Федеральное агентство связи

Северо-Кавказский филиал ордена Трудового Красного Знамени

федерального государственного бюджетного образовательного учреждения

высшего образования

«Московский технический университет связи и информатики»

Кафедра «Информатики и вычислительной техники»

Отчет

по лабораторной работе №9

«Введение в shell-программирование»

по дисциплине «Операционные системы»

выполнил студент группы ДП-31

Гаврилов Владислав Вадимович

Ростов-на-Дону

2019

**Вариант 3**

Задача.

Вводит строку N слов, анализирует длину каждого слова, упорядочивает слова по их алфавиту и выводит список на экран. Значение N задается параметром.

Текст разработанного скрипта с комментариями:

#!/bin/bash

#Проверяем количество переданных аргументов

#Если количество переданных аргументов не равно 1, то выводим ошибку и прекращаем работу скрипта

if [ $# -ne 1 ]; then

echo "Ошибка! Требуется указать ровно один аргумент."

exit

fi

#В переменную N кладем требуемое количество слов

N=$1

#Вводим N штук слов

echo "Ожидается ввод $N слов:"

read WORDS

#Проверяем количество введенных слов

#Если количество введенных слов равно 0, то выводим ошибку и прекращаем работу скрипта

if [ ! "$WORDS" ]; then

echo "Ошибка! Введенная строка пуста."

exit

fi

#Обрамляем каждое слово комментариями и вычисляем длину

#Полученный результат помещаем в файл temporary1

for i in $WORDS; do

echo "Слово: \"$i\"; его длина = " `expr length $i`

done > temporary1

#Проверяем, что количество веденных слов равно количеству запрашиваемых слов

#Если количество введенных слов не равно количеству запрашиваемых слов, то выводим ошибку и прекращаем работу скрипта

COUNT=`cat temporary1 | wc -l`

if [ $COUNT -ne $N ]; then

echo "Ошибка! Количество введенных слов ($COUNT) не равно количеству запрошенных слов ($N)."

