МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Московский Авиационный Институт
(Национальный Исследовательский Университет)

Институт №8 "Информационные технологий и прикладная математика"
Кафедра 806 "Вычислительная математика и программирование"

Курсовая работа
по курсу "Вычислительные системы"
2 семестр

Задание 6
Обработка последовательной файловой структуры на языке Си

Автор работы:
студент 1 курса, группы М8О-106Б-20
Леухин М. В.
Руководитель проекта:
Дубини А. В.

Москва, 2020

Содержание

1 Постановка задачи

Разработать последовательную структуру данных для представления простейшей базы данных на файлах в СП Си в соответствии с заданным вариантом. Составить программу генерации внешнего нетекстового файла заданной структуры, содержащего представительный набор записей (не менее 20). Распечатать содержимое сгенерированного файла в виде таблицы и выполнить над ним заданное действие для 3–5 значений параметров запроса p и распечатать результат. Действие по выборке данных из файла оформить в виде отдельной программы с параметрами запроса, вводимыми из стандартного входного текстового файла, или получаемыми из командной строки UNIX. Второй способ задания параметров обязателен для работ, оцениваемых на хорошо и отлично.

Входные данные: данные из приказов о зачислении №240808/ст от 24.08.2020 и №260811/ст от 26.08.2020 (источник: *https://priem.mai.ru/results/orders*).

Задание: рассчитать средние баллы по предметам студентов, зачисленных на направление 01.03.02 Прикладная математика и информатика и вывести таблицу студентов с баллами выше средних в зависимости от вводя пользователя.

Структура проекта: в корневой директории проекта расположены 2 файла и 1 директория. Файл `src.txt` — текстовый файл с входными данными. Файл `src.bin` — бинарный файл с обработанными данными из файла `src.txt`. В директории `code/` расположены ещё 3 файла. Файл `code/person.h` — заголовочный файл с объявлением структуры `Person`. Файл `code/encode.c` — программа обработки входных текстовых данных, на вход получает файлы `src.txt` и `src.bin` (при отсутствии файла `src.bin` он будет создан — наличие этого файла в исходном состоянии проекта необязательно). Файл `code/main.c` — программа обработки входного бинарного файла, на вход получает файл `src.bin` и (опционально) параметры.

Параметры программы:

- -m — вывести средний балл по математике и распечатать таблицу студентов с баллом по математике выше среднего.
- -p — вывести средний балл по физике и распечатать таблицу студентов с баллом по физике выше среднего.
- -r — вывести средний балл по русскому языку и распечатать таблицу студентов с баллом по русскому языку выше среднего.
- -h — учитывать только студентов, получивших место в общежитии.

Ключи можно комбинировать. Например: вызов `./main src.bin -hmr` выведет таблицу студентов, получивших место в общеитии, с баллами по маетматике и русскому языку выше средних.

При вызове без параметров (или только с параметром -h) будет выведена таблица студентов с баллами по всем трём предметам выше средних.

2 Теоретическая часть

2.1 Файлы в unix-подобных операционных системах

Везде далее рассматриваются unix-подобные операционные системы (если не указано иное).

Файл — именованная область внешней памяти, выделенная для хранения массива данных. Данные, содержащиеся в файлах, имеют самый разнообразный характер: программы на алгоритмическом или машинном языке; исходные данные для работы программ или результаты выполнения программ; произвольные тексты; графические изображения etc. На самом деле, всё является файлами — директории, физические устройства, даже сетевые соединения.

Файловая система — функциональная часть операционной системы, обеспечивающая выполнение операций над файлами. Примеры операционных систем — FAT, NTFS, UDF etc. В основном в системах с ядром Linux используется файловая система ext4.

2.2 Ввод-вывод в Linux. Файловые дескрипторы

В зяыке C для осуществления файлового ввода-вывода используются механизмы, описанные в стандартной библиотеке языка `stdio.h`. Консольный ввод-вывод - это всего лишь частный случай файлового ввода-вывода. В C++ для ввода-вывода чаще используются потоковые типы данных. Однако все эти механизмы явлвяются лишь надстройками над низкоуровневыми механизмами ввода-вывода ядра операционной системы.

Пользовательские программы взаимодействуют с ядром операционной системы посредством специальных механизмов, называемых *системными вызовами*. Внешне системные вызовы реализованы в виде обычных функций языка C, однако каждый вызов такой функции обращается к ядру ОС. Список всех системных вызовов можно найти в файле `/usr/include/asm/unistd.h`. Основные системные вызовы, обеспечивающие ввод-вывод: open(), close(), read(), write(), lseek() и некоторые другие.

В языке C при осуществлении ввода-вывода используется указатель FILE*. Даже функци printf() в итоге сводится к вызову vprintf(stdout, ...), где константа stdout имеет тип struct _IO_FILE*, синонимом которого является FILE*. Но почему stdout, который является стандартным потоком вывода, в C является FILE*?

Стандартные потоки — потоки процесса потоки процесса, имеющие номер (дескриптор), зарезервированные для выполнения некоторых функций. **Дескриптор** — неотрицательное число, являющееся идентификатором потока ввода-вывода. При запуске программи на C автоматически открываются стандартный поток ввода stdin, сдандартный поток вывода stdout и стандартный поток ошибок stderr с дескрипторами 0, 1 и 2 соответственно. Все эти стандартные потоки по факту являются файлами.

Про организацию потоков ввода-вывода (и в целом процессов) в Linux можно говорить очень много, но не будем вдаваться в столь низкоуровневые подробности и вернёмся к работе с файлами в языке C.

2.3 Работа с файлами в языке C

Язык программирования C поддерживает множество функций для работы с файлами, которые объявлены в заголовочном файле stdio.h.

2.3.1 FILE* fopen(const char* fname, const char* mode) FILE* freopen(const char* fname, const char* mode, FILE* stream)

Функция fopen() открывает файл, имя которого указано аргументом fname и возвращает связанный с ним указатель. Тип операций, разрешённых над файлом, определяется аргументом mode.

- "r" — создаёт файл для записи.
- "w" — создаёт файл для записи.
- "a" — дописывает информацию к концу файла.
- "r+" , "w+" — открвыает файл для чтения/записи.

Если функции fopen() удалось открыть указанный файл, возвращается указатель FILE*, иначе возвращается NULL.

Также существует функция freopen(), которая используется для связывания текущего потока с новым файлом.

2.3.2 int fclose(FILE* stream) int fcloseall(void)

Функция fclose() закрывает файл, связанный со stream, и очищает буфер. В случае успеха возвращает 0, иначе ненулевое значение. Попытка закрытия уже закрытого файла является ошибкой.

Функция fcloseall() закрывает все открыте потоки, кроме stdin, stdout, stderr, stdaux, stdprn.

2.3.3 int fgetc(FILE* stream) int fgetchar(void)

Функция fgetc() возвращает следующий за текущей позицией символ из входного потока и даёт приращение указателю положения в файле. При достижении конца файла fgetc() возвращает EOF (также EOF возваращется в случае ошибки).

Макрос fgetchar() определён как fgetc(stdin)

2.3.4 size_t fread(void* buf, size_t size, size_t count, FILE* stream) size_t fwrite(const void* buf, size_t size, size_t count, FILE* stream)

Функция fread() считывает count объектов размера size из потока stream и сохраняет их в блоке памяти, на который указывает buf. При этом указатель положения в файле увеличивается на считанное количество байт. Функция fread() возвращает количество действительно считанных объектов.

Функция fwrite() записывает count объектов размера size в поток stream из блока памяти, на который указывает buf. При этом указатель положения в файле увеличивается на считанное количество байт. Функция fwrite() возвращает количество действительно записанных объектов.

