Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Национальный исследовательский университет «МЭИ»

|  |  |  |  |
| --- | --- | --- | --- |
| Институт: | *ИВТ* | Кафедра: | *ВТ* |
| Направление подготовки: | | *09.03.01 Информатика и вычислительная техника* | |

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ 4**

**Выполнили студенты**

**группы А-06-20:**

Янкин М.И.

Чернышов Д.М.

**Москва 2023**

**Оглавление**

[**1. Задание:** 3](#_Toc150278231)

[**2. Анализ задания и распределение ролей** 4](#_Toc150278232)

[**3. Решение первой задачи** 5](#_Toc150278233)

[**3.1. Пункт 1 – Вывод имени студента, не сдавшего хотя бы один тест (с указанием номера теста)** 5](#_Toc150278234)

[**3.2. Пункт 2 – Вывод списка группы, упорядоченного по количеству попыток сдачи теста** 5](#_Toc150278235)

[**4. Решение второй задачи** 6](#_Toc150278236)

[**4.1. Пункт 1 – Вывод по фамилии студента его средней оценки по всем предметам** 6](#_Toc150278237)

[**4.2. Пункты 2 и 3 – вывод по номеру группы занятий с максимальной (минимальной) общей посещаемостью** 6](#_Toc150278238)

[**5. Обработка исключений и унификация кода** 7](#_Toc150278239)

[**6. Ход разработки и тестирование** 8](#_Toc150278240)

[**7. ПРИЛОЖЕНИЕ** 11](#_Toc150278241)

# **1. Задание:**

Преподаватель (далее - заказчик) выдвигает следующие функциональные требования к разрабатываемому Решению под ОС Linux.

1. В качестве предметной области и исходных данных выступает Файловая Система Преподавателя из Лабораторной работы №3.

2. Базовое решение для успешной сдачи лабораторной работы должно предоставлять следующие возможности по заданному пользователем номеру группы:

\* вывод имени студента, не сдавшего хотя бы один тест (с указанием номера теста);

\* вывод списка группы, упорядоченного по количеству попыток сдачи теста.

3. Для получения Отличной оценки (и потенциально дополнительных баллов) также необходимо в рамках решения реализовать:

\* вывод по фамилии студента его средней оценки по всем тестам предмета;

\* вывод по номеру группы занятий с минимальной общей посещаемостью;

\* вывод по номеру группы занятий с максимальной общей посещаемостью.

Требования могут уточняться менеджером или аналитиком бригады лично с заказчиком.

# **2. Анализ задания и распределение ролей**

Задание лабораторной работы задано в виде двух задач: базовая и задача для получения отличной оценки. Каждое из заданий включает в себя несколько пунктов.

Для решения задания внутри бригады было принято решение о распределении ролей:

Янкин М.И. – разработчик, аналитик, менеджер

Чернышов Д.М. – тестировщик, разработчик

Первая задача досталась для разработки Чернышову Д.М., вторая – Янкину М.И. В качестве требуемого решения каждый должен был предоставить bash-скрипт, решающий выделенную задачу. После выполнения и тестирования задач по отдельности было принято решение о написании еще одного скрипта, запускающего по выбору пользователя тот или иной скрипт, отвечающий за конкретную задачу. Это выполнил Чернышов Д.М.

В процессе разработки необходимо было уточнение у заказчика некоторых деталей решения задач. Например, было выяснено, что во второй задаче в первом пункте подразумевается, что пользователь может ввести только часть фамилии, а не все имя студента целиком (Ivanov, а не IvanovMK), и программа должна выводить результат для всех студентов, у которых имя включает введенную фамилию. Второе уточнение – средний балл должен учитывать только успешные попытки сдачи теста, а не все подряд.

Скрипт, решающий первую задачу, называется ./Part1.sh, вторую - ./Part2.sh. Скрипт, позволяющий пользователю выбрать, какую задачу предстоит решить - ./Program.sh.

# **3. Решение первой задачи**

При запуске скрипта пользователю предлагается ввести номер группы, либо символ «0», чтобы выйти из скрипта (к выбору задачи).

