

Отчёт по прохождению внешнего курса

3 этап

Ярослав Антонович Меркулов

Содержание

1	Цель работы	4
2	Выполнение этапа	5
3	Выводы	17

Список иллюстраций

2.1	Задание 1	5
2.2	Задание 2	6
2.3	Задание 3	6
2.4	Задание 4	7
2.5	Задание 5	8
2.6	Задание 6	8
2.7	Задание 7	9
2.8	Задание 8	10
2.9	Задание 9	11
2.10	Задание 10	12
2.11	Задание 14	14
2.12	Задание 15	14
2.13	Задание 16	15
2.14	Задание 17	15
2.15	Задание 18	16

1 Цель работы

Пройти третий этап внешнего курса “Введение в Linux”.

2 Выполнение этапа

1. Теоретический вопрос (рис. 2.1).

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.

Выберите один вариант из списка

Верно решили 32 523 учащихся
Из всех попыток 69% верных

☒ Отличное решение!

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

Следующий шаг Решить снова

[Ваши решения](#) Вы получили: 1 балл

Рис. 2.1: Задание 1

2. Вот объяснение команды :%s/Windows/Linux в vim:

: — вводит командный режим в vim. % — означает, что команда применяется ко всему файлу (все строки). s — команда замены (substitute). /Windows/ — шаблон, который ищется в строке; в данном случае слово “Windows”. /Linux — текст, на который заменяется найденный шаблон; в данном случае “Linux”.(рис. 2.2).

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**.

Напишите текст

✓ Всё правильно.

Верно решил 24 631 учащийся
Из всех попыток 57% верных

Следующий шаг
Решить снова

Ваши решения Вы получили: 2 балла

Рис. 2.2: Задание 2

- Команды из набора А (первой оболочки `bash`) не будут доступны, потому что они были введены в другой сессии. Команды из набора В (оболочка `sh`) не будут доступны, потому что они были введены в другой сессии `sh`. Команды из набора С (последняя запущенная оболочка `bash`) будут доступны при перемещении по истории.(рис. 2.3).

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

✓ Хорошая работа.

Верно решили 30 266 учащихся
Из всех попыток 65% верных

☐ Из наборов В и С
 ☐ Из наборов А и С
 ☐ Никакие команды появляться не будут
 ☒ Только из набора С
 ☐ Только из набора А

Следующий шаг
Решить снова

Ваши решения Вы получили: 1 балл

Рис. 2.3: Задание 3

- Файл создан в каталоге `/home/bi`. Потом просто перешли в другой каталог. Это не поменяло путь к файлу (рис. 2.4).

Вы можете скачать и изучить скрипты, которые мы показали в видеофрагменте: [script1.sh](#), [script2.sh](#).

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

```
#!/bin/bash
cd /home/bi/
touch file1.txt
cd /home/bi/Desktop/
```

Как будет выглядеть **абсолютный путь** до созданного файла `file1.txt` по окончании работы скрипта?

Выберите один вариант из списка

✓ Хорошие новости, верно!

Верно решили 29 905 учащихся
Из всех попыток 76% верных

- ☒ `/home/bi/file1.txt`
- ☐ Никак (файла `file1.txt` не будет существовать после завершения работы скрипта)
- ☐ `/home/bi/Desktop/file1.txt`
- ☐ `/home/bi/Documents/file1.txt`

Следующий шаг Решить снова

Ваши решения Вы получили: 1 балл

Рис. 2.4: Задание 4

5. VARiable

Начинается с буквы, содержит только буквы. Подходит `123variable`

Начинается с цифры. Не подходит `variable_123`

Начинается с буквы, содержит буквы, цифры и подчеркивание. Подходит `variable`

Начинается с буквы, только буквы. Подходит `variable123`

Начинается с буквы, содержит буквы и цифры. Подходит `_variable`

Начинается с подчеркивания, что допустимо. Подходит `__variable`

Начинается с двух подчеркиваний, что тоже допустимо. Подходит (рис. 2.5).

Вы можете скачать и изучить скрипты, которые мы показали в видеофрагменте: [variables1.sh](#), [variables2.sh](#).

Какие из представленных ниже строк **могут** быть именами переменных в bash? Выберите **все** подходящие варианты!

Подсказка: если все варианты ответов являются неверными, то не отмечайте ни один из них и нажимайте кнопку "Отправить"/"Submit".

Выберите все подходящие ответы из списка

Верно решили 27 188 учащихся
Из всех попыток 25% верных

✓ Верно. Так держать!

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](#).

☒ VARiable
☐ 123variable
☒ variable_123
☒ variable
☒ variable123
☒ _variable
☒ __variable

Следующий шаг
 Решить снова

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

Рис. 2.5: Задание 5

6. Легкий скрипт, который выводит то, что получает(рис. 2.6).

