Ввод-вывод: диски, таймер

Лекция

Ревизия: 0.2

Последняя модификация: Vlad Kovtun

Дата последней модификации: 03.04.2015 14:43:00

История изменений

29.03.2010 – Версия 0.1. Первичный документ. Ковтун В.Ю.

03.08.2014 – Версия 0.2. Внесены изменения в разделы Магнитные диски, RAID. Ковтун В.Ю.

Содержание

Лекция 6. Ввод-вывод: диски, таймеры. Часть 2

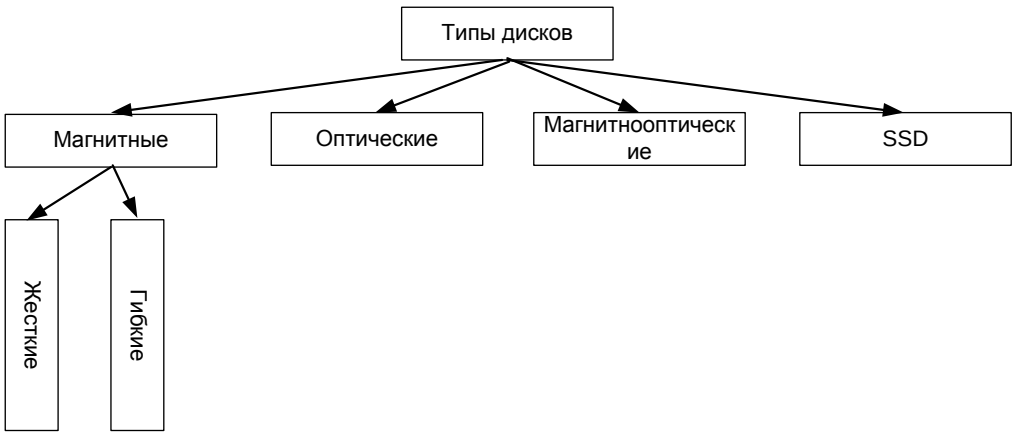
Вопросы

1. Магнитные диски.
2. Оптические диски.
3. Таймеры.

Диски

Аппаратная часть дисковых устройств хранения

Существует множество типов дисков (в зависимости от технологии хранения информации):



Их особенностью является (приблизительно) одинаковая скорость чтения и записи, которая может изменяться от положения сектора на диске, это делает их идеальными в качестве дополнительной памяти (страничная подкачка файлов, файловые системы и т. д.).

Для распространения программ, данных и фильмов используются различные виды оптических дисков: CD-ROM, CD-R, CD-RW, DVD. Следующим поколением является HD DVD и конкурентная технология Blue Ray.

Магнитные диски

При работе с магнитными дисками используются следующие понятия:

Головка (Head) - электромагнит, скользящий над поверхностью диска, для каждой поверхности используется своя головка. Нумерация начинается с 0.

Дорожка (Track) - концентрическая окружность, которое может прочитать головка в одной позиции. Нумерация дорожек начинается с внешней (первая имеет номер - 0).

Цилиндр (Cylinder) - совокупность всех дорожек с одинаковым номером на всех дисках, т.к. дисков может быть много и на каждом диске запись может быть с двух сторон.

Маркер - от него начинается нумерация дорожек, есть на каждом диске.

Сектор - на сектора разбивается каждая дорожка, сектор содержит минимальный блок информации. Нумерация секторов начинается от маркера.

Геометрия жесткого диска - набор параметров диска, количество головок, количество цилиндров и количество секторов.

Большинство магнитных дисков являются достаточно простыми устройствами, они обеспечивают на выходе поток битов (**RAW** – необработанный поток данных). На других типах дисков, в частности на **IDE-дисках** (IDE, Integrated Drive Electronics — встроенный интерфейс накопителей), **SCSI-дисках** (SCSI-Small Computer Systems Interface) либо же **SATA-дисках** (SATA, Serial ATA), само устройство содержит микроконтроллер, выполняющий значительный объем работ, и позволяющий собственно контроллеру обращаться к нему с набором команд высокого уровня.