Ввод номера группы организован с помощью цикла: после ввода скрипт ищет файл посещаемости группы в файловой системе, и, если такой есть – происходит выход из цикла, иначе – сообщение об ошибке и повторный запрос на ввод.

После успешного ввода группы пользователю предлагается выбрать пункт задачи, либо вернуться к выбору группы.

# **3.1. Пункт 1 – Вывод имени студента, не сдавшего хотя бы один тест (с указанием номера теста)**

При выборе этого пункта, скрипт формирует полный список группы из файла посещаемости, а затем, проходя в цикле каждый предмет и каждый тест, для каждого студента ищет записи в файле теста об успешной сдаче этим студентом (т.е. запись с положительной оценкой). Если такая запись не была найдена – имя студента добавляется в массив не сдавших, и номер теста также в специальный массив номеров тестов (при условии, что этого студента еще нет в этом массиве).

Затем для каждого предмета выводится список из элементов этого массива с указанием номера теста.

# **3.2. Пункт 2 – Вывод списка группы, упорядоченного по количеству попыток сдачи теста**

Скрип проходится в цикле по всем предметам и по всем тестам и подсчитывает количество попыток сдачи каждого студента. Для этого использована конструкция *«uniq -c»*, убирающая повторяющиеся строки и добавляющая количество повторений перед строкой. Получившийся список сортируется оператором sort и выводится пользователю.

# **4. Решение второй задачи**

При запуске соответствующего скрипта, пользователю предлагается выбрать пункт задачи, либо, при вводе символа «0», выйти из скрипта (к выбору задачи).

# **4.1. Пункт 1 – Вывод по фамилии студента его средней оценки по всем предметам**

Пользователю предлагается ввести фамилию студента – полностью или частично. После этого формируется массив найденных студентов. Если ни одного студента с такой фамилией не было найдено – выводится сообщение об ошибке и пользователю предлагается повторный ввод.

Для каждого найденного студента в цикле по каждому предмету и по каждому тесту формируется средняя оценка: складывается положительные оценки по каждому тесту и затем делится на количество предметов. Результат выводится на экран.

# **4.2. Пункты 2 и 3 – вывод по номеру группы занятий с максимальной (минимальной) общей посещаемостью**

Пользователю предлагается ввести номер группы, затем в файловой системе ищется файл посещаемости группы хотя бы по одному предмету. Если файл не найден – выводится сообщение об ошибке. Если файл найден, тогда скрипт для каждого предмета в файле посещаемости считает количество единиц, т.е. знака посещения занятия, для каждого из занятий. Эти значения заносятся в массив.

В массиве посещаемости в зависимости от пункта ищется максимум или минимум и запоминается индекс этого элемента. Получившееся значение выводится на экран.

# **5. Обработка исключений и унификация кода**

В обеих задачах пользователю требуется вводить какие-либо значения, что, очевидно, может привести к ошибкам. Основная масса ошибок в виде опечаток или каких-то некорректных значений номера группы/фамилии студента, а также выбора пункта задачи отсеивается с помощью конструкции *«case»,* которая не воспринимает никаких значений, кроме установленных, а также команд поиска группы/студента, признаком некорректного поиска в которых является возврат пустой строки.

Унификация кода заключается в том, что все значения, которые могут использоваться в циклах, в коде описываются не вручную, а с помощью поиска соответствующих файлов. Примерами такой унификации являются названия предметов и пути к соответствующим директориям, количество занятий у группы по предметам, а также количество тестов по предмету.

Пути к предметам заносятся в массив, их можно получить с помощью конструкции *«find -type d»*, которая ищет директории из указанной начальной. С помощью пути к папке можно получить имя этой папки, т.е. предмета.

Количество тестов можно получить сочетанием конструкций *«find»* для поиска файлов, содержащих название файлов с тестами *«TEST-#»*, и *«grep -c»* для нахождения количества этих файлов.