exit

fi

#Сортируем файл temporary1, вывод направляем в temporary2

sort temporary1 > temporary2

#Удаляем файл temporary1

rm temporary1

#Считываем построчно файл temporary2

while read LINE; do

echo $LINE

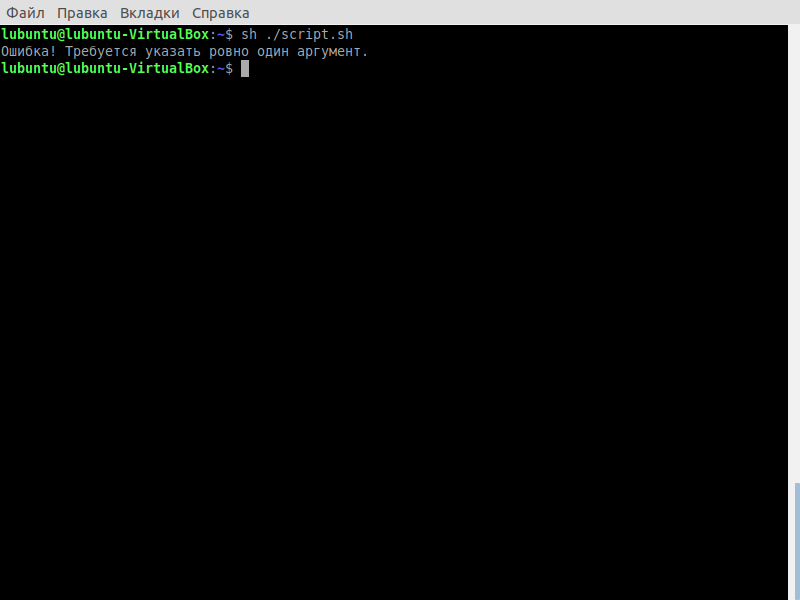
done < temporary2

#Удаляем файл temporary2

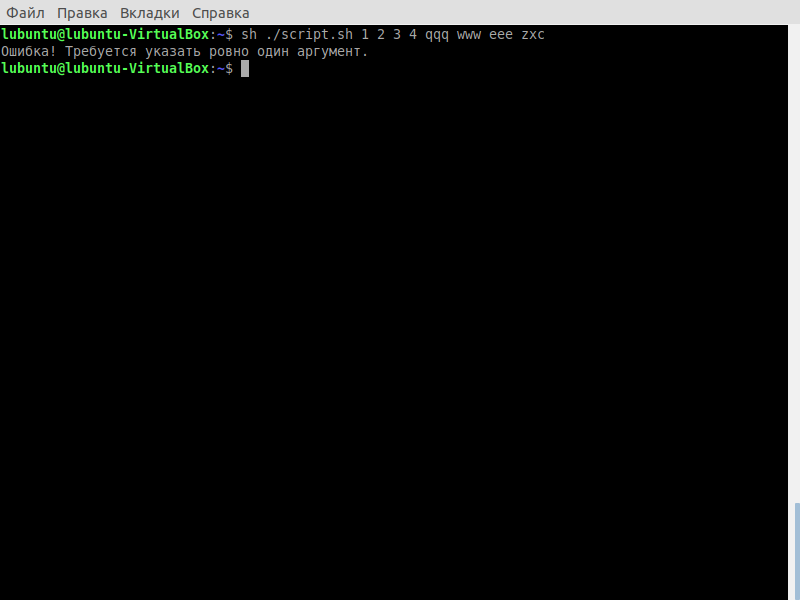
rm temporary2

Демонстрация работы скрипта.

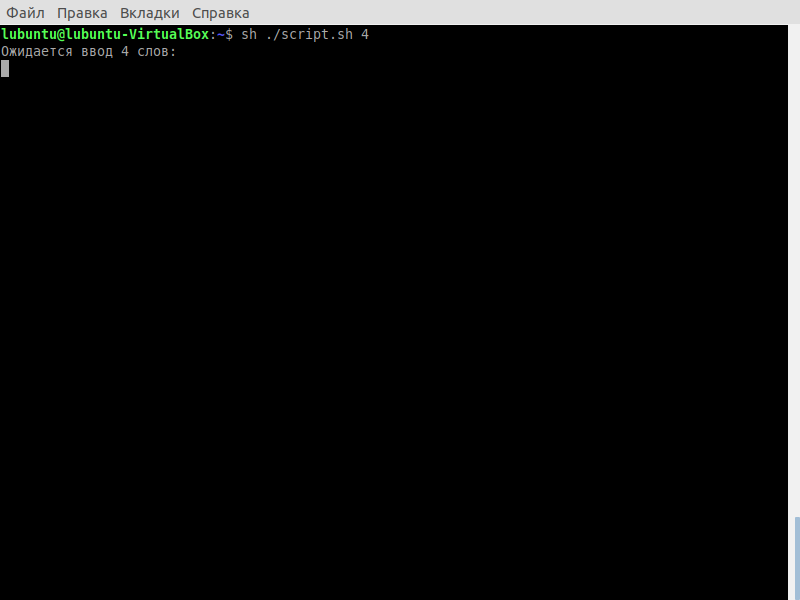
Запускаем скрипт без аргументов.

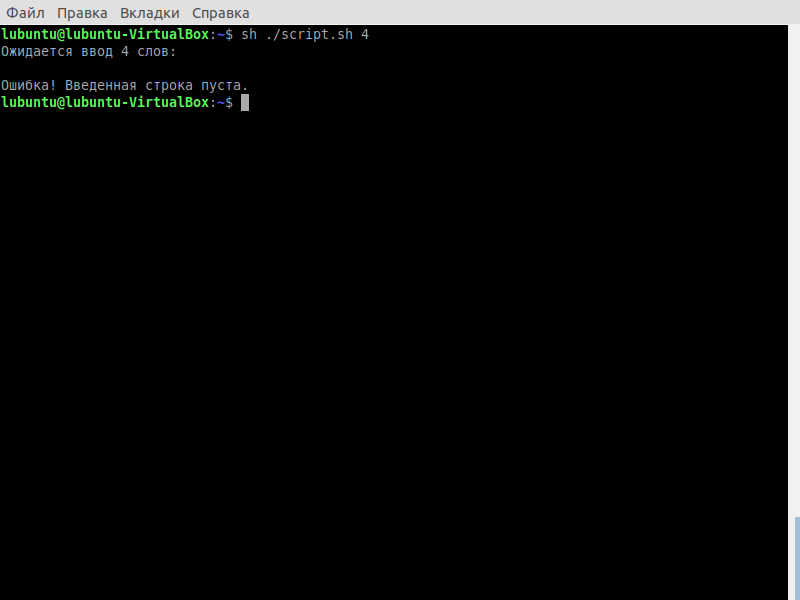


Запускаем скрипт с многими аргументами.

