипт, то **сколько раз** на экран будет выведено слово "start", а сколько раз слово "finish"?

Выберите один вариант из списка

Верно решили 24 582 учащихся
Из всех попыток 45% верных

☒ Так точно!

☐ 3 раза "start" и ни разу "finish"
☐ 3 раза "start" и 3 раза "finish"
☐ 3 раза "start" и 2 раза "finish"
☒ 5 раз "start" и 4 раза "finish"

Рис. 4.18: Задание 3.3 (4)

Запустив этот скрипт на своём компьютере (рис. 4.19), отмечаем правильный ответ.

```
[deibatulina@fedora ~]$ gedit bash3.sh
[deibatulina@fedora ~]$ ./bash3.sh
start
finish
start
finish
start
finish
start
finish
start
[deibatulina@fedora ~]$
```

Рис. 4.19: Проверка работы скрипта

В заключительном задании этого блока требуется написать скрипт (рис. 4.20):

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 **"<Имя>, your group is <группа>"**, где **<группа>** 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:

№1

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

Рис. 4.20: Задание 3.3 (5)

```
while [[ 1==1 ]]
```

```
do
```

```
    group=""
```

```
echo "enter your name:"

read name # читаем имя

if [[ -z $name ]] # если строка пустая

then

    break # то выходим из цикла

fi

echo "enter your age:"

read age # читаем возраст

if [[ $age -eq 0 ]] # если возраст равен 0

then

    break # то выходим из цикла

fi

if [[ $age -le 16 ]] # если возраст <= 16

then

    group="child" # то группа: child
```

```

elif [[ $age -le 25 ]] # если же 16 < возраст <= 25

then

    group="youth" # то группа: youth

else # иначе (если возраст > 25)

    group="adult" # группа: adult

fi

echo "$name, your group is $group"

done

echo "bye"

```

4.4 Скрипты на bash: разное (3.4)

Задание 1 звучит так: Какие(ая) из предложенных ниже инструкций увеличат значение переменной *a* на значение переменной *b*? Например, если в *a* было записано 10, в *b* было 5, то в *a* должно записаться 15. Выберите все подходящие варианты! Варианты ответов:

1. let "a+=b"
2. let a=a+b
3. let "a+=b"
4. let "a = a + b"

5. `let a = a + b`

Из предложенных вариантов нам подходят 1, 2 и 4 (рис. 4.21):

Какие(ая) из предложенных ниже инструкций увеличат значение переменной `a` на значение переменной `b`? Например, если в `a` было записано 10, в `b` было 5, то в `a` должно записаться 15. Выберите **все подходящие** варианты!

Примечание: если вы планируете проверять варианты ответов у себя в терминале, обратите внимание на то, что содержащие символ `$` тексты *могут* изменяться при копировании — не забудьте отредактировать их в соответствии с изображением на экране. Это связано с особенностями написания `$` в некоторых видах заданий на Stepik.

Подсказка: обратите особое внимание на кавычки и **пробелы**, они могут как принципиально изменить команду, так и ни на что не повлиять (в зависимости от команды и контекста)!

Выберите все подходящие ответы из списка

Верно решили 22 116 учащихся
Из всех попыток 20% верных

✓ Правильно.

Вы решили сложную задачу, поздравляем! Вы можете помочь остальным учащимся в [комментариях](#), отвечая на их вопросы, или сравнить свое решение с другими на [форуме решений](#).

- ☒ `let "a+=b"`
- ☒ `let a=a+b`
- ☐ `let "a+=b"`
- ☒ `let "a = a + b"`
- ☐ `let a = a + b`

Следующий шаг Решить снова

[Ваши решения](#) Вы получили: 1 балл из 1

Рис. 4.21: Задание 3.4 (1)

Задание 2 такое:

Пусть вы находитесь в директории `/home/bi/Documents/` и запускаете в ней скрипт следующего содержания:

```
cd /home/bi/ # переход в директорию /home/bi/  
echo "`pwd`" # вывод текущей директории
```

Что в этом случае выведет команда `echo` на экран?

Разберём все варианты ответа:

- `/home/bi/Documents`
- Код возврата команды `pwd` (0 в случае успешного выполнения и не 0 в случае ошибок)
- `/home/bi`

- pwd
- pwd

Проверим работу данной программы на своём терминале, предварительно заменив строку `/home/bi/` на `/home/deibatulina/` (рис. 4.22). Под условие задачи подходит ответ 3, поэтому и отмечаем его.

```
[deibatulina@fedora Документы]$ gedit bash3.sh
[deibatulina@fedora Документы]$ ./bash3.sh
/home/deibatulina
[deibatulina@fedora Документы]$
```

Рис. 4.22: Задание 3.4 (2)

Переходим к следующему заданию блока:

Мы рассказали, что можно проверить код возврата внешней программы прямо в конструкции `if` при помощи `if program options arguments` (действия внутри `if` выполняются, если программа закончилась с кодом 0). Однако это не всегда правда! Если запуск внешней программы выводит что-то в `stdout`, то в проверку `if` поступит именно этот вывод, а не код возврата! Вы можете убедиться в этом, написав простой `bash`-скрипт с использованием, например, `if pwd`.

Однако как быть, если хочется всё-таки запустить программу `program`, которая пишет что-то в `stdout` и потом выполнить какие-то действия если ее код возврата равен 0? Выберите все верные утверждения или правильно работающие конструкции `if`.