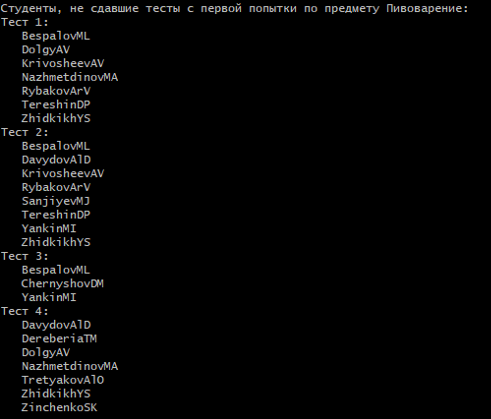
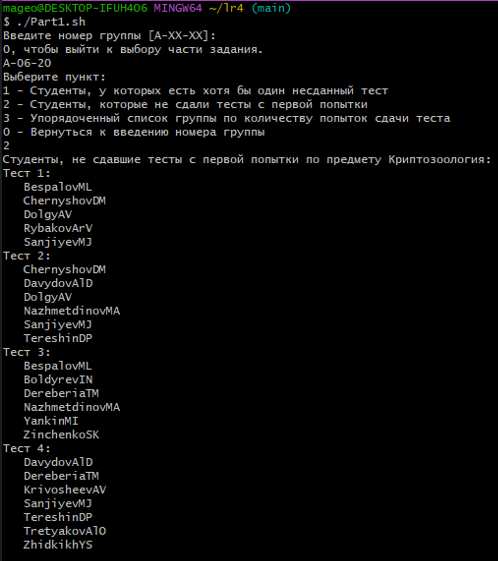
Количество занятий по предмету узнаем следующим образом: получаем первую строку из файла посещаемости группы, оставляем в ней с помощью конструкции *«sed»* подстроку, содержащую нули и единицы, и затем возвращаем длину строки.

# **6. Ход разработки и тестирование**

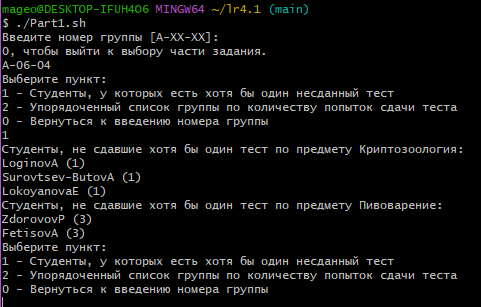
Для тестирования задачи изменялись некоторые файлы в директории ./labfiles – с посещаемостью и результатами тестов, а также добавлялась папка с третьим предметом для проверки работоспособности унификации кода. Так, например, в записях с результатами тестов были убраны некоторые записи об успешных сдачах тестов у студентов группы A-06-04. Также изменялся файл посещаемости групп A-06-20 и A-09-20 для проверки условных граничных условий – первое и последнее занятие, которые в измененном виде представляли собой самое посещаемое или наименее посещаемое занятие.

Достаточное кол-во проблем было связано с отладкой вывода (как для первой, так и для второй части задания). Все они были решены путем прогона и поиска наиболее «читаемого» формата.

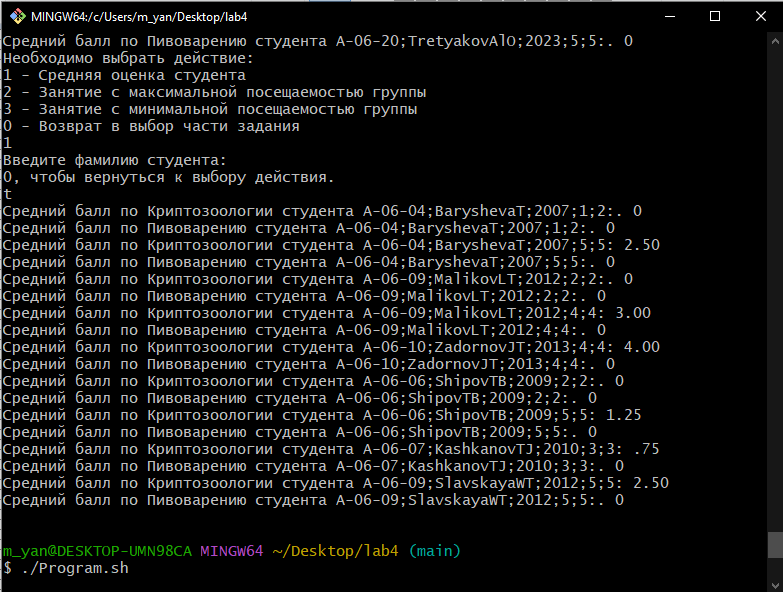
При первой попытке сдачи найдена основная проблема решения первой задачи: неправильно был интерпретирован первый пункт: Первоначальное решение искало людей, которые просто не сдали тест с первого раза.



