Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ»

КАФЕДРА «ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ»

ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ

Студент **Шахнович Дмитрий Сергеевич**

Группа **ИУ7-12Б**

Тип практики **Проектно-технологическая практика**

Название предприятия **НУК ИУ МГТУ им. Н. Э. Баумана**

Студент _____ **Шахнович Д С**

Руководитель практики _____ **Ломовской И. В.**

Руководитель практики _____ **Кострицкий А. С.**

Оценка _____

2022 г.

Оглавление

1.Введение

Целью работы является разработка скрипта командной оболочки для сравнения содержимого двух текстовых файлов по определённым правилам.

Варианты сравнения:

1. Сравнение целых чисел, содержащихся в файлах. Целые числа отделены от других элементов пробельным символами. Числа сравниваются как текст, то есть числа 100 и 0100 – разные.
2. Сравнивается текст в файлах после первого вхождения подстроки «string:». Подразумевается строгое сравнение с учётом разницы в пробельных символах и символах окончания строки.
3. Сравнение чисел с плавающей точкой, записанных не в экспоненциальной форме, содержащихся в файлах. Такие числа отделены от других элементов пробельным символами. Числа сравниваются как текст, то есть числа 1.00 и 1.0 – разные
4. Сравнение чисел с плавающей точкой, записанных в том числе в экспоненциальной форме, содержащихся в файлах. Такие числа отделены от других элементов пробельным символами. Числа сравниваются как текст, то есть числа 1.00 и 1.0 – разные, как и 1.0 и 1.0e0

2.Сценарии командной оболочки

У всех вариантов одинаковая структура запуска сценария, которая выглядит так:

```
comparator.sh file1 file2 [-v],
```

где file1 и file2 – файлы для сравнения, а -v – необязательный ключ, при котором в поток стандартного вывода будут выводиться комментарии во время исполнения сценария.

Далее приводится описание отдельных частей программ.

Ввод и проверки ввода

Так как программы имеют одинаковый формат ввода, то и сам ввод, и его проверки у всех сценариев также совпадают.

```
n=0
verbose=""
while [[ $n -lt 3 ]] && [[ -n $1 ]]; do
    if [[ $1 == "-v" ]]; then
     verbose=1
    else
     if [[ -z $file1 ]]; then
         file1=$1
     else
         file2=$1
     fi
    fi
    shift
    n=$(($n + 1))
done
```

Листинг 2.1 Ввод названия файлов и ключей

Так выглядит стандартный ввод написанных сценариев. В цикле происходит чтение до 3-х позиционных параметров или пока не встретится пустой ввод. При вводе ключа -v будет поставлен флаг, благодаря которому в будущем будет выводиться дополнительная информация при работе сценария.

Далее идут проверки корректности ввода и введенных файлов.

```
if [[ -z $file1 ]] || [[ -z $file2 ]]; then
    if [[ -n $verbose ]]; then
        echo "Incorrect input"
    fi
    exit 2
```

Листинг 2.2 Проверка на то, что ввели названия файлов

```
if [[ ! -f $file1 ]] || [[ ! -f $file2 ]]; then
    if [[ -n $verbose ]]; then
        echo "One of files is not a file or not exists"
    fi
    exit 3
fi
```

Листинг 2.3 Проверка на то, что введенные файлы существуют и являются файлами

```
if [[ ! -r $file1 ]] || [[ ! -r $file2 ]]; then
    if [[ -n $verbose ]]; then
        echo "One of files are not available for reading"
    fi
    exit 4
fi
```

Листинг 2.4 Проверка на наличие у файлов разрешений на чтение

Программа возвращает 2, если введен некорректный ввод, 3 если файла не существует или он не является файлом и 4, если у запускающего сценарий нет

прав на чтение одного из файлов. Если введен ключ -v, то при каждой из этих ошибок будет выведено соответствующее сообщение.