Если в косых кавычках просто записано название программы, то отработает программа (и в `if` пойдет результат отработки программы, т.е. то, что выводится в `stdout`). Если же в косых кавычках помимо названия программы есть еще что-то: `pwd > file.txt`, то в `if` будет попадать именно код завершения. При этом квадратные скобки не нужны.

Разберём все варианты ответа:

- `if program > some_file.txt #` нам подходит

- Сначала `var=program` (это результат `stdout`), затем `if [[$var -eq 0]]` # не подходит
- Ничего сделать нельзя # один верный вариант уже нашёлся значит, этот вариант не подходит
- Сначала запустить `program`, затем `if [[$? -eq 0]]` # код завершения программы пойдёт в `if`, подходит
- `if [[program -eq 0]]` # не подойдёт, т.к. в `if` пойдёт результат `stdout`

Таким образом, отмечаем варианты 1 и 4 и оказываемся правы.

Следующий номер: *Посмотрите на функцию из `bash-скрипта`:*

```
counter () # takes one argument {    local let "c1+= $1"    let
"c2+= ${1}*2" }
```

Впишите в форму ниже строку, которую выведет на экран команда `echo "counters are $c1 and $c2"` если она находится в скрипте после десяти вызовов функции `counter` с параметрами сначала 1, затем 2, затем 3 и т.д., последний вызов с параметром 10.

Разбираем работу функции по шагам: `c1` - локальная переменная, обратиться к ней вне функции нельзя, а что касается переменной `c2`, она не локальная, поэтому обратиться к ней вне функции можно: 1-м шаге (при вызове функции с параметром 1) $c2 = 2$, на 2-м: $c2 = 2 + 2 * 2 = 6$, на 3-м: $c2 = 6 + 3 * 2 = 12$, и т.д. На 10-м шаге значение `c2` будет равно 110, поэтому искомая строка будет выглядеть так: `counters are and 110`.

Вводим искомую строку в поле ответа, и получаем, что ответ верен (рис. 4.23).

Посмотрите на функцию из bash-скрипта:

```
counter () # takes one argument
{
    local let "c1+=$1"
    let "c2+=$((1)*2"
}
```

Впишите в форму ниже строку, которую выведет на экран команда `echo "counters are $c1 and $c2"` если она находится в скрипте **после десяти вызовов** функции `counter` с параметрами сначала 1, затем 2, затем 3 и т.д., последний вызов с параметром 10.

Подсказка: этот пример можно решить в уме, но если система проверки не принимает ваше решение, то возможно вы что-то упустили (возможно что-то совсем небольшое/невидимое 😊). В этом случае имеет смысл написать небольшой скрипт на bash, который проделает ровно то, что указано в задании и посимвольно сверить свой ответ с тем, что он выдаст на экран.

Напишите текст

✓ Хорошие новости, верно!

Верно решили 20 009 учащихся
Из всех попыток 28% верных

Вы решили сложную задачу, поздравляем! Вы можете помочь остальным учащимся в [комментариях](#), отвечая на их вопросы, или сравнить своё решение с другими на [форуме решений](#).

counters are and 110

Рис. 4.23: Задание 3.4 (4)

Задание с онлайн-терминалом:

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 ”, 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 *M* больше *N*, то запускаем ту же функцию *gcd*, но в качестве первого аргумента передаем (*M-N*), а в качестве второго *N*. Если же наоборот, *M* меньше *N*, то запускаем функцию *gcd* с первым аргументом *M*, а

вторым ($N-M$).

Напишем скрипт и объясним его:

```
while [ true ] # пока верно

do

    read n1 n2 # считываем n1, n2

    if [ -z $n1 ]; then # условие: если пользователь ничего не ввёл или ввёл пустую строку

        echo "bye" # прощаемся с ним

        break

    else # иначе

        gcd () { # вызываем функцию gcd внутри себя самой

            remainder=1 # остаток = 1

            if [ $n2 -eq 0 ] # условие: если n2 = 0, то

            then

                echo "bye" # прощаемся с ним

            fi

            while [ $remainder -ne 0 ] # пока остаток не равен 0
```

```
do

remainder=$((n1%n2)) # находим остаток от деления n1 на n2

n1=$n2 # приравниваем теперь n1 к n2

n2=$remainder # а n2 - это теперь остаток

done # конец do

}

gcd $1 $2 # НОД от n1 n2

echo "GCD is $n1" # выводим НОД

fi

done # конец do
```

Система приняла код (рис. 4.24).

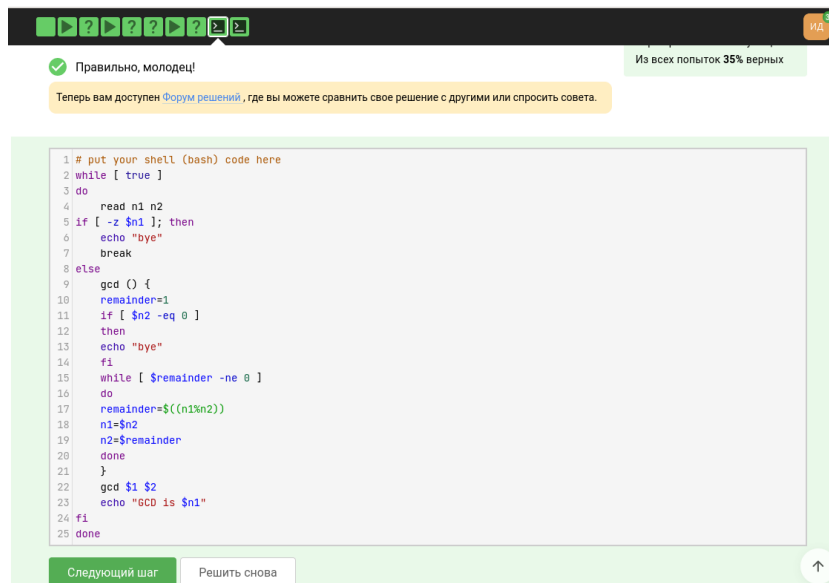


Рис. 4.24: Задание 3.4 (5)