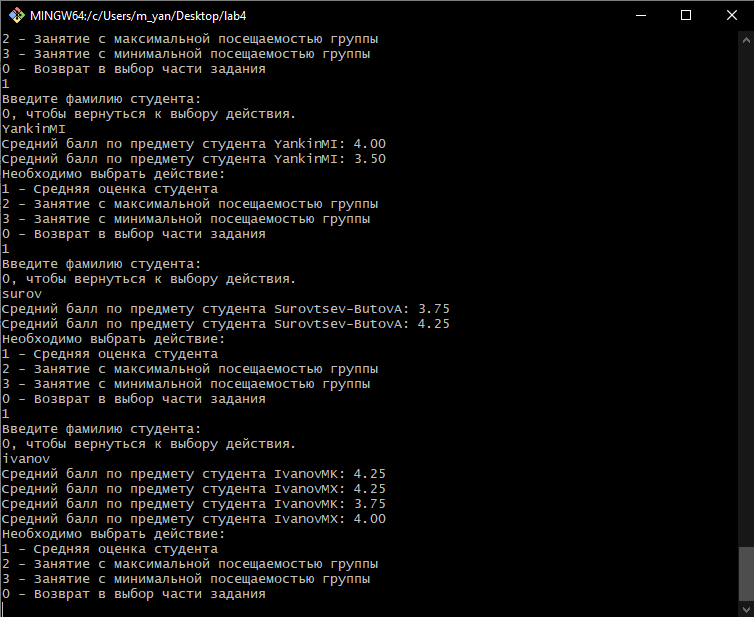
После общения с заказчиком были внесены правки, и теперь скрипт выдает корректный результат (фамилию студента, который вообще не сдал какой-либо тест и номер этого самого теста):



К сожалению, ход разработки второй задачи не очень хорошо задокументирован, в следствие ограниченного времени на разработку. В ходе первой сдачи найдена основная проблема – неправильно описано условие поиска студента по фамилии: в некоторых случая строка не обрабатывалась никак:



После изменения условий поиска в конструкции “grep”, скрипт выдает правильный результат:



# **7. ПРИЛОЖЕНИЕ**

**Код скрипта Program.sh:**

#!/bin/bash

bM=true

while $bM;

do

echo

echo "Приветвуем вас в лучшей программе для работы с вашими файлами!"

echo

echo "1 - Первая задача"

echo "2 - Вторая задача"

echo "0 - Выход."

echo

read number

case "$number" in

"0")

bM=false

;;

"1")

./Part1.sh

;;

"2")

./Part2.sh

;;

\*)

echo "Ошибка: Нет такой команды. Повторите ввод."

;;

esac

done

**Код скрипта Part1.sh:**

#!/bin/bash

bMain=true

groupEnter=true

while $bMain;

do

#создаем и заполняем массив предметов

declare -a subjects

subjects+=( $(find ./labfiles/ -maxdepth 1 -type d -not -name "labfiles" -not -name "students" | sed 's/\(.\*\)$/\1\//') )

#Вводим номер группы

while $groupEnter;

do

echo "Введите номер группы [A-XX-XX]:"

echo "0, чтобы выйти к выбору части задания."

read group

case "$group" in

"0")

unset subjects

exit 1

;;

\*)

group=$( echo $group | sed -e 's/а/A/' -e 's/А/A/' -e 's/a/A/' )

#ищем записи о группе

buffer=$(find "${subjects[0]}" -name "$group-attendance")

if [ -n "$buffer" ] #если строка не пустая

then

groupEnter=false

else

echo "Ошибка: Группа не найдена."