Вы можете скачать и изучить скрипт, который мы показали в видеофрагменте: [arguments.sh](#).

Напишите скрипт на bash, который принимает на вход два аргумента и выводит на экран строку следующего вида:

```
Arguments are: $1=первый_аргумент $2=второй_аргумент
```

Например, если ваш скрипт называется `./script.sh`, то при запуске его `./script.sh one two` на экране должно появиться:

```
Arguments are: $1=one $2=two
```

а при запуске `./script.sh three four` будет:

```
Arguments are: $1=three $2=four
```

Подсказка: в случае проблем с решением задачи, обратите внимание на [наши рекомендации по написанию скриптов](#).

Напишите программу. Тестируется через stdin → stdout

✓ Так точно!

Теперь вам доступен [Форум решений](#), где вы можете сравнить свое решение с другими или спросить совета.

Верно решили 25 053 учащихся
Из всех попыток 41% верных

```

1 echo "Arguments are:" \ $1=$1 \ $2=$2
2
3
4
5
6
7
8
  
```

Рис. 2.6: Задание 6

7. Довольно сложное задание. На решение понадобилось достаточно много времени.(рис. 2.7).

Вы можете скачать и изучить скрипт, который мы показали в видеофрагменте: [branching1.sh](#).

Предположим, вы пишете скрипт на bash и хотите использовать в нем конструкцию `if` в следующем фрагменте:

```
if [[ ... ]]
then
  echo "True"
fi
```

Вы можете вписать вместо `"..."` (внутри `[[...]]` и не забудьте про пробелы после `[[` и перед `]]`) любое из перечисленных ниже условий. Однако мы просим вас выбрать только те из них, при которых `echo` напечатает на экран `True` вне зависимости от того, с какими параметрами был запущен ваш скрипт и какие в нем есть переменные.

Например, условие `0 -eq 0` **подходит**, т.к. ноль всегда равен нулю вне зависимости от аргументов и переменных внутри скрипта и на экран будет напечатано `True`. В то же время условие `$var1 -eq 0` **не подходит**, так как в переменной `var1` как может быть записан ноль (тогда будет напечатано `True`), так его может и не быть (тогда ничего напечатано не будет).

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.

Выберите все подходящие ответы из списка

Верно решили 23 156 учащихся
Из всех попыток 16% 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.

- ☐ `$var1 == $var2 && $var1 != $var2`
- ☒ `-n $0`
- ☐ `-n $1`
- ☐ `$# -gt 0`
- ☒ `! (4 -le 3)`
- ☒ `$# -ge 0`

Следующий шаг Решить снова

Рис. 2.7: Задание 7

8. Первый запуск: `var=3` Подставим `var=3` в условия:

`if [] — 3 > 5?` Нет, условие ложно. `elif [] — 3 3?` Нет, равно, условие ложно. `elif [] — 3 == 4?` Нет, условие ложно. Все условия ложны, значит выполнится блок `else`:
`echo "four"`

Второй запуск: `var=5` Подставим `var=5`:

`if [] — 5 > 5?` Нет, равно, условие ложно. `elif [] — 5 3?` Нет, условие ложно. `elif [] — 5 == 4?` Нет, условие ложно. Все условия ложны, снова выполняется блок `else`:
`echo "four"`(рис. 2.8).

Вы можете скачать и изучить скрипты, которые мы показали в видеофрагменте: [branching2.sh](#), [branching3.sh](#).

Посмотрите на фрагмент bash-скрипта:

```
if [[ $var -gt 5 ]]
then
  echo "one"
elif [[ $var -lt 3 ]]
then
  echo "two"
elif [[ $var -eq 4 ]]
then
  echo "three"
else
  echo "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`.

Выберите один вариант из списка

✓ Хорошие новости, верно!

Верно решили 25 138 учащихся
Из всех попыток 64% верных

- ☒ Сначала four, потом four
- ☐ Сначала one, потом two
- ☐ Сначала two, потом four
- ☐ Сначала four, потом one

Рис. 2.8: Задание 8

9. Простая программа на `bash`, с обычными ветвлениями через `if/elif/else`(рис. 2.9).

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 б): 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 появиться:

```
A lot of students
```

Подсказка: в случае проблем с решением задачи, обратите внимание на [наши рекомендации по написанию скриптов](#).

Напишите программу. Тестируется через stdin → stdout

✓ Верно.

Верно решили 23 310 учащихся
Из всех попыток 38% верных

Теперь вам доступен [Форум решений](#), где вы можете сравнить свое решение с другими или спросить совета.