IDE (SCSI, SATA)-диски обладают одним важным свойством для драйверов: **контроллер способен выполнять одновременно поиск дорожки на двух и более дисках**. Это свойство известно под названием **перекрывающегося поиска**. В то время как контроллер и ПО ожидают окончания операции поиска на одном устройстве, контроллер может инициировать поиск на другом устройстве. Многие контроллеры HDD также могут совмещать операцию чтения или записи на одном диске с поиском на другом или даже нескольких дисках. Однако контроллеры гибких дисков не могут одновременно читать и писать на двух дисководах. В случае HDD с их встроенными микроконтроллерами ситуация радикально меняется, поэтому несколько HDD могут одновременно работать в одной системе, по крайней мере переносить данные между диском и буфером контроллера. Однако между контроллером и ОЗУ в каждый момент времени может происходить только одна операция по переносу данных. Способность одновременного выполнения двух или более дисковых операций может существенно снизить среднее время доступа.

Глядя на спецификации современных HDD, следует помнить, что указанная геометрия, используемая ПО драйвера, может отличаться от физического формата. На старых HDD число секторов на дорожке было одинаково на всех цилиндрах. Современные HDD разделены на зоны с большим числом секторов на внешних дорожках и меньшим на внутренних. На Рис. 1(а) изображен крошечный HDD с двумя зонами. На внешней зоне каждая дорожка состоит из 32 секторов, тогда как на внутренней зоне по 16 секторов на дорожке. Реальные HDD, часто имеют по 16 зон с числом секторов, увеличивающимся примерно на 4 % в каждой следующей зоне при движении от центра диска к краю.

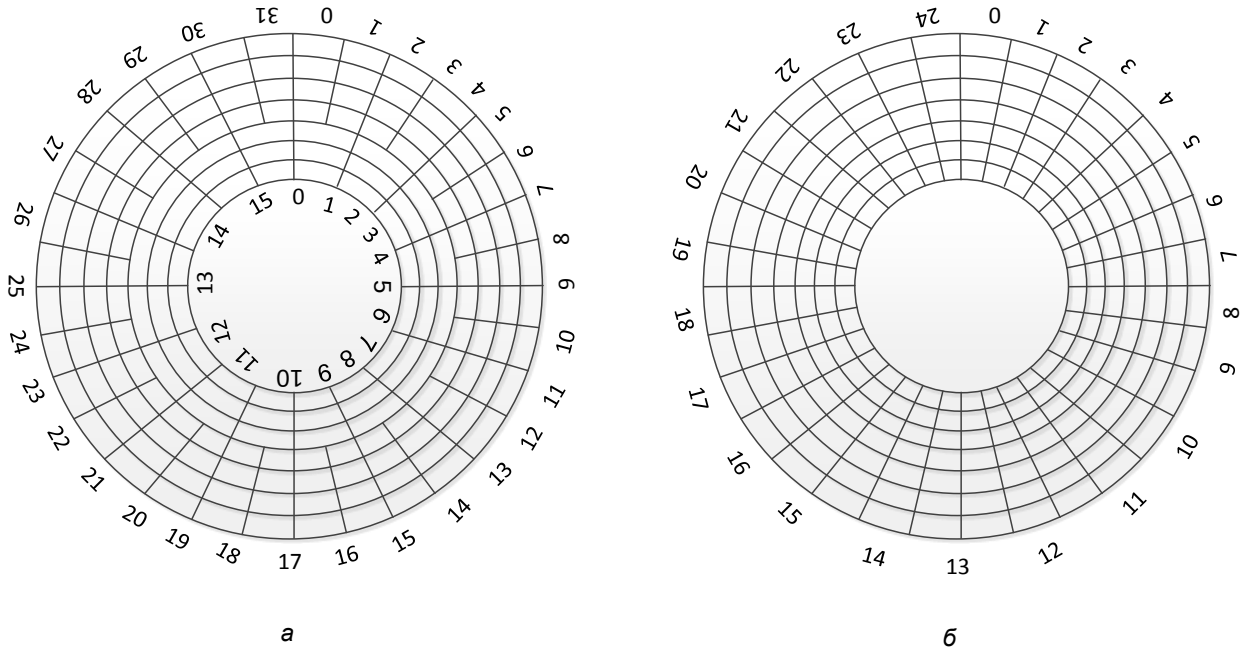


Рис. 1. Физическая геометрия диска с двумя зонами (а); возможная виртуальная геометрия этого диска (б)

Чтобы не усложнять жизнь ОС излишними подробностями, **современные HDD скрывают истинную геометрию** и предоставляют в качестве интерфейса виртуальную геометрию с одинаковым числом секторов на всех цилиндрах. Встроенный микроконтроллер HDD преобразует обращение к:

- виртуальному цилиндру,
- виртуальной головке,
- виртуальному сектору,

в физические соответствующие параметры. Возможная физическая геометрия HDD, изображенного на Рис. 1(а), виртуальная геометрия показана на Рис. 1(б). В обоих случаях диск имеет 192 сектора.

RAID

RAID массивы позволяют с помощью нескольких винчестеров создать систему хранения данных, которая будет обладать нужной степенью отказоустойчивости. Например, в случае массива RAID-5 ваши данные останутся в целости при сгорании одного из винчестеров, RAID-6 позволяет обеспечить гарантированное сохранение данных при вылете уже двух винчестеров.

Существуют три основных возможности:

- аппаратный RAID-массив,
- аппаратно-программный RAID-массив,
- программный RAID-массив.

Первые два способа требуют наличия достаточно дорогих RAID-контроллеров, не всегда это возможно.

Еще одно свойство всех RAID-систем состоит в том, что данные распределяются по HDD, а это позволяет распараллеливать операции. Существует несколько различных схем использования RAID-систем, получивших известность как **уровни от нулевого до пятого**. Помимо них существует еще несколько второстепенных уровней. Употребление термина «уровень» не совсем правильно в этом случае, так как данные схемы использования системы RAID не образуют иерархии. **Есть просто шесть различных вариантов организации совместной работы HDD.**

Основные шесть уровней RAID:

RAID 0 (Рис. 2(а)) - **чередующий набор**, соединение нескольких дисков в один большой логический диск, но логический диск разбит так, что запись и чтение происходит сразу с несколько дисков. Например, записываем блок 1, 2, 3, 4, 5, каждый блок будет записываться на свой диск.

Преимущества:

- удобство одного диска;
- увеличивает скорость записи и чтения.

Недостатки:

- уменьшает надежность (в случае выхода одного диска, массив будет разрушен), избыточность не предусмотрена.

RAID 1 (Рис. 2(б)) - **зеркальный набор**, параллельная запись и чтение на несколько дисков с дублированием (избыточность).

Преимущества:

- дублирование записей;
- увеличивает скорость чтения (но не записи).

Недостатки:

- требует в два раза больше дисковых накопителей.

RAID 2 (Рис. 2(в)) - работает на уровне слов и даже байт. Например, берется полбайта (4 бита) и прибавляется 3 бита четности (1, 2, 4 - рассчитанные по Хэммингу), образуется 7-битовое слово. В случае семи дисков слово записывается побитно на каждый диск. Так как слово пишется сразу на все диски, они должны быть синхронизированы.

Преимущества:

- надежность;
- увеличивает скорость записи и чтения (при потоке, но при отдельных запросах не увеличивает).

Недостатки:

- нужна синхронизация дисков.

RAID 3 (Рис. 2(г)) - упрощенная версия RAID 2, для каждого слова считается только один бит четности.

Преимущества:

- надежность;

- увеличивает скорость записи и чтения (при потоке, но при отдельных запросах не увеличивает).