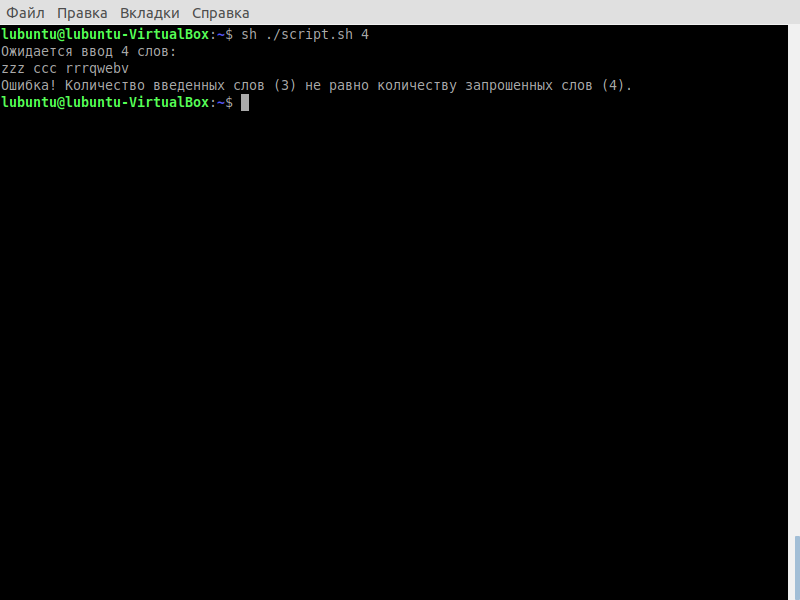


Запускаем скрипт с одним аргументом, равным 4.

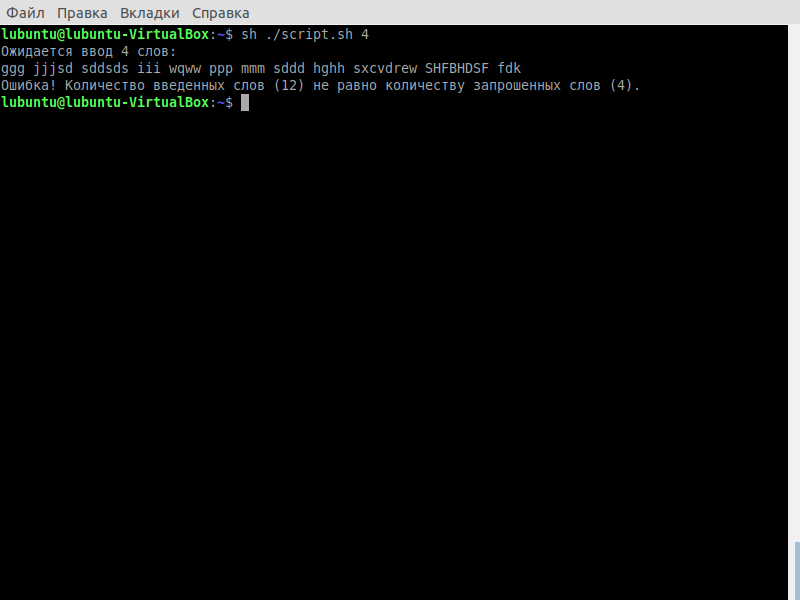


Введем пустую строку.

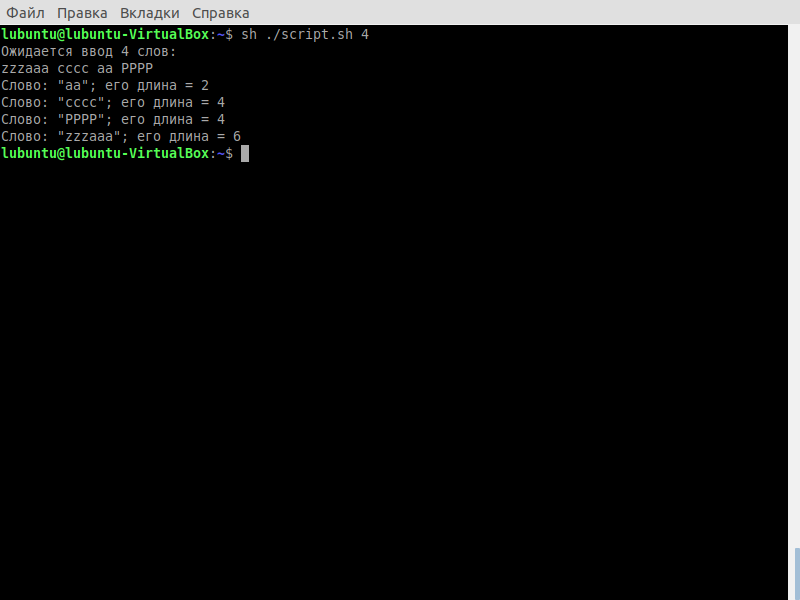
Введем 3 слова через пробел.



