Отчёт по прохождению внешнего курса

3 этап

Ярослав Антонович Меркулов

Содержание

Список иллюстраций

# 1 Цель работы

Пройти третий этап внешнего курса “Введение в Linux”.

# 2 Выполнение этапа

1. Теоретический вопрос (рис. 1).

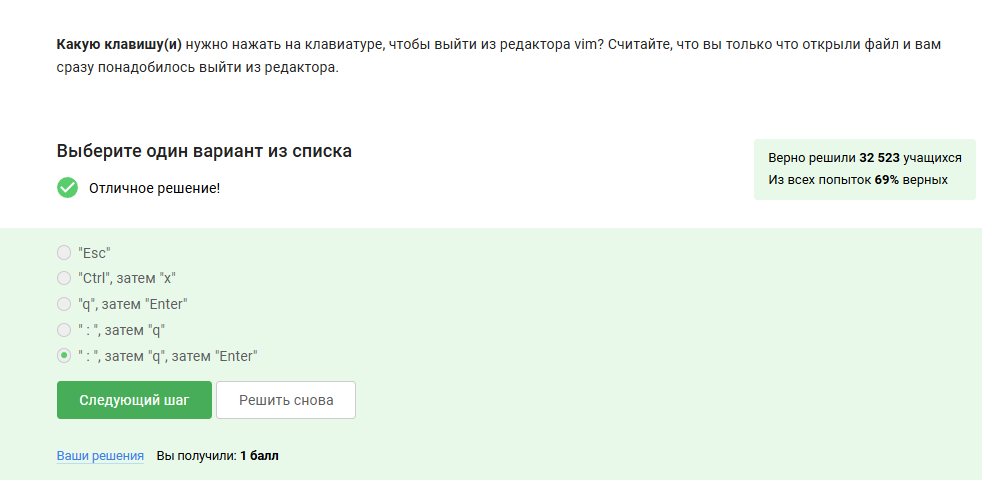


Рис. 1: Задание 1

1. 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).

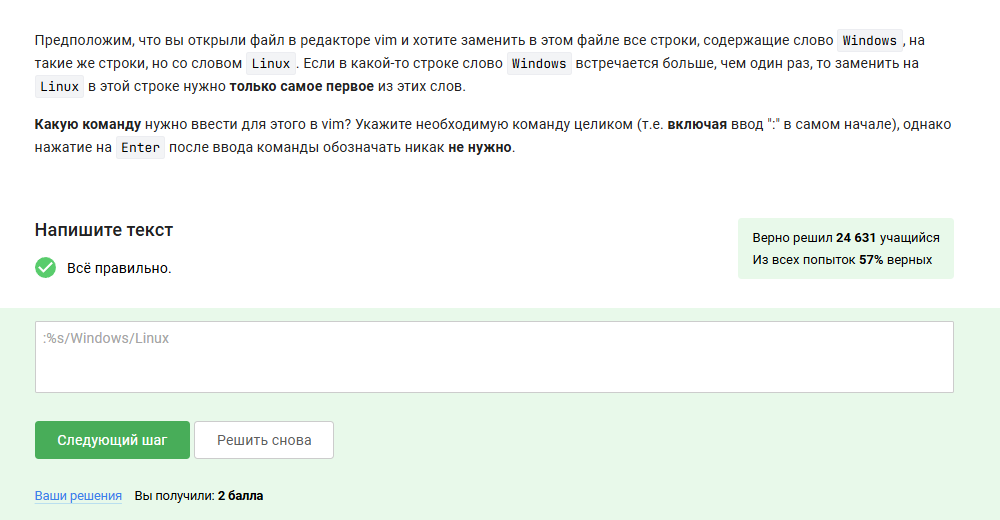


Рис. 2: Задание 2

1. 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. 3).

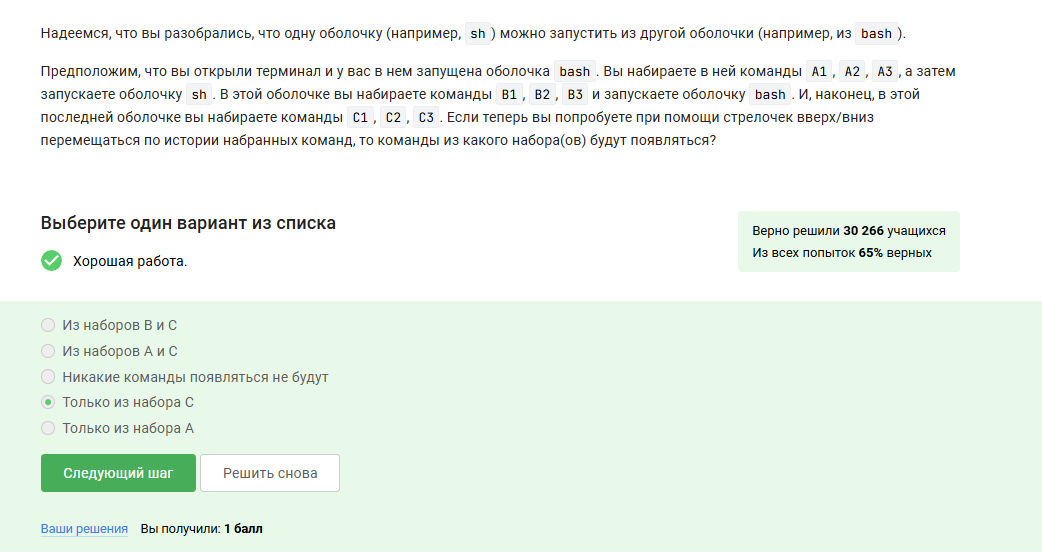


Рис. 3: Задание 3

1. Файл создан в каталоге /home/bi. Потом просто перешли в другой каталог. Это не поменяло путь к файлу (рис. 4).