```
1 if [[ $1 -eq 1 ]]; then
2   echo "$1 student"
3 elif [[ $1 -gt 1 && $1 -le 4 ]]; then
4   echo "$1 students"
5 elif [[ $1 -ge 5 ]]; then
6   echo "A lot of students"
7 else
8   echo "No students"
9 fi
10
```

Рис. 2.9: Задание 9

10. Подсчитаем сколько раз выводятся слова на каждой итерации:

1 a start Нет finish 2 , start Нет finish 3 b start Нет finish 4 , start Нет finish 5 c_d
start Да continue (пропуск finish)
(рис. 2.10).

Вы можете скачать и изучить скрипты, которые мы показали в видеофрагменте: [loops1.sh](#), [loops2.sh](#).

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

☒ Отличное решение!

Верно решили 24 582 уча
Из всех попыток 45% вер

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

Следующий шаг

Решить снова

Рис. 2.10: Задание 10

11. Опять же довольно простая программа. Надо только написать бесконечный цикл, прописать условия выхода(рис. ??).

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
```

```
1 child=16
2 adult=25
3 stdout=0
4
5 while [[ $stdout != 1 ]]
6 do
7     echo "enter your name: "
8     read name
9     if [[ (-z $name) || ($name = 0) ]]; then
10         echo "bye"
11         stdout=1
12     elif [[ -n $name ]]; then
13         while [[ $stdout != 1 ]]; do
14             echo "enter your age: "
15             read age
16             if [[ ($age -eq 0) || (-z $age) ]]; then
17                 echo "bye"
18                 stdout=1
19             elif [[ $age -le $child ]]; then
20                 echo "$name, your group is child"
21             elif [[ $age -gt $adult ]]; then
22                 echo "$name, your group is adult"; else
23                 if [[ ($age -ge 17) && ($age -le 25) ]]; then
24                     echo "$name, your group is youth"; fi
25             fi ;break
26         done ;fi
27 done
```

12. Программа для поиска НОД. Уже более сложная. Так же используется бесконечный цикл, if else, while.(рис. ??).

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 ждать 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 **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**).

Пример корректной работы скрипта:

```
./script.sh
10 15
GCD is 5
7 3
GCD 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.

```
1 # put your shell (bash) code here
2
3 while [ true ]
4 do
5     read n1 n2
6     if [ -z $n1 ]; then
7         echo "bye"
8         break
9     else
10        gcd () {
11            remainder=1
12            if [ $n2 -eq 0 ]
13            then
14                echo "bye"
15            fi
16            while [ $remainder -ne 0 ]
17            do
18                remainder=$((n1%n2))
19                n1=$n2
20                n2=$remainder
21            done
22        }
23        gcd $1 $2
24        echo "GCD is $n1"
25    fi
26 done
27
28
29
```

13. Калькулятор на bash(рис. ??).

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:

```
11
1024
bye
```

Если же на вход поступит следующий файл:

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

```
1 # put your shell (bash) code here
2 #!/bin/bash
3 while [[ True ]]
4 do
5     read birinchi amal ikkinchi
6     if [[ $birinchi == "exit" ]]
7     then
8         echo "bye"
9         break
10    elif [[ "$birinchi" =~ "^[0-9]+$" && "$ikkinchi" =~ "^[0-9]+$" ]]
11    then
12        echo "error"
13        break
14    else
15        case $amal in
16            "+") let "result = birinchi + ikkinchi";;
17            "-") let "result = birinchi - ikkinchi";;
18            "/" ) let "result = birinchi / ikkinchi";;
19            "*" ) let "result = birinchi * ikkinchi";;
20            "%" ) let "result = birinchi % ikkinchi";;
21            "**") let "result = birinchi ** ikkinchi";;
22            *) echo "error" ; break ;;
23        esac
24        echo "$result"
25    fi
26 done
27
28
29
30
```

14. Объяснение параметров команды: -mindepth 2: искать файлы, начиная со второго уровня вложенности относительно /home/bi. -maxdepth 3: искать только до третьего уровня вложенности. -name "file*": искать файлы, имена которых начинаются с "file". Расположение файлов по уровням: /home/bi/dir1 – уровень 1 (относительно /home/bi) /home/bi/dir1/file1 – уровень 2 /home/bi/dir1/dir2 – уровень 2 /home/bi/dir1/dir2/file2 – уровень 3 /home/bi/dir1/dir2/dir3 – уровень 3 /home/bi/dir1/dir2/dir3/file3 – уровень 4

Анализ по условиям: `-mindepth 2`: ищем начиная со второго уровня, то есть начиная с `/home/bi/...` внутри. `-maxdepth 3`: ищем только до третьего уровня. Это значит, что ищем файлы, расположенные на уровнях 2 и 3.

Какие файлы подходят? `file1`: находится в `/home/bi/dir1/file1`, уровень 2 → подходит. `file2`: находится в `/home/bi/dir1/dir2/file2`, уровень 3 → подходит. `file3`: находится в `/home/bi/dir1/dir2/dir3/file3`, уровень 4 → не подходит, так как `maxdepth=3`. (рис. 2.11).

Предположим, что в директории `/home/bi/` есть следующая структура файлов и поддиректорий:

```
/home/bi/
├── dir1
│   ├── file1
│   └── dir2
│       ├── file2
│       └── dir3
│           └── file3
```

Какие(ой) из трех файлов (`file1`, `file2`, `file3`) будут найдены по команде `find /home/bi -mindepth 2 -maxdepth 3 -name "file*" ?`

Выберите один вариант из списка

✔ Отлично!

Верно решили 20 711 учащихся
Из всех попыток 41% верных

☐ Все три файла

☐ Только `file3`

☒ Все кроме `file3`

☐ Только `file2`

☐ Все кроме `file1`

Следующий шаг Решить снова

Рис. 2.11: Задание 14

15. Без `-n`, `sed` по умолчанию печатает каждую строку после обработки, независимо от команд внутри скрипта. (рис. 2.12).

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

Выберите один вариант из списка

✔ Верно.

Верно решили 19 784 учащихся
Из всех попыток 39% верных

☐ Появится сообщение об ошибке

☐ Будут выведены все строки файла `text.txt`, в которых есть только большие буквы латинского алфавита

☒ Каждая строчка будет выведена два раза

☐ На экран ничего не напечатается

Следующий шаг Решить снова

[Ваши решения](#) Вы получили: 1 балл

Рис. 2.12: Задание 15

16. Инструкция sed(рис. 2.13).

Запишите в форму ниже инструкцию `sed`, которая заменит все "аббревиатуры" в файле `input.txt` на слово "abbreviation" и запишет результат в файл `edited.txt` (на экран при этом ничего выводить не нужно). Обратите внимание, что в **инструкции должны быть** указаны и сам `sed`, и оба файла!

Под "аббревиатурой" будем понимать слово, которое удовлетворяет следующим условиям:

- состоит только из больших букв латинского алфавита,
- состоит из хотя бы двух букв,
- окружено одним пробелом с каждой стороны.

При этом будем считать, что в тексте **не может быть две "аббревиатуры" подряд**. Например, текст `" YOU YOU and YOU!"` является **некорректным** (в нем есть две "аббревиатуры", но они идут подряд) и на таких примерах мы проверять вашу инструкцию **не будем**.

Пример: если у вас был текст `"Hi, I heard these songs by ABBA, TLA and DM !"`, то он должен быть преобразован в `"Hi, I heard these songs by ABBA, abbreviation and abbreviation !"`.

Примечание: после вашей замены "аббревиатуры" на слово "abbreviation" **количество пробелов** в тексте **не должно меняться!**

Внимание! Во время проверки **мы не запускаем команду**, которую вы ввели на реальном файле с "аббревиатурами" (это небезопасно, можно же ввести `rm -rf /*`)! Вместо этого мы сперва анализируем структуру вашей инструкции (например, что в ней использован именно `sed` и сделано это ровно один раз, что на вход подается `input.txt`, а результат будет записан в `edited.txt` и т.д.), а затем **запускаем её смысловую часть** (т.е. поиск по регулярному выражению и замена на "abbreviation") на тестовых примерах. К сожалению, наш запуск **не идеально повторяет** `sed`, но он очень близок к нему. Главная "несовместимость" заключается в том, что наша проверка не понимает идущие подряд символы, отвечающие за количество повторов (т.е. `*`, `+`, `?` и `{}`). Однако эту "несовместимость" легко исправить указав при помощи `"(и ")"` какой из символов к чему относится! Например, регулярное выражение `a+?` (ноль или один раз по одной или более букве "a") нужно записать как `(a+)?` (при этом запись `(a)+?`, конечно же, не поможет).

Напишите текст

✓ Правильно, молодец!

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](#).

Верно решили 16 632 учащихся
Из всех попыток 34% верных

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

Рис. 2.13: Задание 16

17. gnuplot(рис. 2.14).

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 `gnuplot`-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!

Напишите текст

✓ Всё правильно.

Верно решили 13 935 учащихся
Из всех попыток 44% верных

```
set xtics ("point 1, value "x1,"point 2, value "x2,"point 3, value "x3 x3)
```

Рис. 2.14: Задание 17

18. Меняем работу программы под условие(рис. 2.15).

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

✔ Отлично!

Верно решили **12 854** учащихся
Из всех попыток **47%** верных

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

Рис. 2.15: Задание 18

3 Выводы

Были получены знания о Линуксе: vim, bash, gnuplot. Были выполнены тесты.