2.3.5 int ferror(FILE* stream)

Функция ferror() проверяет, имеются ли файловые ошибки в данном потоке stream. Возвращает 0 в случае отсутствия ошибок, иначе возвращает ненулевую величину.

2.3.6 int fseek(FILE* stream, long offset, int origin)
long ftell(FILE* stream)
void rewind(FILE* stream)

Функция fseek() устанавливает указатель положения в файле, связанного со stream, в соответствии со значениями offset и origin. Аргумент offset — это сдвиг в байтах от позиции origin до новой позиции. В качестве аргумента origin могут быть следующие встроенные константы: SEEK_SET — начало файла; SEEK_CUR — текущая позиция; SEEK_END — конец файла. В случае успеха fseek() возвращает 0, иначе ненулевое значение.

Функция ftell() возвращает текущую позицию указателя.

Функция rewind() устанавливает указатель на начало потока.

3 Практическая часть

3.1 Описание подхода к решению поставленной задачи

Что имеем: файл src.txt, программу code/encode.c, которая должна обработать файл src.txt, поступающий на вход, и записать информацию в бинарный файл src.bin, и программу code/main.c, которая непостредственно обрабатывает бинарный файл src.bin и печатает таблицу в соответствии с вводом пользователя.

В программе code/encode.c открываем для чтения файл src.txt и для записи файл src.bin с помощью функци fopen(). При этом проверяем действительно ли удалось открыть файлы путём сравнения полученного указателя с NULL. Далее в цикле с помощью фнукции fscnaf() считываем строки из входного текстового файла и записываем в бинарный файл с помощью функции fwrite(). При этом также отслеживаем наличие ошибок путём сравнения возвращаемых функциями fscanf() и fwrite() значений с тем, сколько объектов они должна были считать/записать. Для хранения информации будем использовать структуру Person, объявленную в файле code/person.h. В конце программы проверяем наличие ошибок в файлах src.txt и src.bin с помощью функции ferror().

В программе code/main.c необходимо для начала открыть входной бинарный файл, при этом проверив получилось ли действительно его открыть. Далее необходимо считать введённые пользователем параметры. Для их хранения удобно будет использовать множество, представив его в виде unsigned int. При это последовательность хранимых параметров следующая: -h, -r, -p, -m (Например, если пользователь ввёл -hm, то значение unsigned int будет равно 1001_2 или 9_{10}). Следующим принципиально важным моментом является то, что нельзя хранить одновременно всю информацию из бинарного файла в памяти во время выполнения программы — размер файла ограничен размером дискового пространства, а размер памяти, отведённой под программу — размером оперативной памяти (которая значительно меньше размера дискового пространства). Потому будем 2 раза считывать информацию из бинарного файла — при этом в памяти программы будет храниться лишь одна структура Person в определённый момент времени. При первом проходе рассчитываем средние значения баллов, при втором — печатаем таблицу в соответствии с вводом пользователя. После первого прохода необходимо будет вернуть указатель положения в файле на начало с помощью функции rewind().

3.2 Используемые собственные функции и переменные

Используемые в программе собственные переменные:

- **unsigned int avr, avr_m ...** — переменные, в которых хранится значения средних баллов. Использование типа unsigned int обусловлено тем, что баллы не могут быть отрицательными.
- **FILE* in** — указатель на поток; входной бинарный файл.
- **unsigned int set_int** — множество передаваемых программе параметров.

- **Person p** — структура, в которой хранится обрабатываемый в данный момент объект типа Person

Используемые в программе собственные функции:

- **void help()** — функция вывода справки. Вызывается в случае неверного использования программы code/main.c.
- **void key_error(char c)** — функция вывода ошибки неверного использования ключа char c.
- **bool check(Person* p, unsigned int set_int)** — функция, определяющая выводить ли в таблице объект Person p в соответствии с запросом пользователя, информация о котором передана в множестве unsigned int set_int.
- **void print(Person* p)** — функция печати строки таблицы.