4.5 Продвинутый поиск и редактирование (3.5)

Первое задание такое:

Пусть в директории /home/bi лежат файлы Star_Wars.avi, star_trek_OST.mp3, STARS.txt, stardust.mpeg, Eddard_Stark_biography.txt.

Отметьте все файлы, которые найдет команда `find /home/bi -iname "star"`, но НЕ найдет команда `find /home/bi -name "star"`?

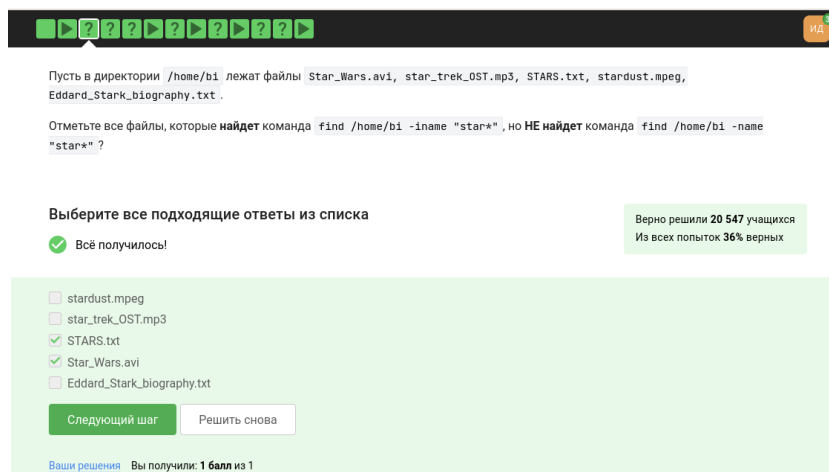
Объяснение: `star*` означает, что до слова никаких символов быть не может, а после слова может идти бесконечное количество символов. Также опция `-iname` ищет без учёта регистра, а просто `-name` - с учётом регистра.

Варианты:

- stardust.mpeg - найдут обе команды
- star_trek_OST.mp3 - найдут обе команды
- STARS.txt - найдёт только первая команда

- Star_Wars.avi - найдёт только первая команда
- Eddard_Stark_biography.txt - не найдёт ни первая, ни вторая команды

Выбираем подходящие нам ответы и получаем сообщение о том, что всё мы сделали правильно (рис. 4.25).



Пусть в директории `/home/b1` лежат файлы `Star_Wars.avi`, `star_trek_OST.mp3`, `STARS.txt`, `stardust.mpeg`, `Eddard_Stark_biography.txt`.

Отметьте все файлы, которые **найдет** команда `find /home/b1 -iname "star*"`, но **НЕ найдет** команда `find /home/b1 -name "star*"` ?

Выберите все подходящие ответы из списка

Всё получилось!

Верно решили 20 547 учащихся
Из всех попыток 36% верных

- ☐ stardust.mpeg
- ☐ star_trek_OST.mp3
- ☒ STARS.txt
- ☒ Star_Wars.avi
- ☐ Eddard_Stark_biography.txt

Следующий шаг Решить снова

Ваши решения Вы получили: 1 балл из 1

Рис. 4.25: Задание 3.5 (1)

В следующем задании нужно отметить все верные утверждения об опциях команды `find`: `-path` и `-name`. Давайте сделаем это, воспользовавшись документацией об этой команде и её опциях, набрав в терминале `man find` (рис. 4.26).

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

Поэтому, правильный ответ - все кроме file3.

Переходим к следующему заданию: необходимо выбрать ту команду, после выполнения которой создастся файл с наибольшим количеством строк в нём. Для выполнения задания прочитаем мануал по команде `grep` и её опциям `-A`, `-B`, `-C` (рис. 4.28).

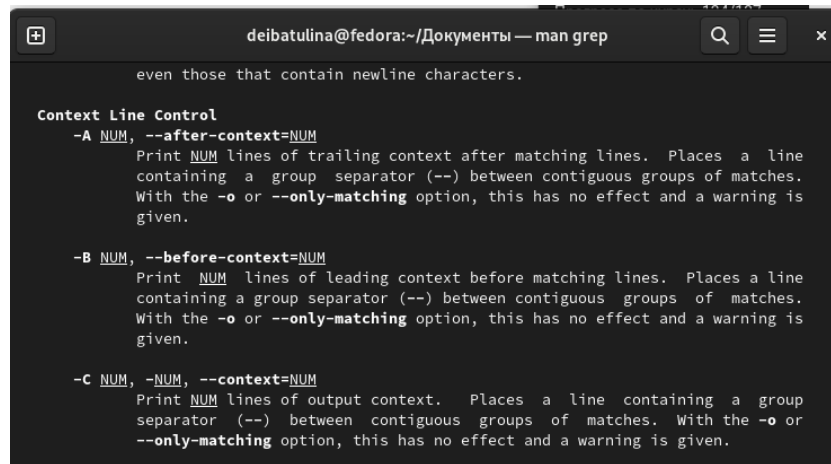


Рис. 4.28: Задание 3.5 (4)

Пятое задание данного блока: (рис. 4.29).