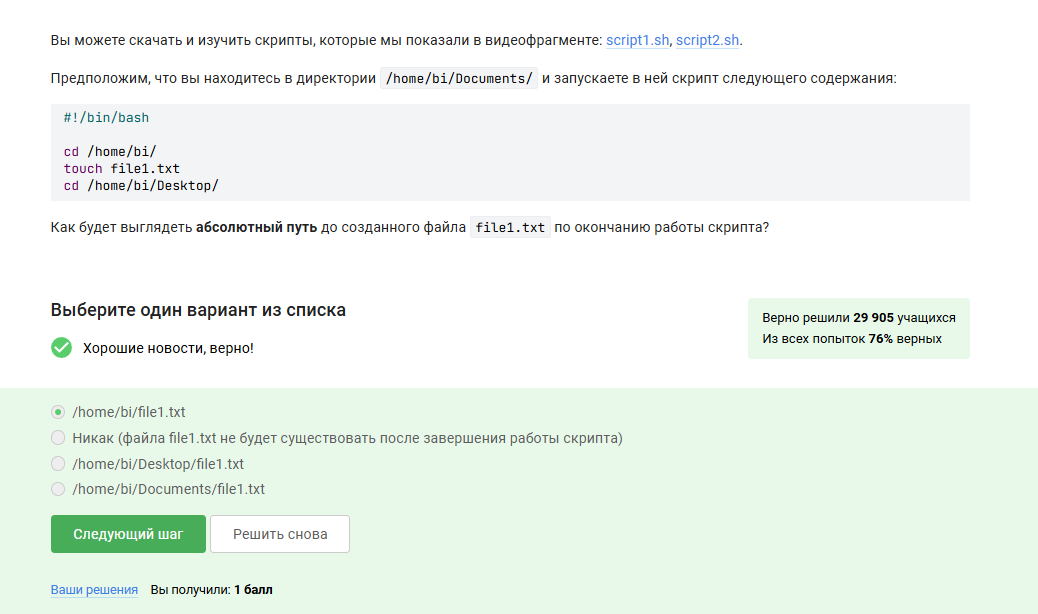


Рис. 4: Задание 4

1. 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. 5).

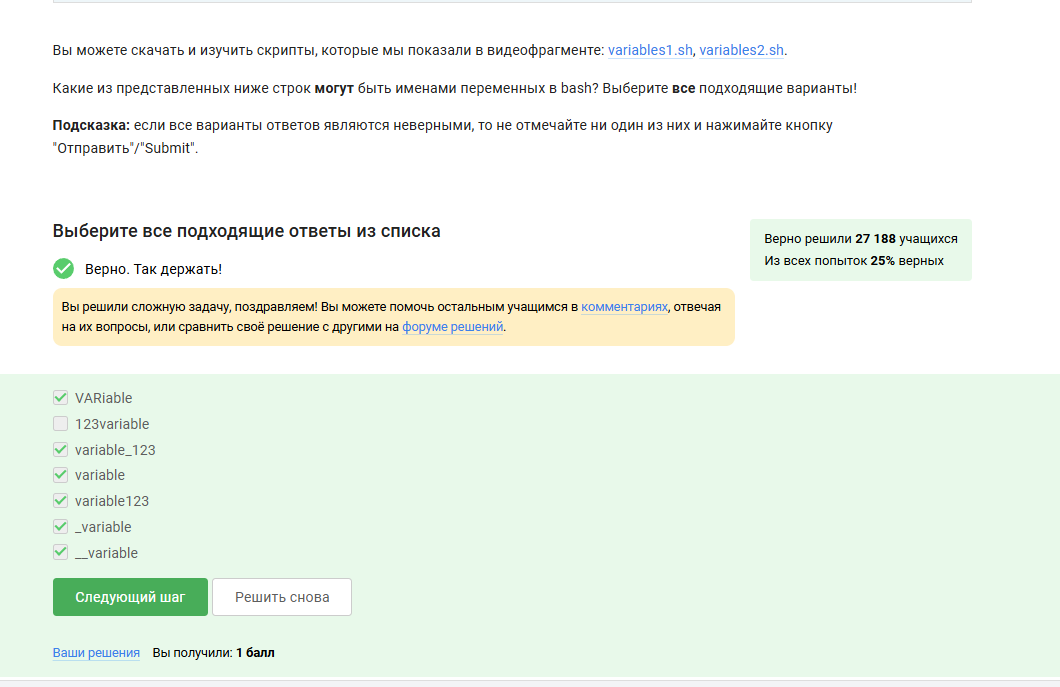


Рис. 5: Задание 5

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

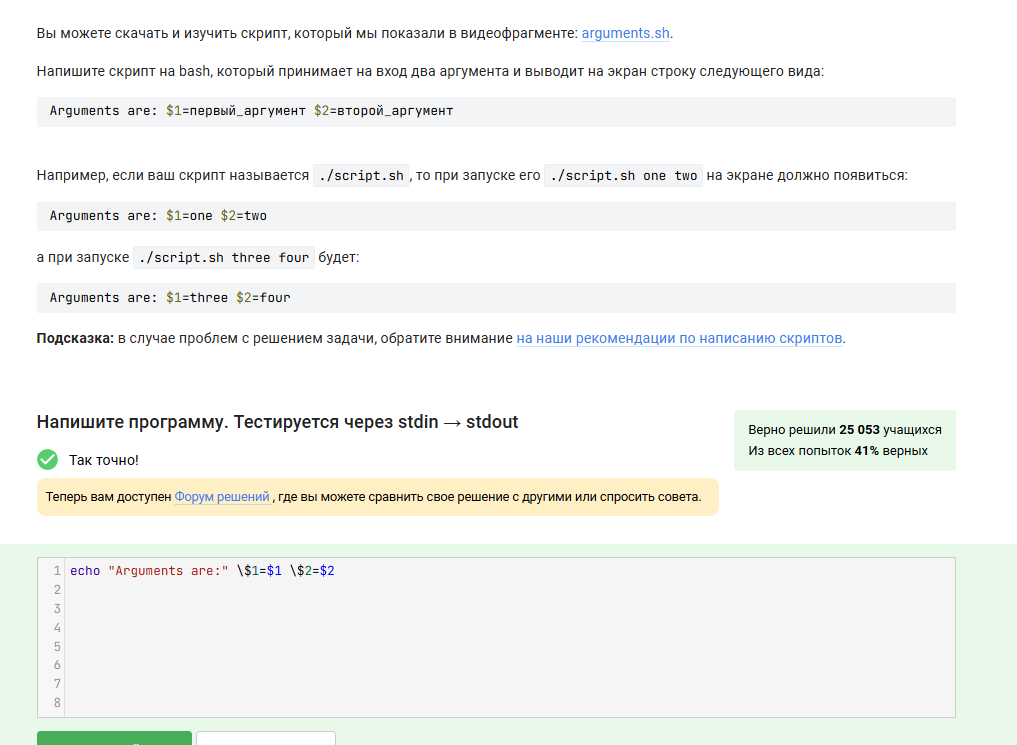


Рис. 6: Задание 6

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

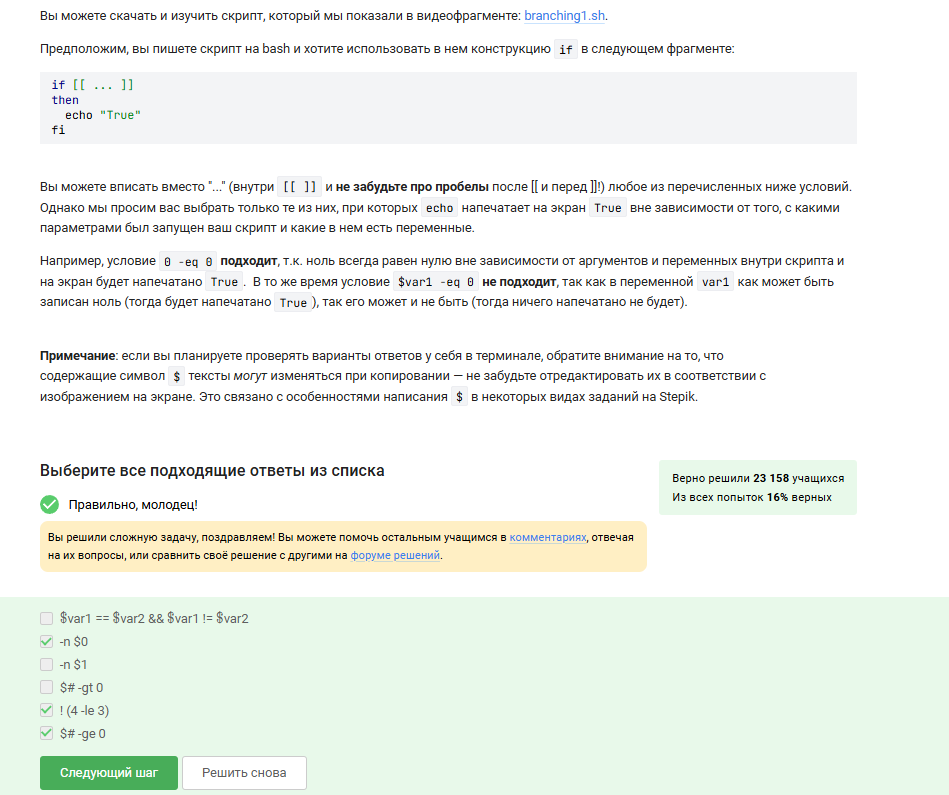


Рис. 7: Задание 7

1. 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. 8).

