Тестовое задание для вакансии «Системный программист С»

Версия задания: 10.02.25г.

Условия задания

Требуется разработать утилиту для анализа бинарных данных.

Данные формируются в виде структур `StatData`.

```
typedef struct StatData {
   long  id;
   int   count;
   float cost;
   unsigned int primary:1;
   unsigned int mode:3;
} StatData;
```

Достаточно использовать стандартную библиотеку С и/или системные вызовы Linux, требуется предложить пути оптимизации алгоритмов обработки данных.

Массивы структур, с которыми оперируют программы имеют произвольную длину в диапазоне от 0 до 100000. Все данные помещаются в ОЗУ (объем ОЗУ доступный программам не менее 64Мб).

Сборка программы должна управляться любым удобным инструментом, например, make / CMake / meson.

Компилятор, любой на выбор: gcc /clang.

Требований к переносимости кода нет, архитектура - x86_64, ядро ОС – Linux.

Стандарт языка С – любой удобный.

При проверке будет оцениваться:

- Работоспособность предложенных решений, в том числе возможность скомпилировать код;
- Эффективность алгоритма обработки данных в JoinDump();
- Степень покрытия кода тестами (какие test case проверяются в утилите для тестирования);
- Обработка ошибок.

Сериализация и десериализация

StoreDump()

Напишите функцию для сохранения массива `StatData` (длина массива произвольная) данных в файл, путь до которого передан в качестве аргумента.

LoadDump()

Напишите функцию для чтения массива записей `StatData` из файла (файл сформирован функцией `StoreDump`).

Обработка данных

JoinDump()

Напишите функцию, объединяющую два массива `StatData` Произвольной длины так, чтобы массиве - результате `id` всех записей был уникален. Не гарантируется что id уникален (даже в пределах одного массива).

При объединении записей с одинаковым `id` поля `count` и `cost` должны складываться, поле `primary` должно иметь значение 0 если хотя бы в одном из элементов оно 0. поле `mode` должно иметь максимальное значение из двух представленных. Записей с повторяющимся `id` может быть произвольное. количество.

SortDump()

Напишите функцию, сортирующую массив структур StatData в порядке возрастания значения поля cost.

Утилита обработки данных

Напишите утилиту слияния и сортировки файлов сформированных `StoreDump()`:

Аргументы:

- два пути до двух файлов, сформированных `StoreDump;`
- путь до файла результата.

Алгоритм работы:

- Считываем файлы (пути указаны в первых двух аргументах);
- Объединяем содержимое файлов (`JoinDump`);
- Сортируем содержимое (`SortDump`);
- Печатаем первые 10 записей в виде таблицы;
- При этом значения должны иметь формат:
  - `id` в шестнадцатеричном виде;
  - `count` в десятичном виде;
  - `cost` в научном/экспоненциальном формате с 3 знаками после запятой (`123` -> `'1.230e+2'`);
  - `primary` нужно печатать `"n"` если оно 0 и `"y"` если оно 1;
  - `mode` в бинарном формате (`5` -> `'101'`);
- Сохраняем файл, по пути, который указан в 3 аргументе.

Утилита для тестирования

Разработать утилиту тестирования выполняющую проверку утилиты обработки данных.
Утилита должна:

- Сформировать файлы данных для утилиты обработки данных из массивов данных определённых в ней в качестве констант;
- Запустить утилиту обработки данных;
- Считать результат обработки и сравнить его с эталонными данными (константами в памяти утилиты тестирования);
- При расхождении ожидаемого результата с полученным печатать сообщение об ошибке (в stderr);
- При успешном прохождении теста печатать отчёт о времени выполнения тестов и списке пройдённых тестов (в stdout).

Пример проверочных данных:

```
/* Содержимое для исходных файлов */
const StatData case_1_in_a[2] =
{{.id = 90889, .count = 13, .cost = 3.567, .primary = 0, .mode=3 },
 {.id = 90089, .count = 1, .cost = 88.90, .primary = 1, .mode=0 }};
const StatData case_1_in_b[2] =
{{.id = 90089, .count = 13, .cost = 0.011, .primary = 0, .mode=2 },
 {.id = 90189, .count = 1000, .cost = 1.00003, .primary = 1, .mode=2}};
/* Ожидаемый результат обработки */
const StatData case_1_out[3] =
{{.id = 90189, .count = 1000, .cost = 1.00003, .primary = 1, .mode = 2 },
 {.id = 90889, .count = 13, .cost = 3.567, .primary = 0, .mode = 3 },
 {.id = 90089, .count = 14, .cost = 88.911, .primary = 0, .mode = 2 }};
```