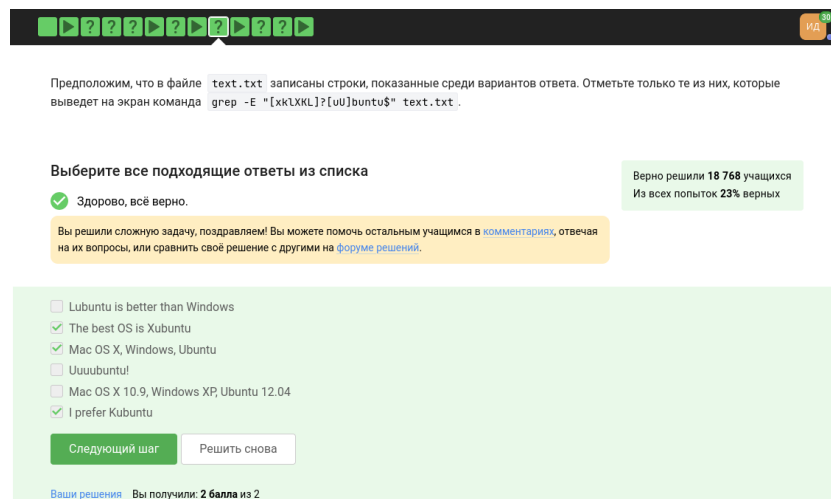


Рис. 4.29: Задание 3.5 (5)

Итак, выполним данное задание у себя в Linux (рис. 4.30).

```
[deibatulina@fedora ~]$ touch text.txt
[deibatulina@fedora ~]$ gedit text.txt
[deibatulina@fedora ~]$ grep -E "[xk\XKL]?[uU]buntu$" text.txt
The best OS is Xubuntu
Mac OS X, Windows, Ubuntu
I prefer Kubuntu
[deibatulina@fedora ~]$
```

Рис. 4.30: Задание 3.5 (5) - проверка на моём Linux

Итак, для выполнения следующего задания (рис. 4.31) нам потребуется знать это:

После обработки каждой строки sed выводит ее в выходной поток. Ключ -n запрещает это, т.е. обработанная строка не будет выводиться второй раз. Если мы уберём опцию -n, то это значит, что каждая строка будет выведена 2 раза.

Вы прошли больше 80% курса, оставьте отзыв

Оставить отзыв Нет, спасибо

Что произойдет, если в команде `sed -n "/[a-z]*/p" text.txt` не указывать опцию `-n`?

Выберите один вариант из списка

✓ Хорошие новости, верно!

Верно решили 19 784 учащихся
Из всех попыток 39% верных

- ☐ На экран ничего не напечатается
- ☐ Будут выведены все строки файла text.txt, в которых есть только большие буквы латинского алфавита
- ☒ Каждая строчка будет выведена два раза
- ☐ Появится сообщение об ошибке

Следующий шаг Решить снова

Ваши решения Вы получили: 1 балл из 1

Рис. 4.31: Задание 3.5 (6)

Делаем последнее задание данного раздела (рис. 4.32):

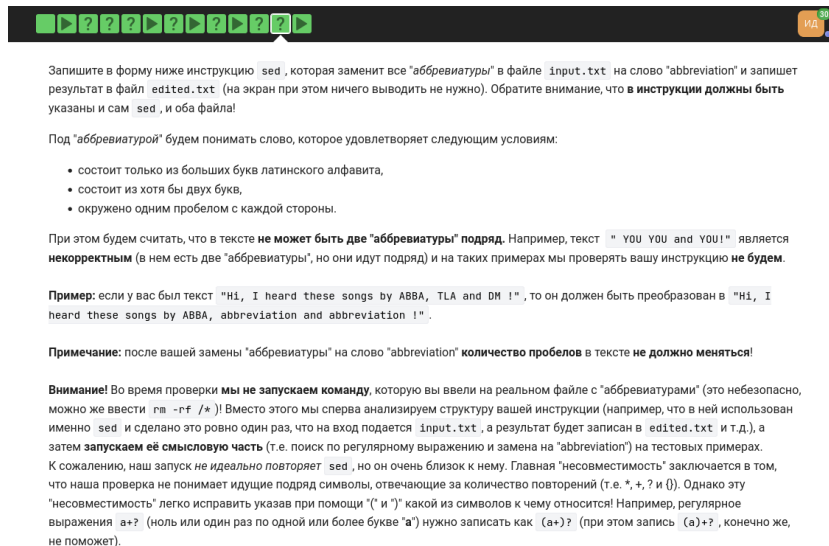


Рис. 4.32: Задание 3.5 (6)

Таким образом, ответ будет таким:

```
sed 's/ [A-Z]\{2,\} / abbreviation /g' input.txt > edited.txt'
```

Объяснение:

' ' регулярное выражение

s замена регулярного выражения

\ перевод строки

{2, \} два и более раз

/ abbreviation на что меняем

/g (global) Операция выполняется над всеми найденными соответствиями

внутри каждой из заданных строк

input.txt > edited.txt файл для чтения > файл для записи

4.6 Строим графики в gnuplot (3.6)

Делаем первое задание данного блока (рис. 4.33):

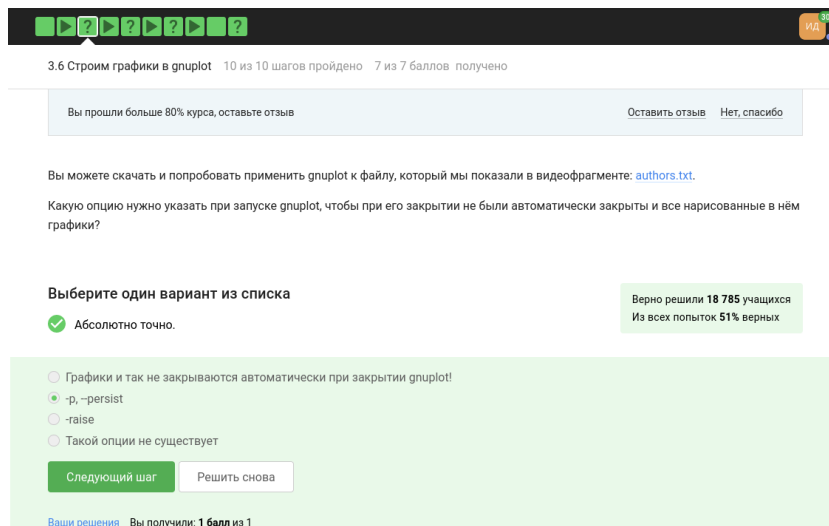


Рис. 4.33: Задание 3.6 (1)

Для его выполнения ознакомимся с мануалом по gnuplot (рис. 4.34):

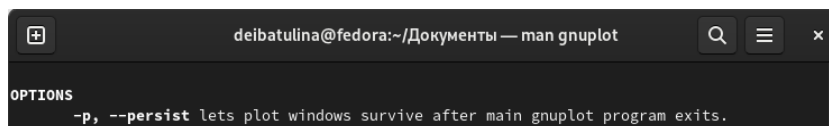


Рис. 4.34: Задание 3.6 (1) - мануал по gnuplot

Идём дальше (рис. 4.35):

12345678910

Вы прошли больше 80% курса, оставьте отзыв

Оставить отзывНет, спасибо

Предположим у вас есть файл `data.csv` с двумя столбцами по 10 чисел в каждом. В первой строке не записаны названия столбцов, т.е. ряды данных начинаются прямо с первой строки. Вы запускаете `gnuplot` и вводите в него две команды:

```
set key autotitle columnhead
plot 'data.csv' using 1:2
```

Какое в этом случае будет **название** у построенного **ряда данных** и **сколько** будет нарисовано **точек** на графике?

Выберите один вариант из списка

Верно решили **17 975** учащихся
Из всех попыток **32%** верных

☒ Правильно, молодец!

Вы решили сложную задачу, поздравляем! Вы можете помочь остальным учащимся в [комментариях](#), отвечая на их вопросы, или сравнить своё решение с другими на [форуме решений](#).

☐ Название – первое значение из второго столбца, нарисовано 9 точек (точка из первой строки пропущена)

☐ Название – первое значение из первого столбца, нарисовано 9 точек (точка из первой строки пропущена)

☐ Название 'noname', нарисовано 10 точек

☐ Название "data.csv" using 1:2, нарисовано 10 точек

☐ Название – первое значение из первого столбца, нарисовано 10 точек

Следующий шаг

Решить снова

Для этого создадим файл data.csv, куда занесём числа, как сказано в условии (рис. 4.36):

	A	B	C
1	13	12	
2	11	10	
3	5	6	
4	7	8	
5	9	10	
6	11	12	
7	13	14	
8	15	16	
9	17	18	
10	19	20	

Рис. 4.36: Задание 3.6 (2) - файл с данными

Введём требуемые команды в gnuplot и получим следующий результат (рис. 4.37):

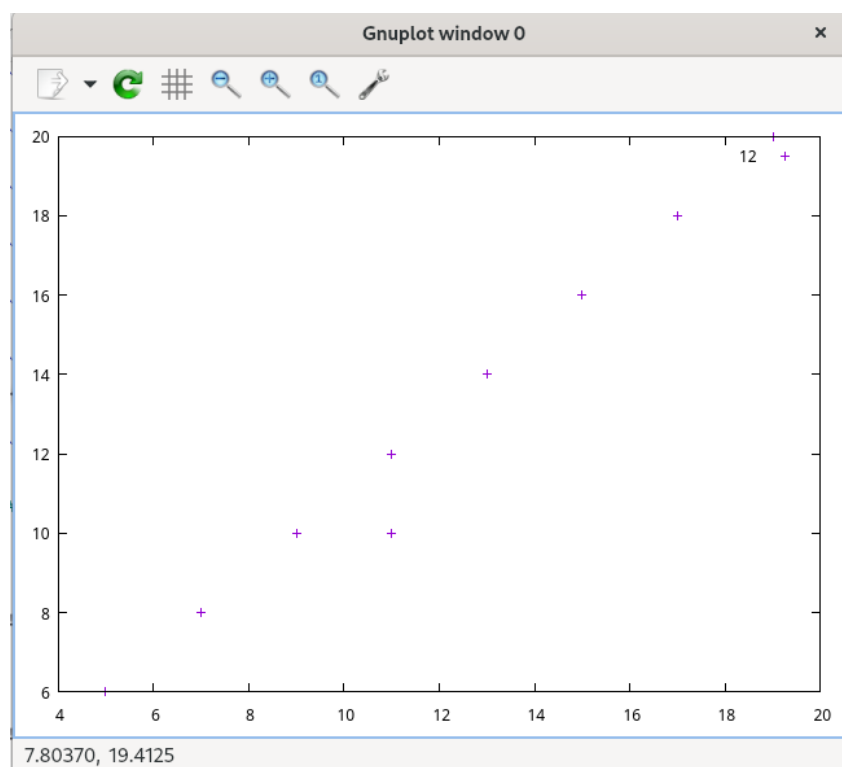


Рис. 4.37: Задание 3.6 (2) - результат

Получаем правильный ответ: будет нарисовано 9 точек, название - первое значение из второго столбца.