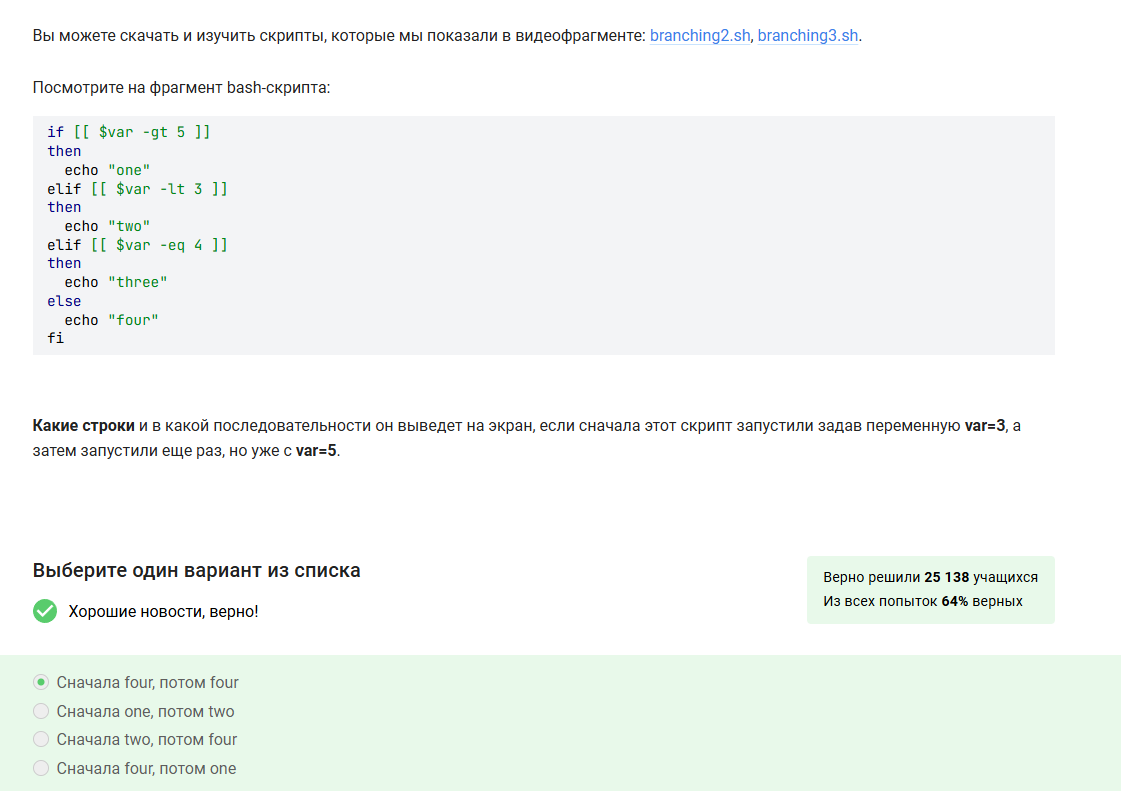


Рис. 8: Задание 8

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

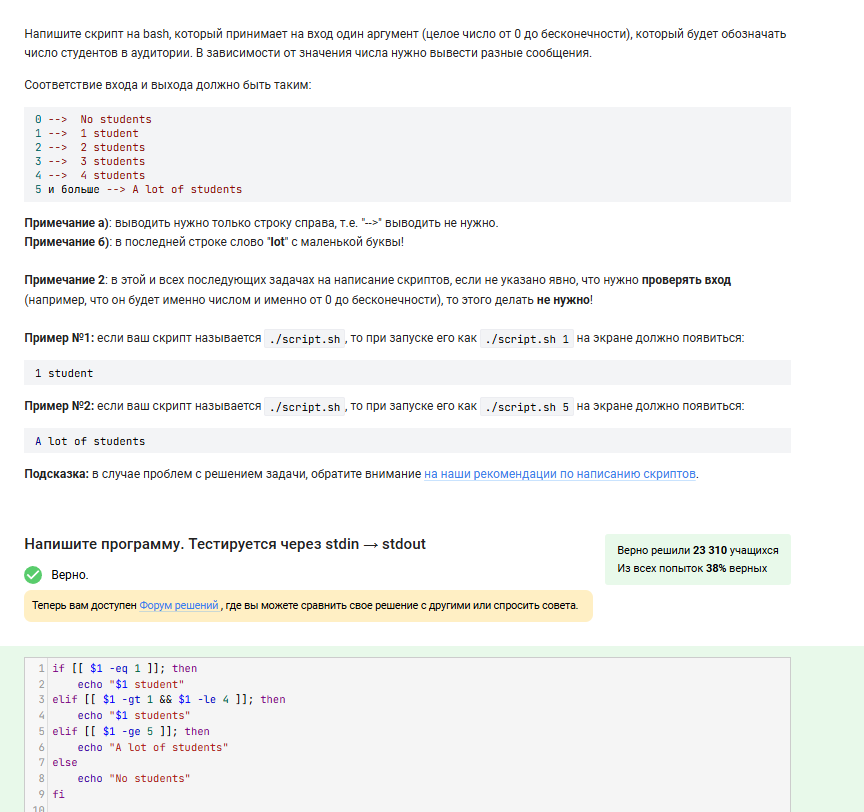


Рис. 9: Задание 9

1. 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. 10).