Недостатки:

- нужна синхронизация дисков.

RAID 4 (Рис. 2(д)) - аналогичен уровню RAID 0, но с добавлением диска чётности. Если любой из дисков выйдет из строя, его можно восстановить с помощью диска чётности.

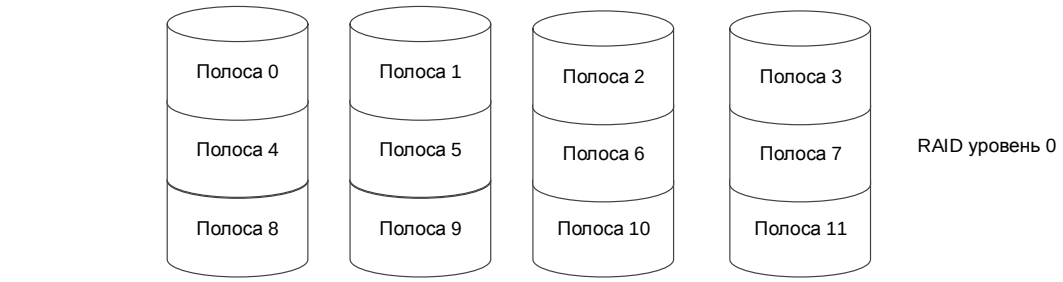
Преимущества:

- надёжность;
- не нужна синхронизация дисков.

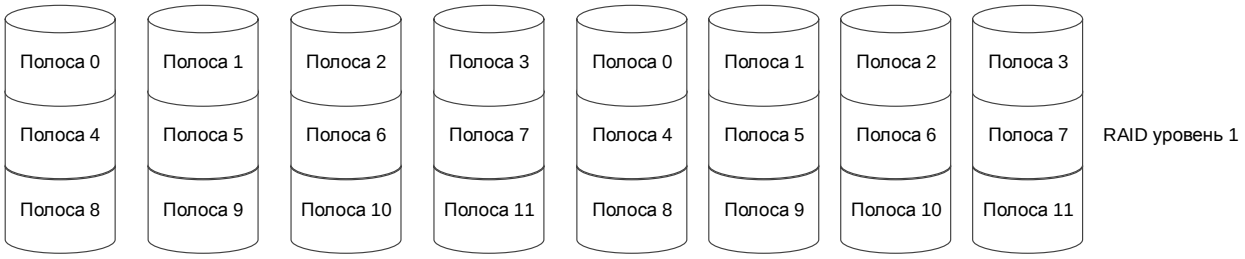
Недостатки:

- не даёт увеличения производительности, узким местом становится диск четности при постоянных пересчётах контрольных сумм.

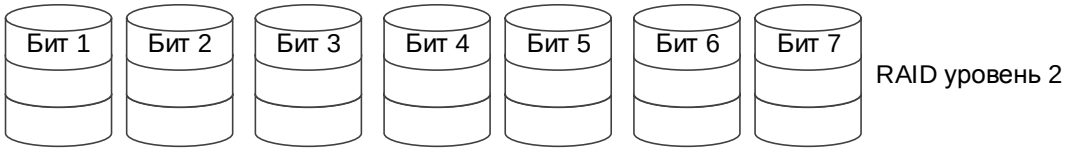
RAID 5 (Рис. 2(е)) - аналогичен уровню RAID 4, но биты четности равномерно распределены по дискам.