fi

;;

esac

done

echo "Выберите пункт:"

echo "1 - Студенты, у которых есть хотя бы один несданный тест"

echo "2 - Упорядоченный список группы по количеству попыток сдачи теста"

echo "0 - Вернуться к введению номера группы"

read Option

case "$Option" in

"0")

groupEnter=true

;;

"1")

#создаем и заполняем массив, содержащий список группы

declare -a namesArray

namesArray+=( $( find "${subjects[0]}" -name "$group-attendance" | xargs sed "s/\(.\*\) [0-1]\*\$/\1/" ) )

#создаем массив студентов, не сдавших хотя бы один тест

declare -a studentsFailed

#создаем массив номеров заваленных тестов

declare -a testFailed

for subj in ${subjects[@]}; do

#получаем количество тестов по предмету

testsNumber=$( find "$subj"/tests -type f -name "TEST-\*" | grep -c "" )

#получаем название предмета

className=$( echo $subj | sed 's/\.\/labfiles\/\(.\*\)\/.\*/\1/' )

studentsFailed=()

testFailed=()

for (( i=1; i<=$testsNumber; i++ ))

do

for student in ${namesArray[@]}; do

#ищем записи об успешных сдачах теста у студента

foundTest=$( grep "$student;20..;.\{1,2\};[3-5]" "$subj"/tests/TEST-$i | sed '/^ \*$/d')

#если записи не найдено, добавляем в массив несдавших

if [ -z "$foundTest" ]

then

searchResult=false

for elem in ${studentsFailed[@]}; do

if [ $student = $elem ]

then

searchResult=true

fi

done

#добавляем в массив, только если студента нет в массиве несдавших

if [ "$searchResult" = false ]

then

studentsFailed+=( $student )

testFailed+=( $i )

fi

fi

done

done

if [[ ${#studentsFailed[@]} -gt 0 ]]

then

echo "Студенты, не сдавшие хотя бы один тест по предмету $className:"

for (( i=0; i<${#studentsFailed[@]}; i++ ))

do

echo "${studentsFailed[i]} (${testFailed[i]})"

done

else

echo "Нет студентов, не сдавших хотя бы один тест по предмету $className."

fi

done

unset studentsFailed

unset namesArray

unset testFailed

;;

"2")

for subj in ${subjects[@]}; do

#получаем количество тестов по предмету

testsNumber=$( find "$subj"/tests -type f -name "TEST-\*" | grep -c "" )

#получаем название предмета

className=$( echo $subj | sed 's/\.\/labfiles\/\(.\*\)\/.\*/\1/' )

echo "Список группы по количеству попыток сдачи тестов по предмету $className:"

for (( i=1; i<=$testsNumber; i++ ))

do

echo "Тест $i:"

egrep \(^\($group\).\*$\) "$subj"/tests/TEST-$i | sed -e 's/A-..-...//g' -e 's/.20[0-9][0-9].\*$//g' | uniq -c | sort

done

echo

done

;;

\*)

echo "Ошибка: Нет такой команды. Повторите ввод."

;;

esac

unset subjects

done

**Код скрипта Part2.sh:**

#!/bin/bash

bMain=true

while $bMain;

do

#создаем и заполняем массив предметов

declare -a subjects

subjects+=( $(find ./labfiles/ -maxdepth 1 -type d -not -name "labfiles" -not -name "students" | sed 's/\(.\*\)$/\1\//') )

echo "Выберите пункт:"

echo "1 - Средняя оценка студента"

echo "2 - Занятие с максимальной посещаемостью группы"

echo "3 - Занятие с минимальной посещаемостью группы"

echo "0 - Возврат в выбор части задания"

read Option

case "$Option" in

"1")

bTask1=true

while $bTask1;

do

echo "Введите фамилию студента:"

echo "0, чтобы вернуться к выбору действия."

read name

name=$( echo "${name^}" )

case "$name" in

"0") bTask1=false ;;

