**Примечания о RDMS23 - программе чтения дисков MS-DOS под CP/M.**

Джон Хэствелл-Баттен {John Hastwell-Batten}

системный оператор

Tesseract RCPM+

P.O. Box 242

Dural, NSW 2158

Австралия

Программа RDMS23 использует BIOS CP/M для чтения файлов с гибких дисков MS-DOS и BDOS CP/M для записи файлов на диск CP/M.

Программа RDMS23 происходит из RDMSDOS.C, которая была написана Бобом Алверсоном {Bob Alverson}. В общественное достояние была выпущена версия 1.1, написанная для компиляции Small-C и работы в CP/M 2.2.

С тех пор я выполнил большую работу по этой программе. Помимо исправления нескольких существенных ошибок, основные изменения вызваны желанием заставить ее работать в используемой мной операционной системе CP/M версии 3.1. Я также добавил несколько комментариев, включая некоторые примечания по реализации. Надеюсь, последняя версия также сможет работать в CP/M 2.2.

Программа RDMS23 написана с помощью Hi-Tech C, но можно использовать практически любой компилятор "K&R" C, поддерживающий функцию malloc() или эквивалентную. Даже это не является существенным препятствием. Вы можете заменить динамически выделяемую память статической за счет увеличенного размера программы. (Программа была изначально написана именно так.)

Работа программы в некоторой степени зависит от того, как BIOS хост-системы CP/M обрабатывает двусторонние дискеты. Хотя я пытался удовлетворить некоторые из наиболее часто используемых методов, возможно, вам придется изменить программу. В некоторых случаях вам, возможно, придется изменить свой BIOS, если вы действительно хотите использовать эту программу. Некоторые замечания по адаптации программы появятся позже в этой документации.

**Предварительные требования:**

Есть некоторые абсолютные требования, которые должны быть выполнены, прежде чем вы можете использовать эту программу:

1. Ваша система должна иметь 5.25" дисковод с 40 дорожками, который может читать двусторонние дискеты двойной плотности.
2. BIOS должен быть способен читать по крайней мере девять 512-байтовых секторов на каждой дорожке.

**Будущая разработка:**

v2.x Сохранение целостности CRC (возможно).

v3.x Разрешить запись на диски MS-DOS.

**Как использовать RDMS23**

Активация программы

Поместите RDMS23.COM на любом удобном диске. На моей системе я поместил его на диск A: в пользовательскую область 0 и сделал его системным файлом, чтобы я мог запускать его с любого диска и любой пользовательской области. Пользователи ZCPR смогут выполнить тот же трюк.

Вероятно, стоит переименовать программу в нечто вроде RDMS.COM, чтобы избежать ненужного ввода. С этого момента эти примечания предполагают это имя.

Теперь зарегистрируйте диск/пользователя, куда вы хотите поместить файлы, скопированные с диска MS-DOS.

Активируйте программу, введя ее имя. RDMS нужно указать дисковод, в который загружен гибкий диск MS-DOS. Вы можете включать букву диска в командной строке, при необходимости, иначе RDMS запросит ее. Формы командной строки следующие:

**rdms** или **rdms d**

где '**d** ' - буква диска MS-DOS.

[Другие параметры командной строки описаны позже.]

RDMS считает таблицу размещения файлов и корневой каталог диска MS-DOS и отобразит корневой каталог. После этого можно копировать файлы с гибкого диска MS-DOS на диск CP/M.

**Копирование файлов**

Для копирования файл с гибкого диска MS-DOS на зарегистрированный диск/пользователя CP/M, необходимо ввести имя одного из файлов, отображенных из каталога MS-DOS. Несколько файлов могут быть скопированы с использованием обычных соглашений о "подстановочных символах" имен.

В любое время можно получить новое отображение каталога, не предоставив имени файла вообще, т.е. просто нажав клавишу <cr>.