Варианты 1, 3 и 4

Варианты 1, 3 и 4 с точки зрения реализации отличаются друг от друга лишь переменными, содержащими регулярные выражения для проверки строки, на то, что она является числом в подходящем формате.

```
re_s="^[-]?[0-9]+$"
```

Листинг 2.5 Регулярное выражения для варианта 1

```
re_s="^[+-]?[0-9]+[.]?[0-9]*$"
```

Листинг 2.6 Регулярное выражения для варианта 3

```
re_s="^[+-]?[0-9]+(\.[0-9]*([eE][+-]?[0-9]*)?)?$"
```

Листинг 2.7 Регулярное выражения для варианта 4

Сценарий использует функцию, которая ищет числа(в принципе, любые слова отделенные пробельные символы) по регулярным выражениям:

```
function find_nums {
   while read -r line; do
       for str in $line; do
           if [[ "$str" =~ $3 ]]; then
               echo $str >> $2
           fi
       done
   done < $1
}
```

Листинг 2.8 Функция для нахождения чисел в файле

Первым аргументом подается файл для поиска чисел, вторым файл, куда записываются найденные числа, а третьим регулярное выражение.

После ввода программа создает временные файлы, куда будут записаны все числа из файла и обрабатывает поданные на вход файлы функцией.

```
nums_file1=$(mktemp)
nums_file2=$(mktemp)

find_nums $file1 $nums_file1 $re_s
find_nums $file2 $nums_file2 $re_s
```

Листинг 2.9 Нахождение чисел в файлах.

Затем мы построчно сравниваем полученные временные файлы с числами до конца одного из файлов или первого несовпадения чисел. Если обработка была завершена досрочно, или один файл кончился раньше другого, то будет возвращен код 1, если же файлы одинаковые, то 0

```
exec 3<$nums_file1
exec 4<$nums_file2

end_f1=1
end_f2=1

while [[ $end_f1 -gt 0 ]] && [[ $end_f2 -gt 0 ]]; do
    if ! read num1 <&3; then
        end_f1=0
    fi
    if ! read num2 <&4; then
        end_f2=0
    fi
    if [[ -n $verbose ]]; then
```

```
        echo "Number from file 1: "$num1
        echo "Number from file 2: "$num2
    fi
    if [[ $num1 != $num2 ]]; then
        break
    fi
done

if [[ $end_f2 -eq 0 ]] && [[ $end_f1 -eq 0 ]]; then
    if [[ -n $verbose ]]; then
        echo "Files have the same content"
    fi
  exit 0
else
    if [[ -n $verbose ]]; then
        echo "Files does not have the same content"
    fi
  exit 1
fi
```

Листинг 2.10 Сравнение чисел в файлах

Вариант 2

Сценарий использует функцию, которая во введенном файле находит подстроку после первого вхождения “string:”:

```
function find_after_string {
    sep="string:"
    found=0
    while read -r line; do
        if [[ "$line" == *"$sep"* ]] && [[ $found -eq 0 ]]; then
            found=1
```

```
            before1=${line%%"$sep"*}
            ind1=${#before1}
            after1=${line:$(($ind1))}
            if [[ -z $after1 ]]; then
                line=$before1
            else
                line=$after1
            fi
            echo $line > $2
        else
            echo $line >> $2
        fi
    done < $1
}
```

Листинг 2.11 Функция для поиска текста после «string:»

Первым аргументом ей подается файл для обработки, а вторым файл, в который записывается текст после «string:». В случае если «string:» в тексте не обнаружен, в файл 2 весь текст входного файла.

После ввода программа создает временные файлы, куда будут записаны результаты работы функции и аналогично другим вариантам эти файлы будут построчно сравниваться.

```
after_text_file1=$(mktemp)
after_text_file2=$(mktemp)

find_after_string $file1 $after_text_file1
find_after_string $file2 $after_text_file2
exec 3<$after_text_file1
exec 4<$after_text_file2
```

```
end_f1=1
end_f2=1
while [[ $end_f1 -gt 0 ]] && [[ $end_f2 -gt 0 ]]; do
    if ! read line1 <&3; then
        end_f1=0
    fi
    if ! read line2 <&4; then
        end_f2=0
    fi
    if [[ -n $verbose ]]; then
        echo "Line from file 1: "$line1
        echo "Line from file 2: "$line2
    fi
    if [[ $line1 != $line2 ]]; then
        break
    fi
done
if [[ $end_f2 -eq 0 ]] && [[ $end_f1 -eq 0 ]]; then
    if [[ -n $verbose ]]; then
        echo "Files have the same content"
    fi
  exit 0
else
    if [[ -n $verbose ]]; then
        echo "Files does not have the same content"
    fi
  exit 1
fi
```

Листинг 2.12 Сравнение файлов.