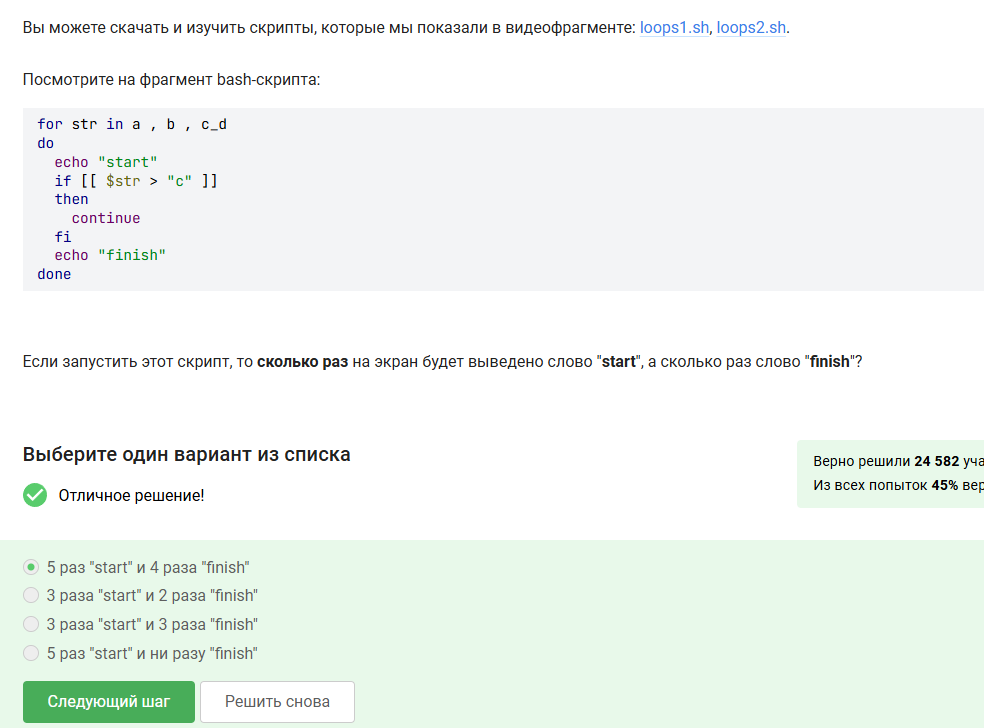
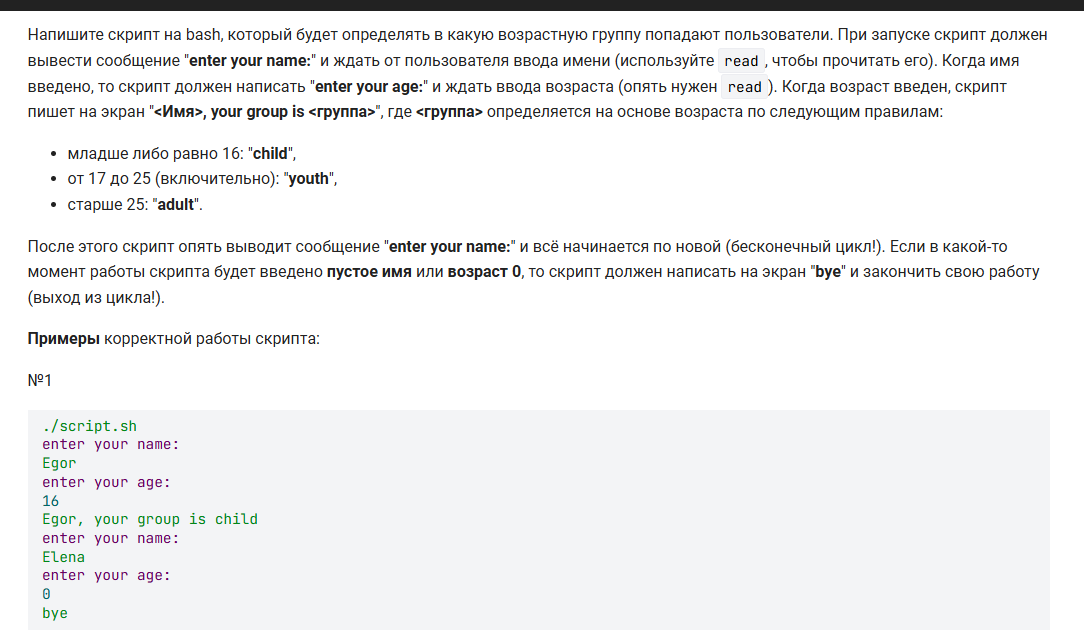
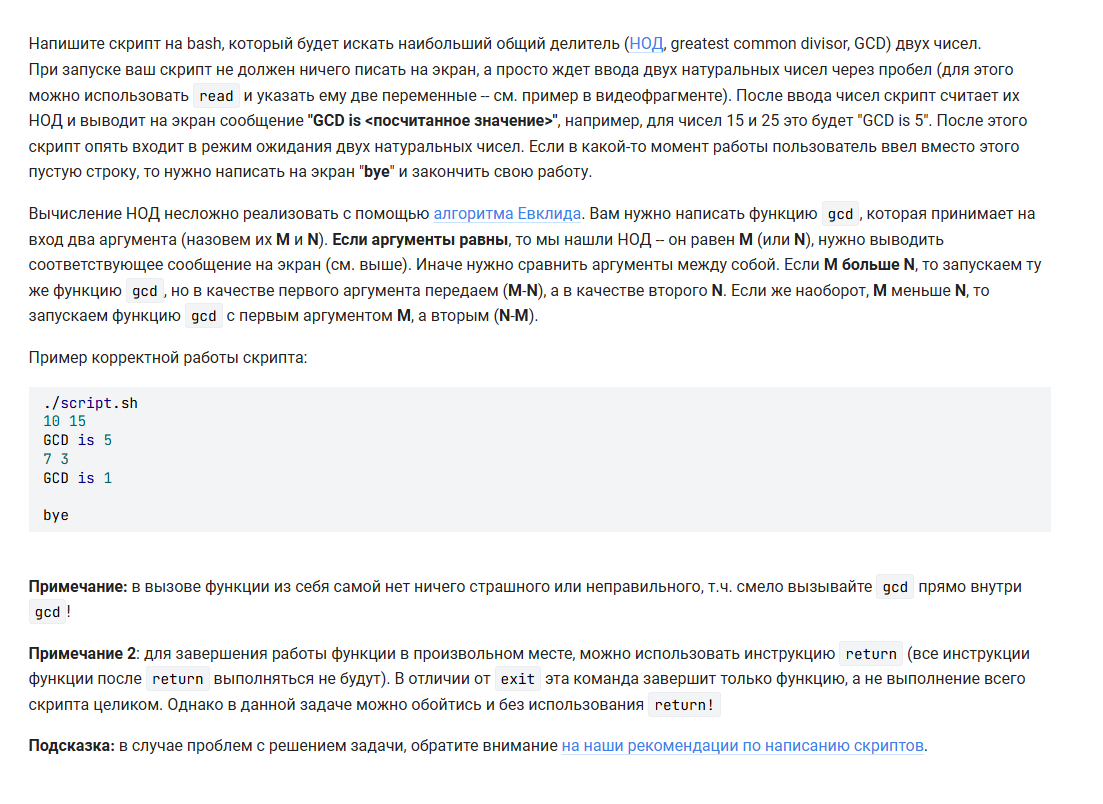


