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**

**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 4 – 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 switch, 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

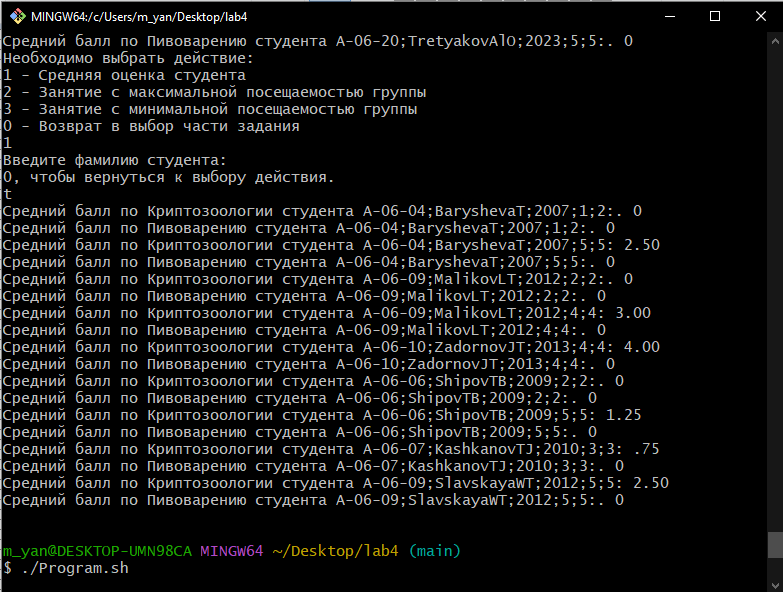
Для тестирования задачи изменялись некоторые файлы в директории ./labfiles/ – с посещаемостью и результатами тестов, а также добавлялась папка с третьим предметом для проверки работоспособности унификации кода. Так, например, в записях с результатами тестов были убраны некоторые записи об успешных сдачах 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.

{Опиши здесь основные моменты, которые у тебя не работали}

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. Это было исправлено:

{сделай скрин пж}

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:

