Файлы, бинарная обработка

Шокуров Антон В.
shokurov.anton.v@yandex.ru
http://машинноезрение.рф

12 февраля 2017 г.

Версия: 0.10

Аннотация

В данной заметке будет показано как взаимодействовать с бинарными файлами. Часть вещей была уже рассмотрена в заметке про текстовые файлы. В данной заметке также речь пойдет об очередных вспомогательных функциях стандартной библиотеки.

Обработка бинарных файлов: поблочное считывание/запись, манипуляция смещением внутри файла, .

Предварительная версия!

1 Файлы

В текстовых файлах главное то, что данные там читабельны человеком. Более того, они им редактируемы через стандартный текстовый редактор.

В бинарные же файлах данные хранятся как последовательность бит. Данные файлы не читабельны человеком и их редактировать можно только специальным программным обеспечением. В самом общем виде – шестнадцатеричный редактор.

Можно написать и соответствующую программу на Си. Для обработки таких данных в библиотеке имеются функции fread и fwrite, а также fseek и ftell.

1.1 Инициирование взаимодействия

Открытие файла производится аналогично тому как это делается для текстовых файлов с той лишь разницей, что нужно (желательно) указать бинарный режим посредством ключа b.

Открытие Для считывания или записи бинарных данных файл сначала необходимо его открыть в соответствующем режиме.

```
1  FILE *in = fopen("input", "rb");//Открыли файл в бинарном
2  //режиме (b) для чтения (r). В переменную in сохранили
3  //хендлер файла, т.е. ссылку на него.
4  if( in == NULL )//По хорошему нужно проверить факт
5  {//успешного открытия файла. Если хендлер равен NULL, то
6    //это значит, что файл не открылся и нам нужно что-то
7    //с этим сделать, например, завершить программу.
8    printf("Error openning file!\n");
9    return -1;
10 }
```

Суть первого аргумента функции fopen (см. стр. 1) не претерпела изменения, а именно – она указывает на имя файла в кавычках.

В данном случае текстовый файл открыт для чтения (для записи нужно было бы использовать w вместо r).

Закрытие Делается в точности так, как и для текстового режима, а именно – данное действие необходимо для завершения взаимодействия с данным файлом. Файл закрывается вызовом функции fclose с указанием хендлера файла:

```
1  FILE *in = fopen("input", "rb");//Открыли файл.
2  fclose( in );//Закрыли файл.
```

После завершения работы с файлом (считывание и запись) его необходимо обязательно закрыть (дабы операционная система могла освободить часть выделенных под него ресурсов).

1.2 Обработка данных

Файл открывается с целью обработки данных находящихся в файле. В случае бинарных файлов данные лучше представлять как большой массив, каждый элемент которого имеет тип байт (с размером есть тонкости, пусть он будет 8-битный). При считывании или записи данные обрабатываются поэлементно, т.е. никакие смысловые нагрузки самих данных на это не влияют. Последние означает, что пробелы считываются как отдельные элементы, они никак не пропускаются. Суть бинарного режима в том, что данные (элементы) считываются группой начиная с какого-то заданного номера элемента по какой-то.

Поэтому с ним ассоциируются два класса операций операций: само считывание/запись и сдвиг/считывание смещения в файле. Считывание (fread) и запись

(fwrite) осуществляется последовательно, т. е. система сама сдвигает указатель, поэтому в некоторых случаях можно обойтись без его явного задания. Но в общем случае, в частности, для повышения эффективности необходимо применять функции связанные с считывание (ftell) и записью (fseek) указателя внутри файла.

Группа элементов файла Как при считывании, так и при записи используется одна и та же концепция. Последовательная группа элементов (из байтов) файла задается как последовательность объектов, каждый из которых имеет некий размер в байтах. Таким образом, как при считывании, так и при записи, необходимо задавать не общее количество необходимых базовых элементов (составляющих файл), который нужно считать из файла, а как количество объектов и размер каждого из этих объектов.

То куда или откуда считать данные в памяти компьютера задается соответствующим указателем, что удобно. Допустим, ты знаешь, что в файле хранятся некое количество объектов определенного типа (например, типа my_object), тогда:

```
1 int n_objects = 10;//Количество объектов.
2 int sz_object = sizeof(my_object);//Вычисляет размер
3 //каждого из объектов.
4 my_object objs[n_onjects];//Массив объектов my_object.
5 //Далее идет код обработки массива.
```

Считывание Для считывания используется функция fread. Эта функция имеет только одно общее с её аналогом для текстового режима функцией fscanf, а именно – ей передается хэндлер того файла с каким необходимо взаимодействовать. Хэндлер указывается в качестве последнего, 4-го аргумента.

```
1 FILE *in = fopen("input", "rb");//открыли файл
2 uint8_t d[5];
3 //Считываем значение первых 5 байтов из файла.
4 //Первый аргумент (d) является указателем на память, куда
5 //необходимо записать данные. Второй и третий задают,
6 //соответственно, размер элемента и их количество.
7 int ret = fread( d, sizeof(uint8_t), 5, in);
8 //В переменную ret указывается количество успешно
9 //считанных элементов, в данном случае величина ret <= 5.
```

Первый элемент задает Сишный указатель в памяти куда необходимо записать данные. Второй и третий элемент задаются в рамках того, что было написано ранее, а именно – размер самого элемента целевого массива и количество элементов.

Возвращаемое значение (в данном случае, сохранено в переменную ret) сообщает о том, сколько элементов массива удалось успешным образом считать. Последнее означает, что невозможна ситуация при которой считалось пол элемента. Успешно считывается всегда счетное количество элементов, а не дробное. Последним удобно пользоваться.

Следует иметь ввиду, что нет никаких гарантий, что все запрашиваемые данные будут считаны успешно, т.е. считается нормальным ситуация, при которой считываются не все данные. Для успешного считывания данных необходимо написать, соответствующий цикл, который в отмеченном случае вызывает функцию считывания заново (т.е. fread) при этом со сдвинутым указателем (d) на нужное актуальное (то которое уже было успешно считано) количество элементов и скорректированным количеством (тоже с учетом того сколько уже было успешно считано данных). В случае ошибки (например, указатель находится в конце файла) возвращается значение -1.

Упражнение. Напиши функцию на Си, которая корректно считывает большие массивы данных.

Дальше можно выполнять обработку ранее считанных данных, например, напечатать их значение в консоль:

```
1 int i;
2 for(i = 0; i < 5; i++)
3   printf("%d ", d[i]);//Печатаем значение байтов.
```

Запись Запись выполняется полностью аналогично считыванию.

```
1  FILE *in = fopen("input", "wb");//открыли файл
2  uint8_t d[5]={1,2,3,4,5};
3  //Записываем значение от 1 до 5 как последовательность
4  //байтов в начало файла.
5  //Первый аргумент (d) является указателем на память,
6  //откуда необходимо считать данные. Второй и третий
7  //задают, как и ранее, размер элемента и их количество.
8  int ret = fwrite( d, sizeof(uint8_t), 5, in);
9  //В переменную ret указывается количество успешно
10 //считанных элементов, в данном случае величина ret <= 5.
```

Все аргументы аналогичны вызову fread с той лишь разницей, что будет выполнена запись массива из памяти в начало файла.

1.3 Внутрефайловое смещение

Для манипуляции со смещением используются две функции: для считывания значения – ftell, для задания значения – fseek.

Считывание В любой момент можно считать текущую позицию смещения в файле. Отмечу, что она одна и та же как для считывания, так и для записи бинарных данных.

```
1 FILE *in = fopen("input", "wb");//открыли файл
2 //...делаем что то с файлом, например считываем данные.
3 //Для считывания текущей позиции в файле вызываем ftell:
4 int pos = ftell(in);//В качестве аргумента указывается
5 //хэндлер. В переменную pos возвращается текущая позиция.
6 printf("current file offset %d ", pos);
```

Вместо типа int рекомендуется использовать тип long. Благодаря нему можно будет считывать файлы существенно большего размера.

Упражнение. Какой размер файла задается смещением int (точнее, uint32_t)? Какой long (точнее uint64_t)?

Задание Внутре файловое смещение можно не только считать, но и задать. Для этого используется функция fseek. Для задания смещения ей необходимо передать хэндлер файла, величину смещения и, внимание, тип смещения. Смещение можно установить относительно:

начала файла	SEEK_SET
конца файла	SEEK_END
текущей позиции	SEEK_CUR

```
1 FILE *in = fopen("input", "wb");//открыли файл
2 //Для перемещения текущей позиции в файле в позицию 3го
3 //элемента вызываем fseek с указанием нужных аргументов:
4 //В качестве первого аргумента указывается хэндлер.
5 int ret = fseek(in, 3, SEEK_SET);//Третий - тип смещения.
6 //Второй - величину смещения.
```

В данном случае, было выполнено перемещения указателя на 3байт (второй аргумент равен 3) с начала файла (третий аргумент равен SEEK_SET).

Упражнение. Как вычислить размер файла (можно использовать как fseek, так и ftell)? Напиши соответствующую программу.