Рис. 10: Задание 10

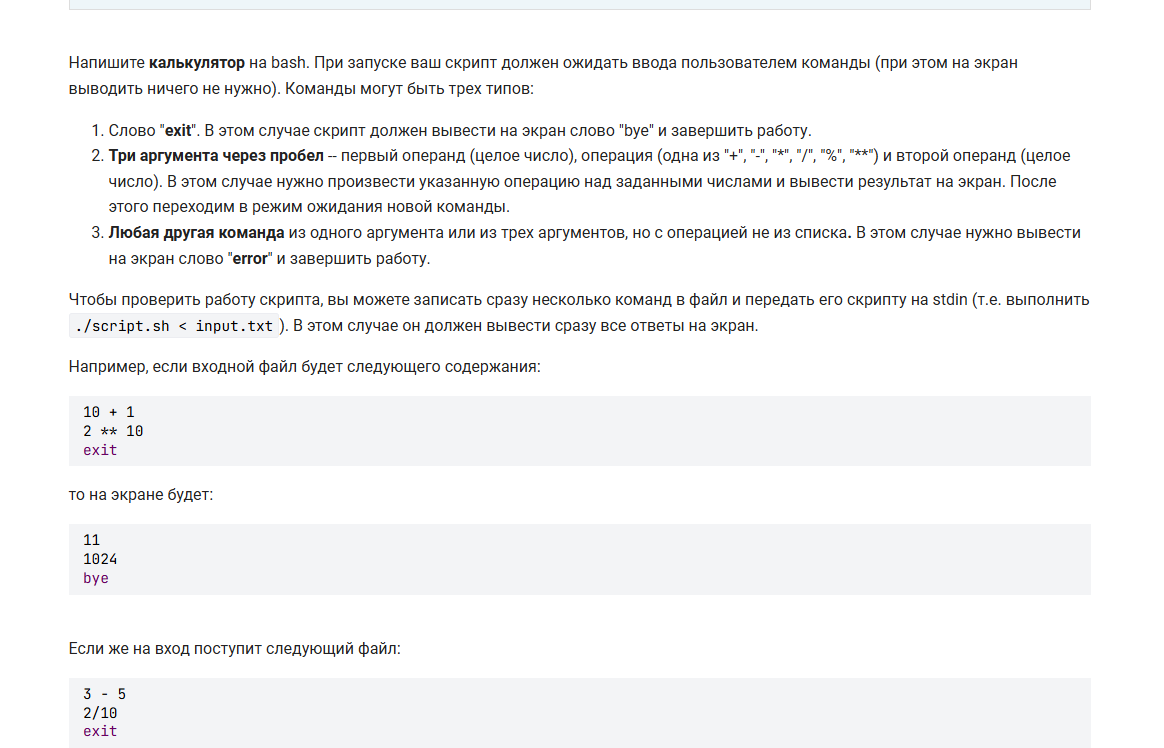
1. Опять же довольно простая программа. Надо только написать бесконечный цикл, прописать условия выхода(рис. **¿fig:011?**).

1. Программа для поиска НОД. Уже более сложная. Так же используется бесконечный цикл, if else, while.(рис. **¿fig:012?**).

1. Калькулятор на bash(рис. **¿fig:013?**).

1. 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. 11).

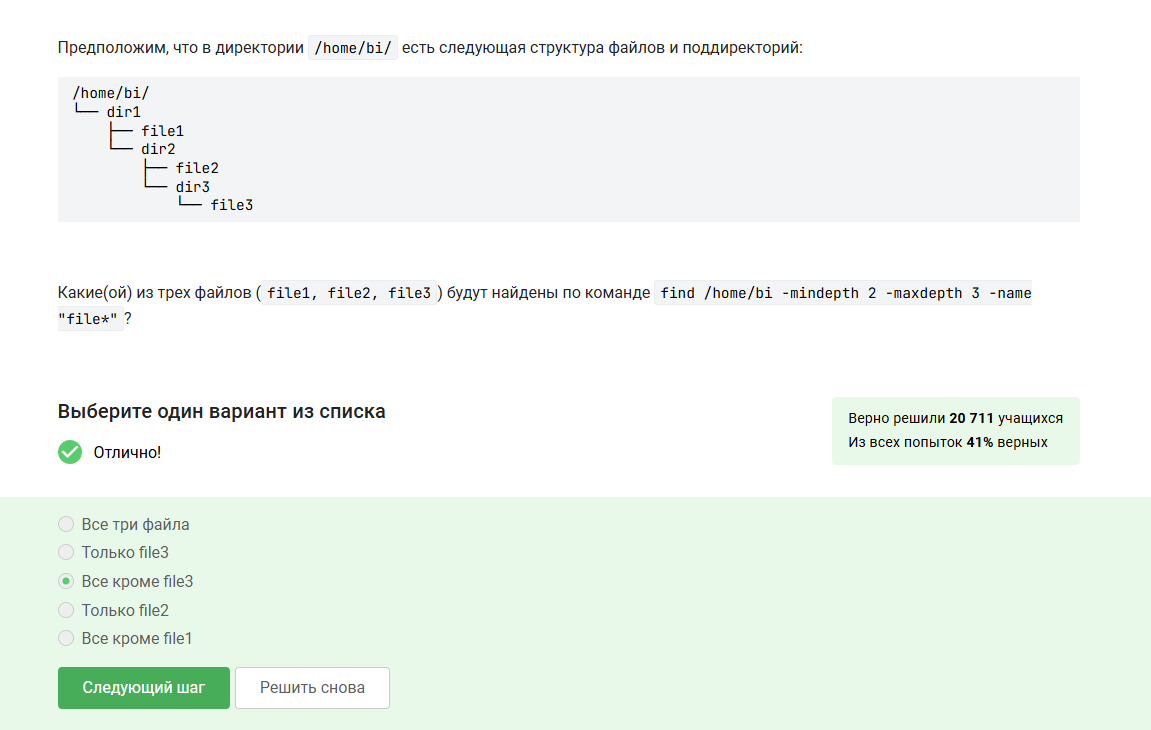


Рис. 11: Задание 14

1. Без -n, sed по умолчанию печатает каждую строку после обработки, независимо от команд внутри скрипта.(рис. 12).

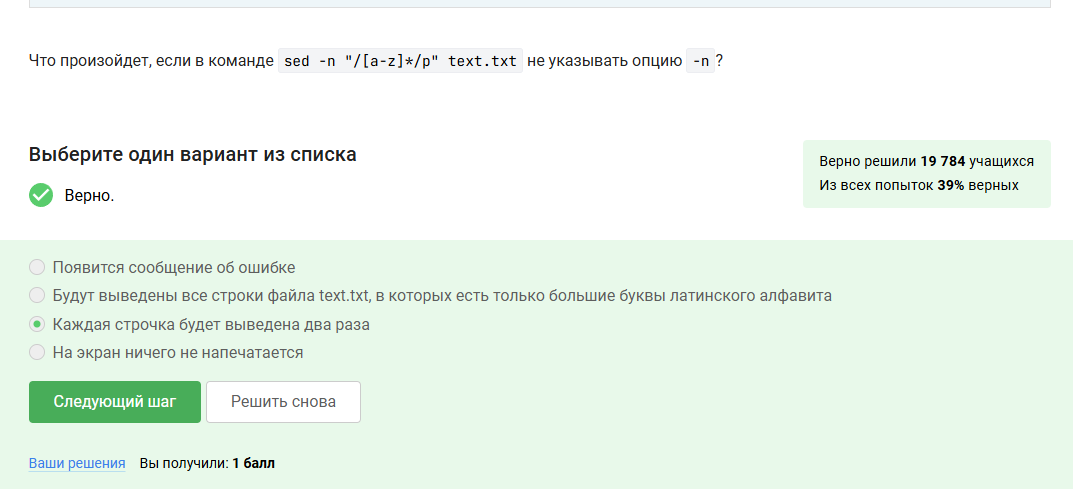


Рис. 12: Задание 15

1. Инструкция sed(рис. 13).

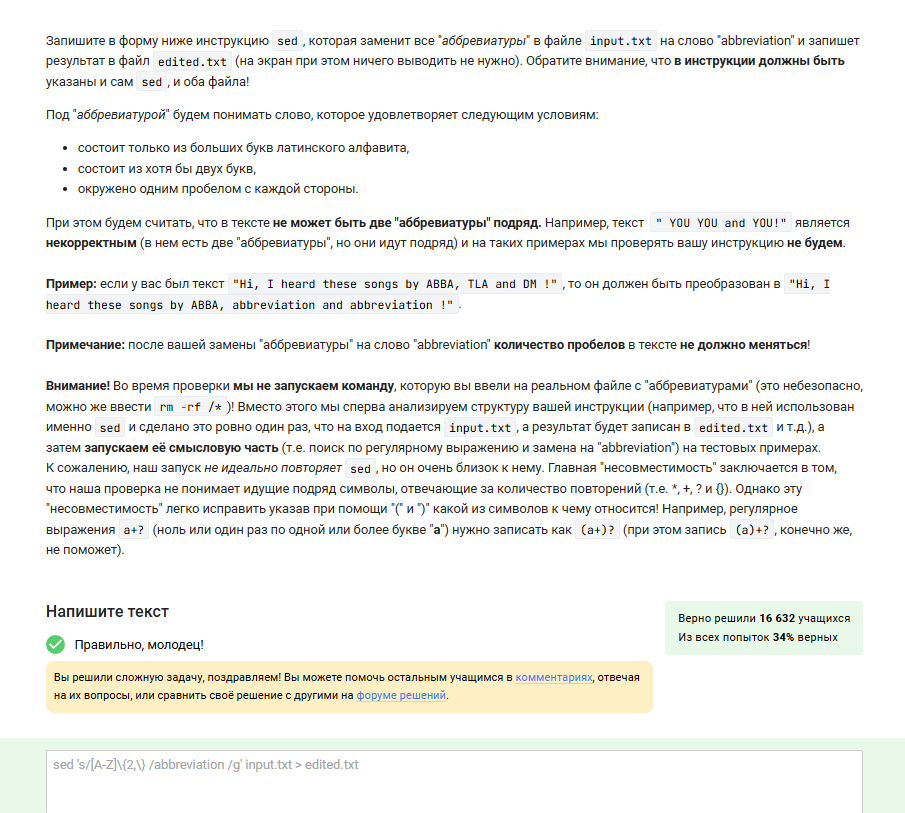


Рис. 13: Задание 16

1. gnuplot(рис. 14).