\*)

buffer=$(grep -rhi ";$name" "${subjects[0]}"/tests)

if [ -n "$buffer" ]

then

for subj in ${subjects[@]}; do

#получаем количество тестов по предмету

testsNumber=$( find "$subj"/tests -type f -name "TEST-\*" | grep -c "" )

#получаем название предмета

className=$( echo $subj | sed 's/\.\/labfiles\/\(.\*\)\/.\*/\1/' )

declare -a namesArray

declare -a namesArrayRaw

namesArrayRaw+=( $(grep -rh ";$name[A-Z]\?[a-z]\?[A-Z]\?[a-z]\?" "$subj"/tests | sed "s/A-..-..;\($name.\*\);20..;.\{1,2\};[2-5]/\1/") )

for nameRaw in ${namesArrayRaw[@]}; do

searchResult=false

for elem in ${namesArray[@]}; do

if [ $nameRaw = $elem ]

then

searchResult=true

fi

done

if [ "$searchResult" = false ]

then

namesArray+=( $nameRaw )

fi

done

declare -a marksArray

for elem in ${namesArray[@]}; do

marksArray=()

summ=0

#получаем массив оценок

marksArray+=( $(grep -rh "$elem" "$subj"/tests | grep ';[3-5]$' | sed "s/.\*;\([3-5]\)$/\1/") )

for i in ${marksArray[@]}; do

summ=$(( $summ + $i ))

done

#получаем среднее

summ=$((100\*summ/testsNumber))

echo "Средний балл по предмету $className студента $elem: $summ" | sed 's/\(..\)$/\.\1/'

done

done

unset marksArray

unset namesArray

unset namesArrayRaw

bTask1=false

else

echo "Ошибка: Студент не найден."

fi

;;

esac

done

;;

"2" | "3")

bTask2=true

while $bTask2;

do

echo "Введите номер группы [A-XX-XX]:"

echo "0, чтобы вернуться к выбору действия."

read group

group=$( echo $group | sed -e 's/а/A/' -e 's/А/A/' -e 's/a/A/' )

case "$group" in

"0") bTask2=false ;;

\*)

buffer=$(find "${subjects[0]}" -name "$group-attendance")

if [ -n "$buffer" ] #если строка не пустая

then

#массив посещаемости занятия

declare -a attendanceArray

for subj in ${subjects[@]}; do

#считаем первую строку, убираем имя - получаем количество занятий по предмету, получаем длину строки = кол-во занятий

classCount=$( find "$subj" -name "$group-attendance" | xargs head -n 1 | sed 's/.\* \([0-1]\*\)/\1/' )

classCount=$(( ${#classCount}-1 ))

#получаем название предмета

className=$( echo $subj | sed 's/\.\/labfiles\/\(.\*\)\/.\*/\1/' )

#получаем массив с количеством посещений каждого занятия

attendanceArray=()

for (( i=0; i<=$classCount; i++ ))

do

attendanceArray+=( $(find "$subj" -name "$group-attendance" | xargs grep -c "[0-1]\{$i\}1[0-1]\{$((17-i))\}") )

done

mIndex=1

mCount=${attendanceArray[0]}

case "$Option" in

"2")

for (( i=1; i<=$classCount; i++ ))

do

if (($mCount<${attendanceArray[$i]}))

then

mIndex=($((i+1)))

mCount=${attendanceArray[$i]}

fi

done

echo "Занятие с максимальной посещаемостью по предмету $className для группы $group: $mIndex."

;;

"3")

for (( i=1; i<=$classCount; i++ ))

do

if ((${attendanceArray[$i]}<$mCount))

then

mIndex=($((i+1)))

mCount=${attendanceArray[$i]}

fi

done

echo "Занятие с минимальной посещаемостью по предмету $className для группы $group: $mIndex."

;;

esac

echo "Число студентов, посетивших занятие: $mCount"

done

unset attendanceArray

bTask2=false

else

echo "Ошибка: Группа не найдена."

fi

;;

esac

done

;;

"0")

bMain=false

;;

\*)

echo "Ошибка: Нет такой команды. Повторите ввод."

;;

esac

unset subjects

done