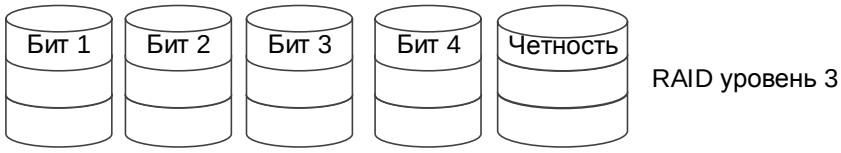
а



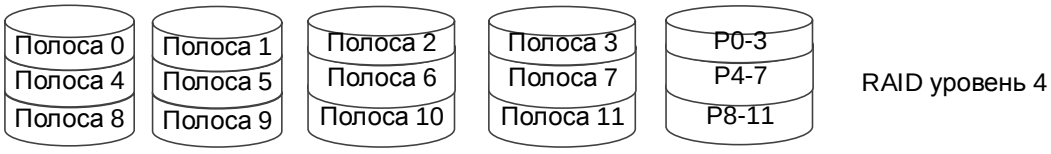
б



в



г



д

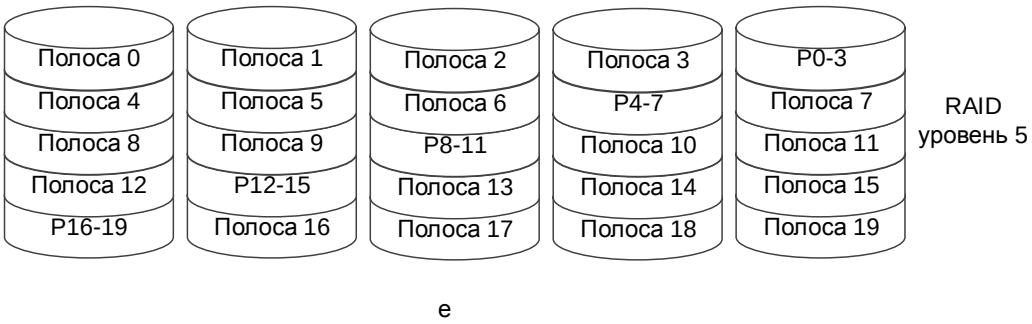


Рис. 2. Системы RAID уровней от 0 до 5. Резервные диски затенены

Форматирование дисков

Жесткий диск состоит из стопки сделанных из алюминия, стекла или других материалов пластин диаметра 5,25 дюйма или 3,5 дюйма (или даже меньших для ноутбуков). На каждую пластину нанесен тонкий магнитный слой оксида металла.

Прежде чем HDD можно будет пользоваться, каждой пластине нужно программно придать низкоуровневый формат. Этот формат состоит из серий концентрических дорожек, содержащих определённое число секторов, с короткими промежутками между секторами, Рис. 3.

Рис. 3. Сектор диска

Заголовок начинается с определенной последовательности битов, для распознавание начала сектора аппаратурой. Он также содержит: номера цилиндра и сектора и еще некоторую информацию. **Размер порции данных** определяется программой низкоуровневого форматирования. В большинстве дисков используются 512-байтовые секторы. **Поле ECC** (Error Correction Code) содержит избыточную информацию, позволяющую обнаруживать и исправлять ошибки чтения. **Размер и содержимое** этого поля зависят от производителя диска и желания увеличить емкость диска за счет снижения надежности или наоборот, а также способности контроллера поддерживать тот или иной алгоритм ECC. Не является редкостью, например, 16-байтовое поле ECC. Кроме того, все HDD обладают некоторым количеством запасных секторов, используемых для замены секторов с производственными дефектами.

При низкоуровневом форматировании 0-й сектор располагается на каждой следующей дорожке со сдвигом относительно предыдущей дорожки. Это смещение, называемое **перекосом цилиндров**, служит увеличению производительности диска. Замысел состоит в том, чтобы HDD мог читать несколько дорожек за одну операцию без потери времени на ожидание. Суть проблемы можно проиллюстрировать на Рис. 1(а). Предположим, необходимо прочитать 18 секторов, начиная с сектора 0 внутренней дорожки. Чтение 16 секторов (полной дорожки) занимает один оборот диска, но для перехода на следующую дорожку требуется время. К тому моменту, когда считывающая головка будет установлена на нужное место, 17-й сектор (сектор 0 следующей дорожки) уже успеет провернуться под головкой, поэтому, чтобы его прочитать, придется ждать целый оборот диска. Эта проблема решается при помощи разметки секторов со смещением, как показано на Рис. 4.