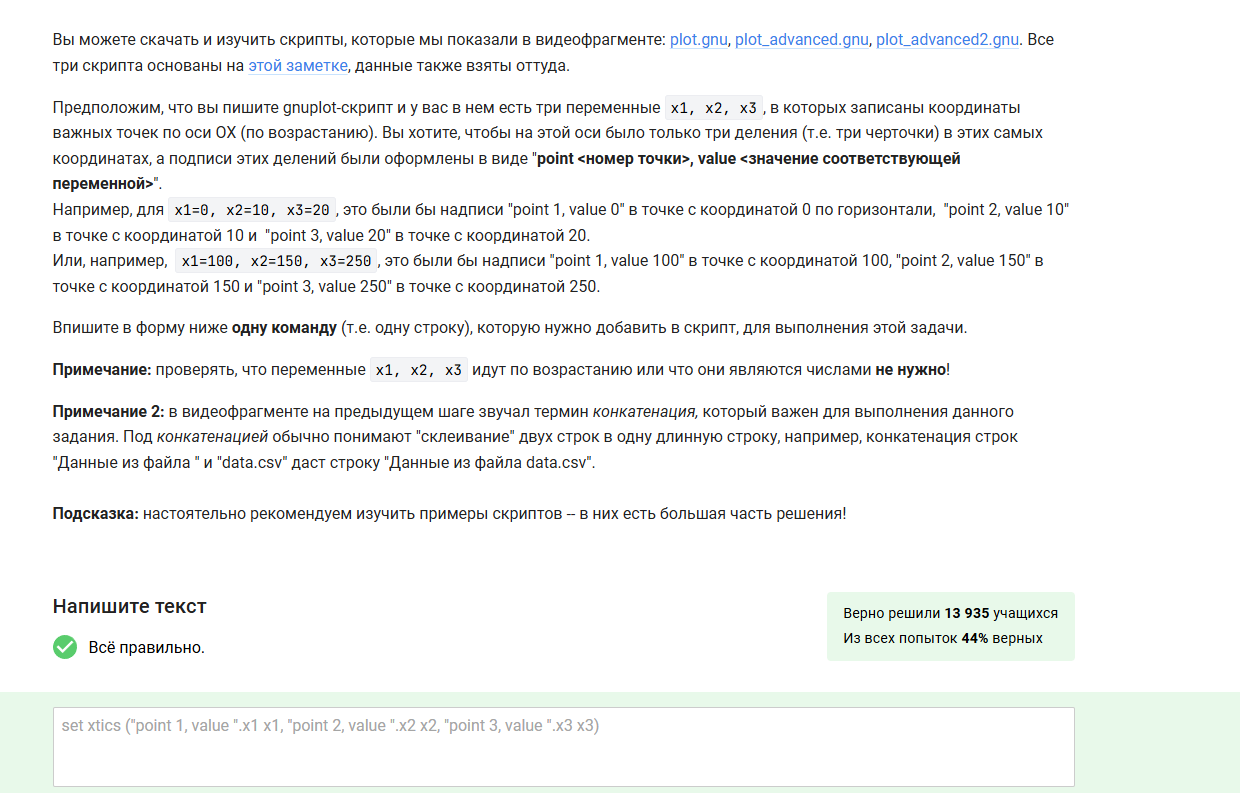


Рис. 14: Задание 17

1. Меняем работу программы под условие(рис. 15).

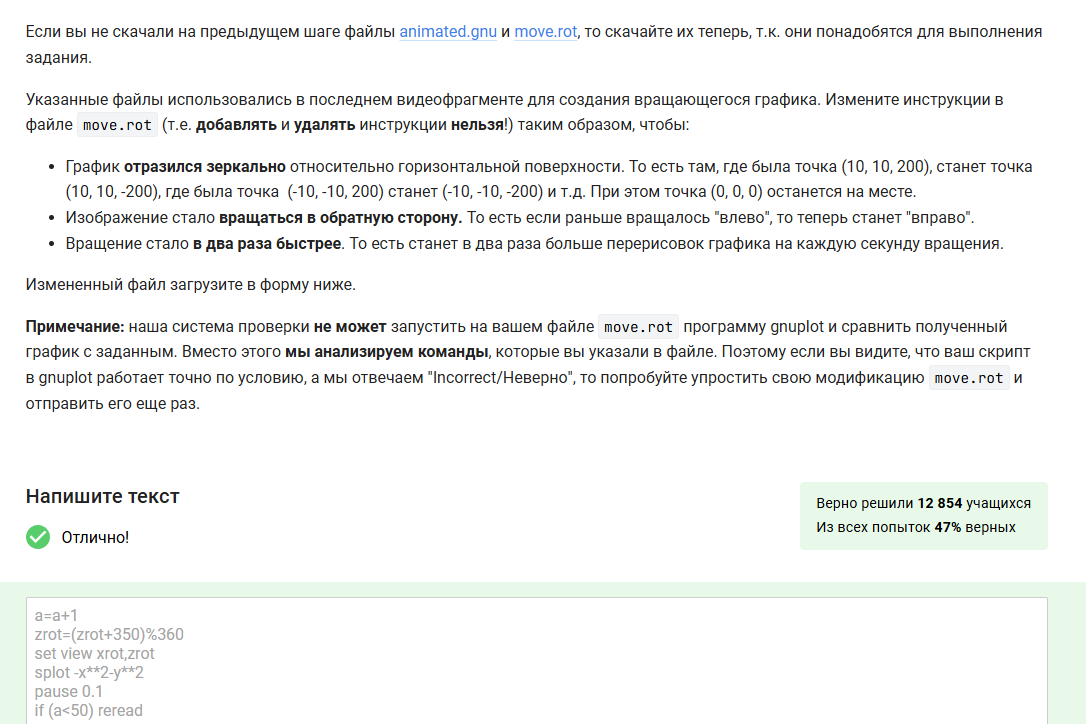


Рис. 15: Задание 18

# 3 Выводы

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