Третье задание (рис. 4.38):

3.6 Строим графики в gnpplot 10 из 10 шагов пройдено 7 из 7 баллов получено

Вы прошли больше 80% курса, оставьте отзыв

[Оставить отзыв](#) Нет, спасибо

Вы можете скачать и изучить скрипты, которые мы показали в видеофрагменте: [plot_gnu](#), [plot_advanced_gnu](#), [plot_advanced2.gnu](#). Все три скрипта основаны на [этой заметке](#), данные также взяты оттуда.

Предположим, что вы пишете gnpplot-скрипт и у вас в нем есть три переменные `x1`, `x2`, `x3`, в которых записаны координаты важных точек по оси OX (по возрастанию). Вы хотите, чтобы на этой оси было только три деления (т.е. три черточки) в этих самых координатах, а подписи этих делений были оформлены в виде "point <номер точки>, value <значение соответствующей переменной>".

Например, для `x1=8`, `x2=10`, `x3=20`, это были бы надписи "point 1, value 0" в точке с координатой 0 по горизонтали, "point 2, value 10" в точке с координатой 10 и "point 3, value 20" в точке с координатой 20.

Или, например, `x1=100`, `x2=150`, `x3=250`, это были бы надписи "point 1, value 100" в точке с координатой 100, "point 2, value 150" в точке с координатой 150 и "point 3, value 250" в точке с координатой 250.

Впишите в форму ниже **одну команду** (т.е. одну строку), которую нужно добавить в скрипт, для выполнения этой задачи.

Примечание: проверять, что переменные `x1`, `x2`, `x3` идут по возрастанию или что они являются числами не нужно!

Примечание 2: в видеофрагменте на предыдущем шаге звучал термин *конкатенация*, который важен для выполнения данного задания. Под конкатенацией обычно понимают "склеивание" двух строк в одну длинную строку, например, конкатенация строк "Данные из файла " и "data.csv" даст строку "Данные из файла data.csv".

Подсказка: настоятельно рекомендуем изучить примеры скриптов – в них есть большая часть решения!

3.6 Строим графики в gnuplot 10 из 10 шагов пройдено 7 из 7 баллов получено

Вы прошли больше 80% курса, оставьте отзыв

[Оставить отзыв](#)

Нет, спасибо

Вы можете скачать и изучить скрипты, которые мы показали в видеофрагменте: [plot.gnu](#), [plot.advanced.gnu](#), [plot.advanced2.gnu](#). Все три скрипта основаны на [этой заметке](#), данные также взяты оттуда.

Предположим, что вы пишете `plot`-скрипт и у вас в нем есть три переменные `x1`, `x2`, `x3`, в которых записаны координаты важных точек по оси Ox (по возрастанию). Вы хотите, чтобы на этой оси было только три деления (т.е. три черточки) в этих самых координатах, а подписи этих делений были оформлены в виде `'point <номер точки>, value <значение соответствующей переменной>'`.

Например, для `x1=0`, `x2=10`, `x3=20`, это были бы надписи "point 1, value 0" в точке с координатой 0 по горизонтали, "point 2, value 10" в точке с координатой 10 и "point 3, value 20" в точке с координатой 20.

Или, например, `x1=100, x2=150, x3=250`, это были бы надписи "point 1, value 100" в точке с координатой 100, "point 2, value 150" в точке с координатой 150 и "point 3, value 250" в точке с координатой 250.

Впишите в форму ниже **одну команду** (т.е. одну строку), которую нужно добавить в скрипт, для выполнения этой задачи.

Примечание: проверять, что переменные `x1`, `x2`, `x3` идут по возрастанию или что они являются числами не нужно!

Примечание 2: в видеофрагменте на предыдущем шаге звучал термин *конкатенация*, который важен для выполнения данного задания. Под *конкатенацией* обычно понимают "склеивание" двух строк в одну длинную строку, например, конкатенация строк "Данные из файла " и "data.csv" даст строку "Данные из файла data.csv".

Подсказка: настоятельно рекомендуем изучить примеры скриптов -- в них есть большая часть решения!

Рис. 4.38: Задание 3.6 (3)

Ответ будет выглядеть следующим образом:

set xtics ("point 1, value ".x1 x1, "point 2, value ".x2 x2, "point 3, value ".x3 x3) , где xtics - отсечки на осях, "point 1, value ".x1 x1 - отсечка (подпись, значение) Выполним последнее задание блока (рис. 4.39), изучив все необходимые файлы и инструкции к заданию:

ИД

3.6 Строим графики в gnuplot 10 из 10 шагов пройдено 7 из 7 баллов получено

Вы прошли больше 80% курса, оставьте отзыв

Оставить отзыв Нет, спасибо

Если вы не скачали на предыдущем шаге файлы [animated.gnu](#) и [move.rot](#), то скачайте их теперь, т.к. они понадобятся для выполнения задания.