Величина перекоса цилиндров зависит от геометрии диска. Например, для диска, вращающегося со скоростью 10 000 об/мин, время одного оборота составляет 6 мс. Если на дорожке содержится 300 секторов, сектор проходит под головкой каждые 20 мкс. За время перемещения головки на соседнюю дорожку, равное 800 мкс, под головкой пройдут 40 секторов. Таким образом, перекос цилиндров должен составлять 40 секторов, а не три сектора, как показано на Рис. 4. Следует также заметить, что переключение с одной головки на другую тоже занимает время, поэтому **перекос головок** присутствует, но он не так велик, как перекос цилиндров.

В результате низкоуровневого форматирования емкость диска немного уменьшается в связи с расходами на заголовки сектора, межсекторные промежутки и поля ECC, а также резервные секторы. Часто полезный объем HDD составляет около 80% от общего

объема. Резервные секторы не входят в емкость отформатированного диска, поэтому у всех дисков одного типа при поставке абсолютно одинаковая емкость, независимо от числа дефектных секторов на дисках.

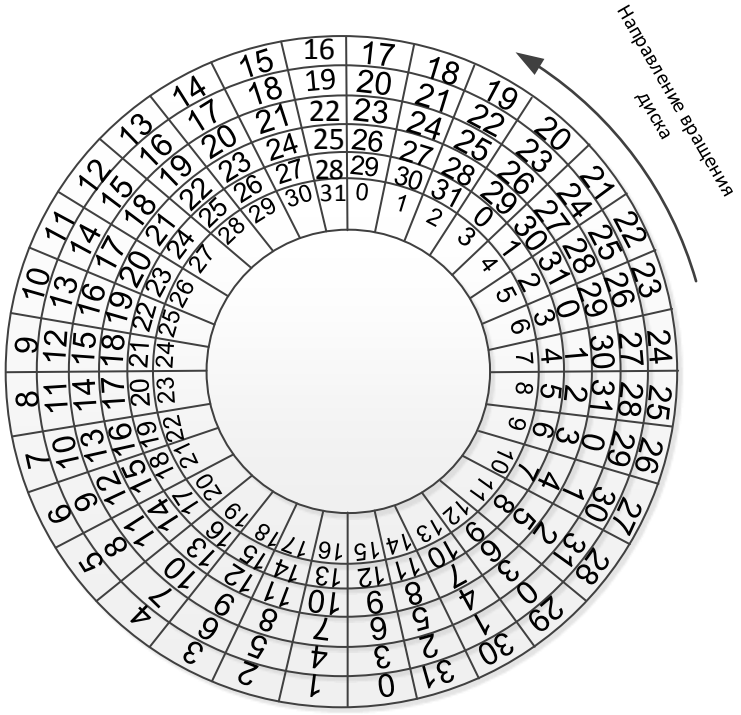


Рис. 4. Иллюстрация перекоса цилиндров

В вопросе емкости HDD существует определенная путаница, вызванная желанием производителей создать впечатление, что их диск больше, чем на самом деле. Поэтому иногда производителем может рекламироваться неформатированная емкость жесткого диска вместо форматированной. Например, возьмем диск с неформатированной емкостью $20x10^9$ байт. Он может быть продан за 20-гигабайтный диск. Однако после форматирования для данных окажутся доступными только $234=17,2x10^9$ байт. ОС обычно считает гигабайты как 2^{30} байт, а не 10^9 байт. В результате она сообщит, что емкость диска составляет 16,0 Гбайт, а не 17,2 Гбайт.

Форматирование также влияет на производительность диска. Если у диска, вращающегося со скоростью 10 000 об/мин, на каждой дорожке по 300 секторов размером по 512 байт, то вся дорожка, то есть 153 600 байт, считывается за 6 мс, что составляет скорость считывания данных 25 600 000 байт/с или 24,4 Мбайт/с. Быстрее считывать данные невозможно, независимо от используемого интерфейса, даже если это SCSI со скоростью передачи данных 80 Мбайт/с или 160 Мбайт/с.