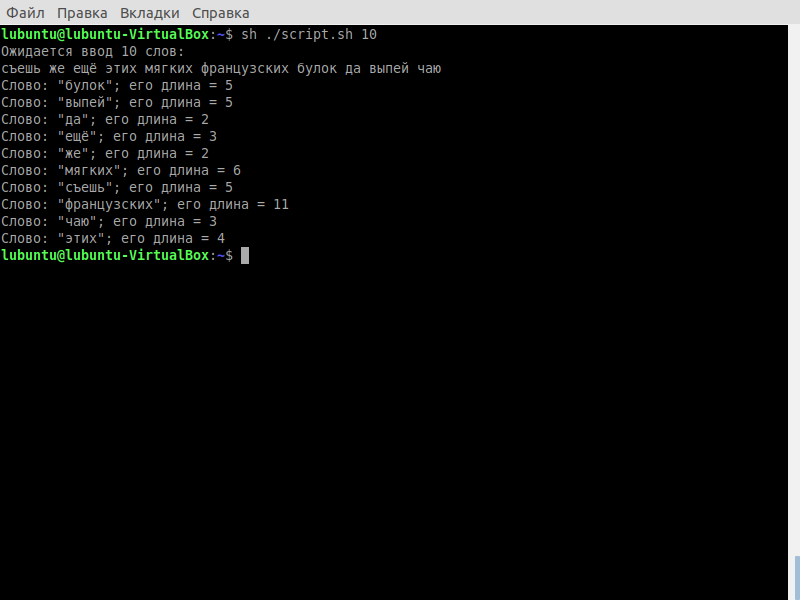
Введем 12 слов.



Наконец, введем ровно 4 слова.

Как видим, скрипт отсортировал в алфавитном порядке введенные слова, а также посчитал их длину.

Съешь же ещё этих мягких французских булок, да выпей чаю.



**Чем отличается обработка процедуры при выполнении от обработки программы на языке высокого уровня?**

Как компилятор так и интерпретатор имеют одно предназначение — конвертировать инструкции языка высокого уровня (как C или Java) в бинарную форму, понятную компьютеру. Это программное обеспечение, используемое для запуска высокоуровневых программ и кодов выполняемых различные задачи. Для разных высокоуровневых языков разработаны специфичные компиляторы/интерпретаторы. Не смотря на то что как компилятор так и интерпретатор преследуют одну и ту же цель, они отличаются способом выполнения своей задачи, то есть конвертирования высокоуровневого языка в машинные инструкции.

Ниже перечислены главные отличия между компилятором и интерпретатором:

* Интерпретатор берет одну инструкцию, транслирует и выполняет ее, а затем берет следующую инструкцию. Компилятор же транслирует всю программу сразу, а потом выполняет ее.
* Компилятор генерирует отчет об ошибках после трансляции всего, в то время как интерпретатор прекратит трансляцию после первой найденной ошибки.
* Компилятор по сравнению с интерпретатором требует больше времени для анализа и обработки языка высокого уровня.
* Помимо времени на обработку и анализ, общее время выполнения кода компилятора быстрее в сравнении с интерпретатором.

**Что такое параметры? Для каких целей они используются? Какое число параметров может быть передано процедуре?**

В bash используются предопределенные переменные для доступа к переданным в скрипт параметрам. Эти переменные имеют вид: $1, $2, $3, …, $n, где $n — переменная, хранящая n-ый переданный параметр. Также мы можем считать все аргументы в массив с помощью $@. Количество всех аргументов, переданных в скрипт, хранится в переменной $#. Рассмотрим пример:

#!/bin/bash

#вывод 1-го, 2-го и 3-го аргументов

echo $1 $2 $3

#запись в массив arg\_arr всех переданных аргументов

arg\_arr=("$@")

#вывод 1-го, 2-го и 3-го аргументов через массив arg\_arr

echo ${arg\_arr[0]} ${arg\_arr[1]} ${arg\_arr[2]}

#вывод вывод аргументов массива с помощью $@

echo $@

#вывод количества аргументов, переданных в скрипт

echo Number of arguments passed: $#

Запустим программу с тремя аргументами:

./hello-world 234 -6 hello

Вывод программы:

234 -6 hello

234 -6 hello

234 -6 hello

Number of arguments passed: 3