Указанные файлы использовались в последнем видеофрагменте для создания вращающегося графика. Измените инструкции в файле `move.rot` (т.е. **добавлять** и **удалять** инструкции **нельзя!**) таким образом, чтобы:

- График **отразился зеркально** относительно горизонтальной поверхности. То есть там, где была точка (10, 10, 200), станет точка (10, 10, -200), где была точка (-10, -10, 200) станет (-10, -10, -200) и т.д. При этом точка (0, 0, 0) останется на месте.
- Изображение стало **вращаться в обратную сторону**. То есть если раньше вращалось "влево", то теперь станет "вправо".
- Вращение стало в **два раза быстрее**. То есть станет в два раза больше перерисовок графика на каждую секунду вращения.

Измененный файл загрузите в форму ниже.

Примечание: наша система проверки **не может** запустить на вашем файле `move.rot` программу `gnuplot` и сравнить полученный график с заданным. Вместо этого **мы анализируем команды**, которые вы указали в файле. Поэтому если вы видите, что ваш скрипт в `gnuplot` работает точно по условию, а мы отвечаем "Incorrect/Неверно", то попробуйте упростить свою модификацию `move.rot` и отправить его еще раз.

Напишите текст

Всего решили 12 854 учащихся

Из всех попыток 47% верных

✓ Так точно!

3.6 Строим графики в gnuplot 10 из 10 шагов пройдено 7 из 7 баллов получено

Вы прошли больше 80% курса, оставьте отзыв

[Оставить отзыв](#)

Нет, спасибо

Если вы не скачали на предыдущем шаге файлы [animated.gnu](#) и [move.rot](#), то скачайте их теперь, т.к. они понадобятся для выполнения задания.

Указанные файлы использовались в последнем видеофрагменте для создания вращающегося графика. Измените инструкции в файле `move.rot` (т.е. **добавлять** и **удалять** инструкции **нельзя!**) таким образом, чтобы:

- График **отразился зеркально** относительно горизонтальной поверхности. То есть там, где была точка $(10, 10, 200)$, станет точка $(10, 10, -200)$, где была точка $(-10, -10, 200)$ станет $(-10, -10, -200)$ и т.д. При этом точка $(0, 0, 0)$ останется на месте.
- Изображение стало **вращаться в обратную сторону**. То есть если раньше вращалось «влево», то теперь стало «вправо».
- Вращение стало в **два раза быстрее**. То есть станет в два раза больше переисходов графика на каждую секунду вращения.

Измененный файл загрузите в форму ниже.

Примечание: наша система проверки **не может** запустить на вашем файле `move.rot` программу `gnuplot` и сравнить полученный график с заданным. Вместо этого **мы анализируем команды**, которые вы указали в файле. Поэтому если вы видите, что ваш скрипт в `gnuplot` работает точно по условию, а мы отвечаем "Incorrect/Неверно", то попробуйте упростить свою модификацию `move.rot` и отправить его еще раз.

Напишете текст

✔ Так точно!

Верно решили **12 854** учащихся

Из всех попыток **47%** верных

Рис. 4.39: Задание 3.6 (4)

Таким образом, ответ будет:

```
a=a+1
zrot=(zrot+350)%360
set view xrot,zrot
splot -x**2-y**2
pause 0.1
if (a<50) reread
```

Объяснение:

Итак, скачиваем файл move.rot - и основа у нас есть. Теперь нам нужно подправить ровно три строки чтоб выполнились все условия.

1. График строится строкой “splot x²+y²”. Нам нужно имеющийся график зеркально отобразить. Вспоминаем курс школьной математики (построение функций на графике и их изменение). У нас есть функция $f(x;y)$, на основе которой построить новую функцию $g(x;y)$. $g(x;y) = -(f(x;y))$.
2. Вращение в другую сторону. В этой задаче вращение задается строкой “zrot=(zrot+10)%360”. 360 - это полный круг. Значит наше смещение вперед (которое было изначально) можно также задать и 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. Тогда, это будет: zrot=(zrot+350)%360.
3. Обратим внимание на строку pause 0.2. Она ставит выполнение на паузу на определенный промежуток времени. В задании сказали перерисовывать чаще. Значит пауза должна быть меньше.

4.7 Разное (3.7)

Выполняем первое задание, ориентируясь на знания, полученные в ходе выполнения лабораторной работы по Операционным системам, связанной с присвоением прав для файла (рис. 4.40):

Вы прошли больше 80% курса, оставьте отзыв

Оставить отзыв Нет, спасибо

Какая команда(ы) установят файлу `file.txt` права доступа `rwXrwx-r--`, если изначально у него были права `r--r--r--`. Укажите **все верные** варианты ответа!

Примечание: запись вида `команда1; команда2; команда3` означает, что в терминале последовательно выполнились все три команды (сначала `команда1`, затем `команда2` и, наконец, `команда3`).

Выберите все подходящие ответы из списка

✓ Хорошая работа.

Верно решили 16 484 учащихся
Из всех попыток 21% верных

Вы решили сложную задачу, поздравляем! Вы можете помочь остальным учащимся в [комментариях](#), отвечая на их вопросы, или сравнить своё решение с другими на [форуме решений](#).