Чтение с такой скоростью в течение длительного периода времени требует большого буфера в контроллере. Представьте контроллер с буфером размером в один сектор, которому дана команда прочитать два сектора подряд. Прочитав первый сектор с диска и проверив его контрольную сумму, контроллер должен передать данные в ОЗУ. Пока первый сектор передается в ОЗУ, второй сектор уже провернется под головкой, и контроллеру придется ждать почти полный оборот диска.

Эта проблема может быть решена, если при **форматировании нумеровать секторы не подряд**. На Рис. 5(а) показана обычная нумерация секторов. На Рис. 5(б) видим однократное чередование секторов, предоставляющее контроллеру возможность перевести дух и скопировать прочитанный сектор из буфера в ОЗУ. Это позволяет снизить скорость считывания всего в два раза, а не, например, в 300 раз, если на одной дорожке помещается 300 секторов.

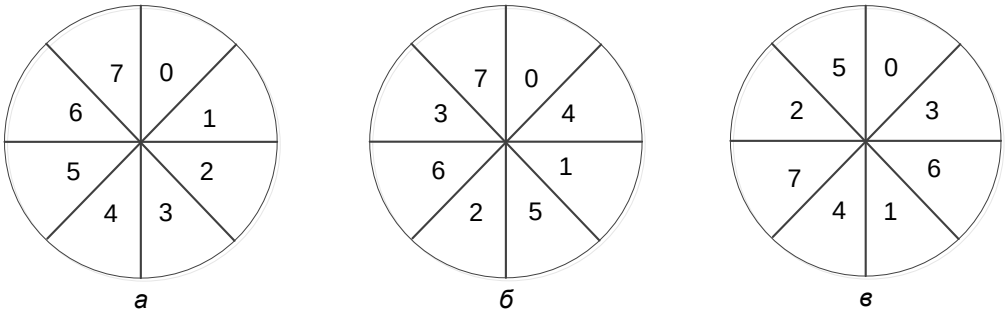


Рис. 5. Без чередования (а); однократное чередование (б); двукратное чередование (в)

Если процесс копирования очень медленный, может потребоваться двукратное чередование, показанное на Рис. 5(в). Если у контроллера буфер только на один сектор, то не важно, выполняется ли копирование в ОЗУ контроллером, CPU или микросхемой DMA. На это все равно нужно время. Чтобы избежать необходимости в чередовании секторов, контроллер должен иметь достаточно большой буфер, чтобы прочитать в него сразу целую дорожку. У многих современных контроллеров такой буфер есть.
После выполнения низкоуровневого форматирования диск разбивается на разделы. Логически каждый раздел диска воспринимается ОС как отдельный диск. На современных компьютерах в секторе 0 помещается **главная загрузочная запись - MBR**, содержащая часть загрузочной программы и таблицу разделов (Partition Table). Таблица разделов содержит номера начальных секторов и размеры каждого раздела. В таблице разделов есть место для четырех разделов. Если все они предназначены для системы Windows, они будут называться устройствами C:, D:, E: и F:. Если на трех из них размещается система Windows, а на четвертом UNIX, тогда Windows будет называть свои разделы C:, D: и E: (четвертый раздел не будет виден системе Windows). Первое устройство чтения CD будет называться F:. Чтобы компьютер мог загружаться с HDD, один из разделов должен быть помечен как активный.

Последним этапом подготовки HDD к употреблению заключается в высокоуровневом форматировании каждого раздела (по отдельности). **Эта операция помещает в раздел:**

- загрузочный блок;
- бит-карту или список свободных блоков устройства;
- корневой каталог;
- пустую ФС.

Кроме того, в таблицу разделов помещается определенный код, указывающий, какая ФС используется в данном разделе, поскольку многие ОС традиционно поддерживают несколько несовместимых ФС. Затем в одном из разделов может быть установлена ОС.