3.Тестирование

Все тесты располагаются в папке tests/ в каждом из вариантов. Сценарий tests.sh, который также лежит в каждом из вариантов, запускает тестирование по этим тестам.

Вариант 1

№	Описание	Результат
1	Пустые файлы	**Пройден**
2	Один из файлов не существует	**Пройден**
3	Подан только один файл на вход	**Пройден**
4	Тест на числа, разделенные переносом строки	**Пройден**
5	Тест с числами разделенным символами	**Пройден**
6	Тест с одним и тем же файлом	**Пройден**
7	Тест с большим файлом	**Пройден**
8	Тест с большим файлом	**Пройден**

Вариант 2

№	Описание	Результат
1	Пустые файлы	**Пройден**
2	Один из файлов не существует	**Пройден**
3	Подан только один файл на вход	**Пройден**
4	Стандартный тест	**Пройден**
5	Тест с несколькими «string:» в файле	**Пройден**
6	Тест с одним и тем же файлом	**Пройден**
7	Тест с различием в перенос строки	**Пройден**
8	Тест с большим файлом	**Пройден**

Вариант 3

№	Описание	Результат
1	Пустые файлы	**Пройден**
2	Один из файлов не существует	**Пройден**
3	Подан только один файл на вход	**Пройден**
4	Стандартный тест	**Пройден**
5	Тест с разницей в перенос строки	**Пройден**
6	Тест с одним и тем же файлом	**Пройден**
7	Стандартный тест	**Пройден**
8	Тест с целым числом	**Пройден**

Вариант 4

№	Описание	Результат
1	Пустые файлы	**Пройден**
2	Один из файлов не существует	**Пройден**
3	Подан только один файл на вход	**Пройден**
4	Стандартный тест	**Пройден**
5	Тест с одинаковыми числами в разных формах	**Пройден**
6	Тест с одним и тем же файлом	**Пройден**
7	Стандартный тест	**Пройден**
8	Стандартный тест	**Пройден**

4.Заключение

В ходе работы были написаны сценарии командной оболочки с вариантами 1-4. Также были реализованы простые тестирующие системы и написаны тесты для проверки реализованных сценариев. Написанные сценарии можно найти в папках task1, task2, task3, task4 соответственно.

Задание было выполнено.

5.Листинги исходного кода

```
#!/bin/bash



function find_nums {
   while read -r line; do
       for str in $line; do
           if [[ "$str" =~ $3 ]]; then
               echo $str >> $2
           fi
       done
   done < $1
}



n=0
re_s="^[-]?[0-9]+$"
verbose=""
while [[ $n -lt 3 ]] && [[ -n $1 ]]; do
    if [[ $1 == "-v" ]]; then
     verbose=1
    else
     if [[ -z $file1 ]]; then
         file1=$1
     else
         file2=$1
     fi
    fi
```

```
    shift
    n=$(($n + 1))
done

if [[ -z $file1 ]] || [[ -z $file2 ]]; then
    if [[ -n $verbose ]]; then
        echo "Incorrect input"
    fi
    exit 2
fi

if [[ ! -f $file1 ]] || [[ ! -f $file2 ]]; then
    if [[ -n $verbose ]]; then
        echo "One of files is not a file or not exists"
    fi
    exit 3
fi

if [[ ! -r $file1 ]] || [[ ! -r $file2 ]]; then
    if [[ -n $verbose ]]; then
        echo "One of files are not available for reading"
    fi
    exit 4
fi

if [[ -n $verbose ]]; then
    echo "Correct input"
fi

nums_file1=$(mktemp)
```

```
nums_file2=$(mktemp)

find_nums $file1 $nums_file1 $re_s
find_nums $file2 $nums_file2 $re_s

exec 3<$nums_file1
exec 4<$nums_file2

end_f1=1
end_f2=1

while [[ $end_f1 -gt 0 ]] && [[ $end_f2 -gt 0 ]]; do
    if ! read num1 <&3; then
        end_f1=0
    fi
    if ! read num2 <&4; then
        end_f2=0
    fi
    if [[ -n $verbose ]]; then
        echo "Number from file 1: "$num1
        echo "Number from file 2: "$num2
    fi
    if [[ $num1 != $num2 ]]; then
        break
    fi
done

if [[ $end_f2 -eq 0 ]] && [[ $end_f1 -eq 0 ]]; then
    if [[ -n $verbose ]]; then
        echo "Files have the same content"
```

```
    fi
  exit 0
else
    if [[ -n $verbose ]]; then
        echo "Files does not have the same content"
    fi
  exit 1
fi
```