- ☒ `chmod 764 file.txt`
- ☒ `chmod ug+w file.txt; chmod u+x file.txt`
- ☒ `chmod a+wx file.txt; chmod o-wx file.txt; chmod g-x file.txt`
- ☐ `chmod 777 file.txt`
- ☐ `chmod o-wx file.txt; chmod g-x file.txt; chmod a+wx file.txt`
- ☒ `chmod u+wx file.txt; chmod g+w file.txt`

Следующий шаг Решить снова

Ваши решения Вы получили: 1 балл из 1

Рис. 4.40: Задание 3.7 (1)

Делаем следующее задание:

Предположим вы использовали команду `sudo` для создания директории `dir`. По умолчанию для `dir` были выставлены права доступа `rwXr-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!

Варианты ответов:

- `chmod o+w dir`
- `sudo chown user dir`

- `chown user:group dir`
- `sudo chmod g+w dir`
- `sudo chown user:group dir`
- `sudo chown :group dir`

Нам подходят ответы 2 и 5, их и выбираем. Система засчитывает их за правильные!

Задание 3 данного блока (рис. 4.41):

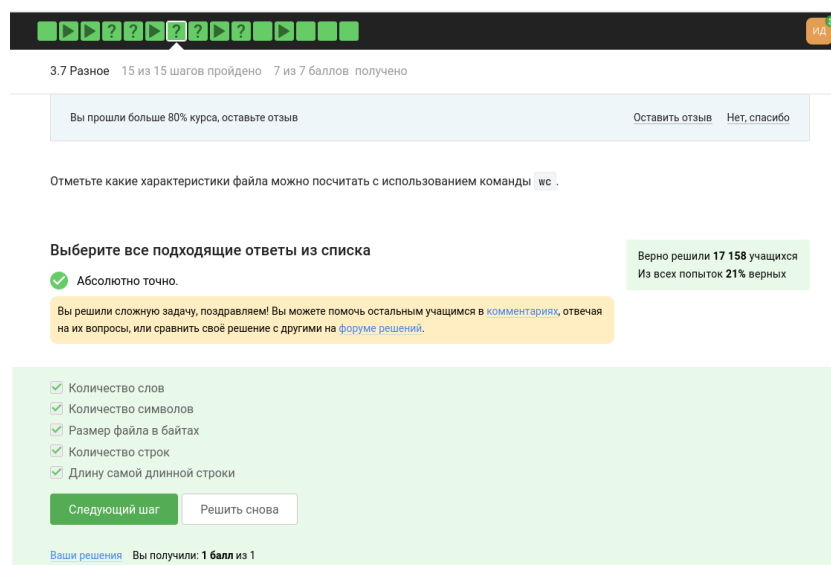


Рис. 4.41: Задание 3.7 (3)

Решила я его, воспользовавшись информацией, которую получила при просмотре лекции.

Далее, задание 4 (рис. 4.42):

3.7 Разное 15 из 15 шагов пройдено 7 из 7 баллов получено

Вы прошли больше 80% курса, оставьте отзыв [Оставить отзыв](#) [Нет, спасибо](#)

Впишите в форму ниже команду, которая выведет сколько места на диске занимает текущая директория (при этом **размер** нужно вывести в **удобном для чтения формате** (например, вместо 2048 байт надо вывести 2.0К) и **больше** на экран выводить **ничего не** нужно). В команде указывайте **только необходимые** для выполнения задания **опции** и **аргументы**, лишних опций указывать не нужно!

Пример: если в текущей директории есть два файла по 800 Кбайт и две поддиректории в каждой из которой лежит по файлу в 400 Кбайт, то загаданная команда должна вывести на экран одно число: 2.4М (также на экране может быть выведен еще и символ "", обозначающий, что это размер именно текущей директории).

Напишите текст

✓ Отличное решение!

Верно решил 16 381 учащийся
Из всех попыток 53% верных

`du -h -s`

[Следующий шаг](#) [Решить снова](#)

[Ваши решения](#) Вы получили: 2 балла из 2

Рис. 4.42: Задание 3.7 (4)

Ответ я нашла, используя мануал по команде `du`. Объяснение таково: стандартная команда `du` + человекочитаемый формат + глубина уровней.

Последнее задание блока: необходимо написать как можно более короткую команду для того, чтобы в текущей директории создать 3 каталога с именами: `dir1`, `dir2`, `dir3` (рис. 4.43):

3.7 Разное 15 из 15 шагов пройдено 7 из 7 баллов получено

Вы прошли больше 80% курса, оставьте отзыв [Оставить отзыв](#) [Нет, спасибо](#)

Впишите в форму ниже максимально короткую команду (т.е. в которой минимально возможное число символов), которая позволит создать в текущей директории 3 поддиректории с именами `dir1`, `dir2`, `dir3`.

Если вы придумали команду, которая выполняет эту задачу, а система проверки сообщает вам "Incorrect"/"Неверно", то скорее всего вы придумали не самую короткую команду из возможных!

Напишите текст

✓ Правильно, молодец!

Верно решили 16 720 учащихся
Из всех попыток 40% верных

`mkdir dir{1..3}`

[Следующий шаг](#) [Решить снова](#)

[Ваши решения](#) Вы получили: 2 балла из 2

Рис. 4.43: Задание 3.7 (5)

Задача решена с использованием массива, в который мы заносим числа 1, 2, 3, не ставя между ними пробел. А в названиях всех трёх директорий, которые нужно создать, есть общее слово `dir`, его мы и пишем перед массивом. Для создания директорий существует команда `mkdir`.

5 Выводы

В результате выполнения данного этапа курса я научилась минимальным навыкам написания скриптов на `bash`, познакомилась с интерфейсом текстового редактора `vim` и порисовала графики в `gnuplot`.

Список литературы