**Поддиректории**

В списке каталога файлы подкаталогов обозначаются "типом" ".dir". Если вы введете имя подкаталога (без ".dir"), то вместо копирования файла RDMS переключится в этот подкаталог и отобразит его содержимое. Затем вы можете продолжить копирование файлов, но будьте внимательны, потому что диск MS-DOS может содержать два разных файла с одинаковым именем в разных каталогах. Если вы скопируете второй файл, он перезапишет первый.

В любом подкаталоге первые две записи сами являются каталогами. Один будет идентифицирован одной точкой, другой - двумя точками. Запись с двумя точками является единственной интересной. Если вы введете ".." в качестве имени файла, то RDMS вернется к предыдущему уровню каталога MS-DOS. Это дает вам возможность пройти через всю иерархию каталогов MS-DOS.

**Выход из программы**

Однозначно, самый правильный способ завершить программу состоит в том, чтобы ответить на запрос "Enter filename: " (Введите имя файла:) ^Z <cr> (т.е. ввести Ctrl-Z и возврат каретки). Если вы сделаете это, то программа правильно повторно зарегистрирует диск CP/M.

На некоторых системах программу возможно завершить с помощью Ctrl-C, но не жалуйтесь мне, если после этого вам придется перезагружать компьютер.

**Технические замечания**

Диски MS-DOS выпускаются в четырех вариантах; Одно- и двухсторонние диски с 8 или 9 секторами по 512 байт на дорожку. Двусторонние диски MS-DOS записываются в режиме "цилиндра". Сектора с 1 по 8/9 записываются на стороне 0, затем сектора с 1 по 8/9 записываются на стороне 1 перед переходом на следующую дорожку. Нет стандартного способа сообщить BIOS CP/M, какая сторона диска должна быть доступна. Все, что вы можете сделать - это задать номера дорожек и секторов перед выполнением операции ввода-вывода. Вы должны составить программу вычисления дорожки и сектора таким образом, чтобы ваш BIOS мог получить доступ к правой части диска. Возможно, вам даже придется изменить свой BIOS!

В простейшем случае ваш BIOS может также использовать режим "цилиндра" для записи и декодирования номера стороны от младшего бита номера дорожки.

|  |  |  |
| --- | --- | --- |
| Дорожка BIOS | Цилиндр | Сторона |
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 2 | 1 | 0 |
| 3 | 1 | 1 |
| 4 | 2 | 0 |
| . | . | . |
| 77 | 38 | 1 |
| 78 | 39 | 0 |
| 79 | 39 | 1 |

Другим распространенным способом передачи в BIOS информации о том, к какой стороне диска обращаться, является установка старшего бита номера сектора, чтобы запрос сектора в диапазоне от 1 до 8/9 обращался к стороне 0, запрос сектора в диапазоне от 129 до 136/137 обращался к стороне 1. В настоящее время программа настроена на использование этого метода.

Некоторые BIOS предполагают, что все дорожки ниже количества цилиндров на диске находятся на стороне 0, все дорожки с более высоким номером находятся на стороне 1.

Ни одна из вышеперечисленных проблем не представляет серьезных затруднений. Если ваш BIOS использует какой-либо из этих методов, вы можете изменить программу, чтобы справиться. У вас возникнут серьезные проблемы, если ваш BIOS для обработки двусторонних дисков использует метод "длинной дорожки", где сектора 1-8/9, как обычно, на стороне 0, но где предполагается, что сектора на стороне 1 физически зарегистрированы с разными номерами из соответствующих секторов на стороне 0. **Нет никакого выхода**, кроме изменения вашего BIOS! Аналогично, ваш BIOS должен либо ожидать, что байты идентификатора сектора на стороне 1 действительно содержат код стороны 1, либо **не осуществлять** проверку стороны при поиске сектора на стороне 1.

Надеюсь, все это имеет смысл.