Листинг 5.1 comparator1.sh

```
#!/bin/bash

n=0
function find_after_string {
    sep="string:"
    found=0
    while read -r line; do
        if [[ "$line" == *"$sep"* ]] && [[ $found -eq 0 ]]; then
            found=1
            before1=${line%%"$sep"*}
            ind1=${#before1}
            after1=${line:$(($ind1))}
            if [[ -z $after1 ]]; then
                line=$before1
            else
                line=$after1
            fi
            echo $line > $2
        else
            echo $line >> $2
        fi
```

```
   done < $1
}

while [[ $n -lt 3 ]] && [[ -n $1 ]]; do
    if [[ $1 == "-v" ]]; then
     verbose=1
    else
     if [[ -z $file1 ]]; then
         file1=$1
     else
         file2=$1
     fi
    fi
    shift
    n=$(($n + 1))
done

if [[ -z $file1 ]] || [[ -z $file2 ]]; then
    if [[ -n $verbose ]]; then
        echo "Incorrect input"
    fi
    exit 2
fi

if [[ ! -f $file1 ]] || [[ ! -f $file2 ]]; then
    if [[ -n $verbose ]]; then
        echo "One of files is not a file or not exists"
    fi
    exit 3
fi
```

```
if [[ ! -r $file1 ]] || [[ ! -r $file2 ]]; then
    if [[ -n $verbose ]]; then
        echo "One of files are not available for reading"
    fi
    exit 4
fi


if [[ -n $verbose ]]; then
    echo "Correct input"
fi

after_text_file1=$(mktemp)
after_text_file2=$(mktemp)

find_after_string $file1 $after_text_file1
find_after_string $file2 $after_text_file2

exec 3<$after_text_file1
exec 4<$after_text_file2

end_f1=1
end_f2=1

while [[ $end_f1 -gt 0 ]] && [[ $end_f2 -gt 0 ]]; do
    if ! read line1 <&3; then
        end_f1=0
    fi
    if ! read line2 <&4; then
```

```
        end_f2=0
    fi
    if [[ -n $verbose ]]; then
        echo "Line from file 1: "$line1
        echo "Line from file 2: "$line2
    fi
    if [[ $line1 != $line2 ]]; then
        break
    fi
done

if [[ $end_f2 -eq 0 ]] && [[ $end_f1 -eq 0 ]]; then
    if [[ -n $verbose ]]; then
        echo "Files have the same content"
    fi
  exit 0
else
    if [[ -n $verbose ]]; then
        echo "Files does not have the same content"
    fi
  exit 1
fi
```