Есть некоторые параметры командной строки, которые фактически указывают RDMS23, какой метод использовать на двусторонних дискетах. Возможно, вы захотите попробовать их перед попыткой перекомпилировать программу.

|  |  |
| --- | --- |
| -h | (для "высокого бита), говорит RDMS23, что BIOS использует старший бит номера сектора, чтобы сигнализировать чтение со стороны 1. |
| -c | (для режима "цилиндр"), говорит RDMS23, что BIOS расценивает все четные дорожки двусторонней дискеты как расположенные на стороне 0 и все дорожки с нечетным номером на стороне 1. |
| -x | (для "удлиненной поверхности"), сообщает RDMS23, что ваш BIOS считает, что дорожки 0-39 находятся на стороне 0, а дорожки 40-79 - на стороне 1. |

Просто включите подходящую опцию где-нибудь в командной строке, если ваш BIOS использует один из вышеупомянутых методов, например.

**rdms23 d -x** или **rdms23 -x d**

Помните, если вы не укажете RDMS, какой метод использовать, он задействует "-c". Возможно, вы захотите изменить это предположение, как только определите, какой метод используется вашим BIOS с тем, чтобы не указывать его RDMS при каждом выполнении. Вы можете перекомпилировать программу, если имеете подходящий компилятор C, но вы можете также исправить RDMS.COM с помощью DDT, ZSID или любого другого. Чтобы сделать это, загрузите RDMS.COM под ZSID и используйте команду "dump" ASCII/HEX для сканирования программы. Ищите ряд символов "Method=H", измените "H" на "C" или "X" и сохраните измененную программу. Последовательность будет правильным путем в программе и чтобы сэкономить время, я предлагаю вам начать поиск около 3000h.

**Известные ошибки**

CRC, вычисленная в MS-DOS, не совпадет с CRC, вычисленной для файла, как только он был скопирован в CP/M. Сам файл в порядке, и я думаю, что различие является результатом механизмов обнаружения конца файла под этими двумя операционными системами.

В CP/M+ и CCP/M есть механизм сохранения точной длины файла, таким образом, эта проблема все же может быть решена.

**Авторское право**

(C)1984 Боб Алверсон {Bob Alverson}

(C)1985 Джон Хэствелл-Баттен {John Hastwell-Batten}

Это программа **общественного достояния**. Вы можете распространять, копировать и использовать эту программу в некоммерческих целях. Мы ничего не требуем с вас за это. Не справедливо если вы будете зарабатывать деньги или украдете кредит наших усилий.

Если вы исправляете какие-либо ошибки или вносите какие-либо улучшения повторно предоставьте измененный **исходный код** общественному достоянию.

**Информация о таблице размещения файлов**

Таблица размещения файлов использует 12-разрядную запись для каждого блока распределения на диске. Эти записи упакованы, по два на каждые три байта. Содержимое записи с номером N определяется путем

1. умножением N на 1,5;
2. добавление результата к базовому адресу таблицы распределения;
3. выборка 16-разрядного слова по этому адресу;
4. Если N нечетное (так что N\*1,5 не является целым числом), сдвиньте слово вправо на четыре бита;
5. маска 12 битов (AND с 0FFF hex)

Нулевая запись используется в MS-DOS как ловушка для конца файла и передается в BIOS, чтобы помочь определить формат диска. Запись 1 зарезервирована для будущего использования. Первому доступному блоку распределения назначается запись номер два, и хотя он является первым, он называется кластером 2. Записи, превышающие 0FF8H, являются маркерами конца файла. Записи нуля не распределены. В противном случае содержимое записи FAT является номером следующего кластера в файле.

Кластеры с плохими секторами помечены FF7H. Любое ненулевое число будет означать, что эти кластеры показаны как выделенные, но не являются частью какой-либо цепочки распределения и, следовательно, никогда не будут назначены файлу. Конкретное число выбрано так, чтобы программы проверки диска знали, что делать (то есть кластер с записью FF7H, не находящийся в цепочке, не является ошибкой).

**Примечания к версии 2.0** (JRS)

Отображение каталога теперь указывает на другой каталог, перечисляя его с расширением имени файла ".dir" (строчными буквами). Войдите в подкаталог, введя его имя (без ".dir"). Вернитесь на предыдущий уровень, набрав ".." в качестве имени файла. Обратите внимание, что все файлы, скопированные из любого каталога, находятся в том же месте под CP/M.

**Примечания к версиям 1.2, 1.3 и 1.4** (JRS)

Я не мог запустить эту программу в ее первоначальном виде. Вызовы BIOS были специфичны для CP/M 2.2, а у меня больше нет системы 2.2. Для меня первая задача состояла в том, чтобы преобразовать программу, чтобы использовать функцию 50 BDOS вместо них. Программа все еще не работала, потому что ожидала, что BIOS деблокирует физические сектора и возвратит 128-байтные секторы. BIOS CP/M Plus имеет дело с физическими секторами, и в нормальном режиме работы деблокирование секторов выполняется BDOS. Вместо того, чтобы исключить любую возможность запуска этой программы в системе 2.2, я добавил некоторую простую логику деблокирования в подпрограмму, выполняющую фактическое чтение диска.

Следующий прием заключался в том, чтобы заставить BIOS читать правильные физические сектора. CP/M не имеет определенного метода для работы с многосторонними дисками, но я попытался удовлетворить некоторые из наиболее распространенных методов, используемых разработчиками системы. Любой, пытающийся адаптировать эту программу к другой машине, должен знать, как BIOS выбирает сектора для чтения с каждой поверхности диска. Эта программа работает на системах, где вторая сторона диска выбирается путем установки старшего бита номера сектора и более распространенного случая записи "цилиндра". Она не была протестирована в системе, которая обычно заполняет всю сторону 0 перед записью стороны 1.

**Примечания по обратному переносу в CP/M 2.2** (JRS)

В более ранних версиях этой программы я не пытался сохранить совместимость с CP/M 2.2, но обратный перенос этой программы было довольно легким. Начиная с версии 2.0, я повторно добавил код для CP/M 2.2, который был изменен до неузнаваемости при преобразовании для CP/M+, и разрешил компиляцию программы для любой операционной системы. В версии 2.1 компилируются оба наборов кода и зависимости от операционной системы определяются во время выполнения.

Есть только один нюанс: ни один из CP/M 2.2 код ни разу не проверялся!

**Примечания по портированию под другие компиляторы C** (JRS)

Hi-Tech C является полным компилятором "K&R" C, поэтому перенос этого кода на другие компиляторы K&R не должен вызывать много проблем.

Если вы планируете преобразовать эту программу в нестандартные диалекты языка C, пожалуйста, не уничтожайте совместимость с K&R. В частности, если вы преобразуете в BDS-C, пожалуйста, используйте набор файлов #include из CUG 39 для "подделки" совместимости.

Пожалуйста, прочитайте примечания в начале функции bios3.

**Заметки о сохранении точной длины файла** (JRS)

CP/M Plus и Concurrent CP/M предусматривают маркер для записи точной длины файла в каталоге, позволяя байту S1 в записи каталога содержать число байтов в последней (128-байтовой) записи файла. Так как количество записей известно, точная длина файла может быть вычислена.

Информация записывается в каталог с помощью функции 30 BDOS, "Установка атрибутов файла". Желаемое содержимое байта S2 помещается в байт CR (текущая запись) правильно отформатированного FCB, и устанавливается атрибут интерфейса F6. Функция 30 BDOS копирует поле CR в байт S1 записи каталога для файла, из которого он может быть извлечен при вызове BDOS при открытии (30), первом (17) и/или последующем (18) поиске.

На сегодняшний день ни одна программа никогда не пользовались этой функцией, и теперь я предлагаю следующий стандарт для того, чтобы быть последовательными и совместимыми с существующими программами и файлами, которые они генерируют:

Чтобы записать точную длину файла использовать функцию 30 BDOS, чтобы присвоить байту S1 значение числа **неиспользованных** байтов в последней записи файла.

Ни одна другая программа не устанавливает байт S1 - он всегда равен нулю. Нулевой байт S1 должен означать, что вся запись содержит достоверные данные.

Точная длина файла может быть вычислена как:

(число записей \* 128) – неиспользованные байты в последней записи