Листинг 5.2 comparator2.sh

```
#!/bin/bash
function find_nums {
   while read -r line; do
       for str in $line; do
           if [[ "$str" =~ $3 ]]; then
               echo $str >> $2
           fi
```

```
        done
    done < $1
}


n=0
re_s="^[+-]?[0-9]+[.]?[0-9]*$"
while [[ $n -lt 3 ]] && [[ -n $1 ]]; do
    if [[ $1 == "-v" ]]; then
     verbose=1
    else
     if [[ -z $file1 ]]; then
         file1=$1
     else
         file2=$1
     fi
    fi
    shift
    n=$(($n + 1))
done

if [[ -z $file1 ]] || [[ -z $file2 ]]; then
    if [[ -n $verbose ]]; then
        echo "Incorrect input"
    fi
    exit 2
fi

if [[ ! -f $file1 ]] || [[ ! -f $file2 ]]; then
    if [[ -n $verbose ]]; then
```

```
        echo "One of files is not a file or not exists"
    fi
    exit 3
fi

if [[ ! -r $file1 ]] || [[ ! -r $file2 ]]; then
    if [[ -n $verbose ]]; then
        echo "One of files are not available for reading"
    fi
    exit 4
fi

if [[ -n $verbose ]]; then
    echo "Correct input"
fi

nums_file1=$(mktemp)
nums_file2=$(mktemp)

find_nums $file1 $nums_file1 $re_s
find_nums $file2 $nums_file2 $re_s

exec 3<$nums_file1
exec 4<$nums_file2

end_f1=1
end_f2=1

while [[ $end_f1 -gt 0 ]] && [[ $end_f2 -gt 0 ]]; do
    if ! read num1 <&3; then
```

```
        end_f1=0
    fi
    if ! read num2 <&4; then
        end_f2=0
    fi
    if [[ -n $verbose ]]; then
        echo "Number from file 1: "$num1
        echo "Number from file 2: "$num2
    fi
    if [[ $num1 != $num2 ]]; then
        break
    fi
done

if [[ $end_f2 -eq 0 ]] && [[ $end_f1 -eq 0 ]]; then
    if [[ -n $verbose ]]; then
        echo "Files have the same content"
    fi
  exit 0
else
    if [[ -n $verbose ]]; then
        echo "Files does not have the same content"
    fi
  exit 1
fi
```

Листинг 5.3 comparator3.sh

```
#!/bin/bash

function find_nums {
   while read -r line; do
```

```
        for str in $line; do
            if [[ "$str" =~ $3 ]]; then
                echo $str >> $2
            fi
        done
    done < $1
}


n=0
re_s="^[+-]?[0-9]+(\.[0-9]*([eE][+-]?[0-9]*)?)?$"
while [[ $n -lt 3 ]] && [[ -n $1 ]]; do
    if [[ $1 == "-v" ]]; then
     verbose=1
    else
     if [[ -z $file1 ]]; then
         file1=$1
     else
         file2=$1
     fi
    fi
    shift
    n=$(($n + 1))
done

if [[ -z $file1 ]] || [[ -z $file2 ]]; then
    if [[ -n $verbose ]]; then
        echo "Incorrect input"
    fi
    exit 2
```

```
fi

if [[ ! -f $file1 ]] || [[ ! -f $file2 ]]; then
    if [[ -n $verbose ]]; then
        echo "One of files is not a file or not exists"
    fi
    exit 3
fi

if [[ ! -r $file1 ]] || [[ ! -r $file2 ]]; then
    if [[ -n $verbose ]]; then
        echo "One of files are not available for reading"
    fi
    exit 4
fi

if [[ -n $verbose ]]; then
    echo "Correct input"
fi

nums_file1=$(mktemp)
nums_file2=$(mktemp)

find_nums $file1 $nums_file1 $re_s
find_nums $file2 $nums_file2 $re_s

exec 3<$nums_file1
exec 4<$nums_file2

end_f1=1
```

```
end_f2=1

while [[ $end_f1 -gt 0 ]] && [[ $end_f2 -gt 0 ]]; do
    if ! read num1 <&3; then
        end_f1=0
    fi
    if ! read num2 <&4; then
        end_f2=0
    fi
    if [[ -n $verbose ]]; then
        echo "Number from file 1: "$num1
        echo "Number from file 2: "$num2
    fi
    if [[ $num1 != $num2 ]]; then
        break
    fi
done

if [[ $end_f2 -eq 0 ]] && [[ $end_f1 -eq 0 ]]; then
    if [[ -n $verbose ]]; then
        echo "Files have the same content"
    fi
  exit 0
else
    if [[ -n $verbose ]]; then
        echo "Files does not have the same content"
    fi
  exit 1
fi
```

Листинг 5.4 comparator4.sh

6.СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1) [Электронный ресурс] https://ru.stackoverflow.com/
2) [Электронный ресурс] https://habr.com