При включении питания компьютера запускается базовая система ввода-вывода BIOS, которая считывает MBR с диска и передает в него управление. Загрузочная программа определяет, который из разделов диска является активным. Из этого раздела считывается и запускается загрузочный сектор. Загрузочный сектор содержит маленькую программу, которая находит в корневом каталоге определенный файл (либо ОС, либо загрузчик больших размеров). Этот файл загружается в память и запускается.

Алгоритмы планирования перемещения головок

Определим, сколько времени занимает считывание или запись одного блока диска. Необходимое для этого время определяется тремя факторами:

1. Время поиска (время перемещения головки на нужный цилиндр).

2. Задержка вращения (время, требуемое для поворота нужного сектора под головку).

3. Время передачи данных.

Для большинства HDD первая составляющая (время поиска) существенно превосходит остальные две, поэтому значительного увеличения производительности системы можно добиться, снижая время поиска.

Если драйвер диска принимает и выполняет запросы по одному в порядке получения, то есть по принципу **«первым пришел — первым обслужен»** (FCFS, First Come, First

Served), тогда мало что можно сделать для оптимизации времени поиска. Однако при сильной загрузке HDD возможно применение другой стратегии. В этом случае высока вероятность поступления новых запросов от других процессов во время перемещения головки для обработки предыдущего запроса. **Многие дисковые драйверы содержат таблицу, индексированную по номерам цилиндров, в которой в единый связный список собираются все поступившие и ждущие обработки обращения к цилиндрам.**

С помощью подобной структуры данных можно создать более совершенный алгоритм планирования, чем простое обслуживание в порядке поступления запросов. Рассмотрим, например, диск с 40 цилиндрами. Поступает запрос на чтение блока с цилиндра 11. Во время перемещения головки на 11-й цилиндр поступают новые обращения к цилиндрам 1, 36, 16, 34, 9 и 12. Новые запросы помещаются в таблицу, состоящую из отдельных списков для каждого цилиндра. Поступившие запросы помечены крестиками на Рис. 7.

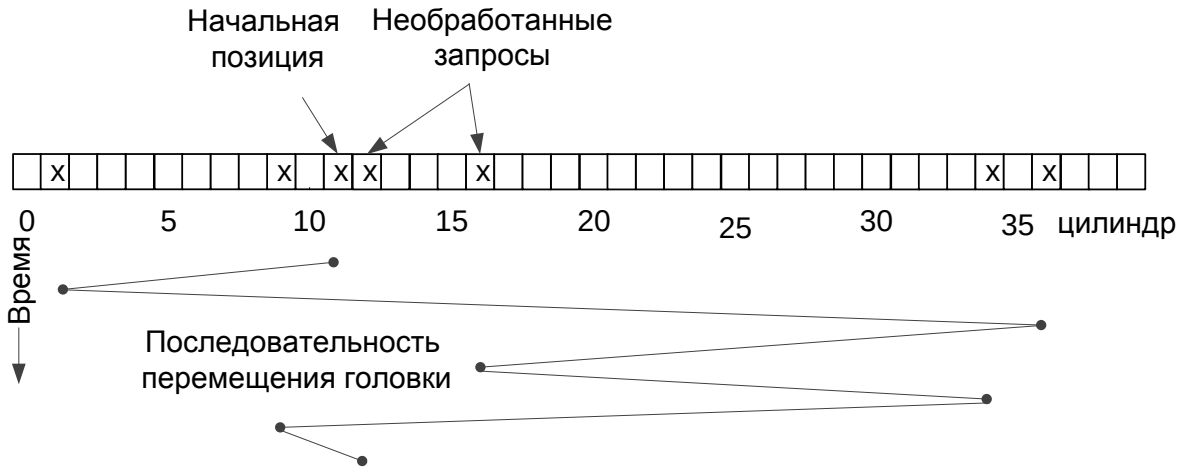


Рис. 6. Алгоритм «первым пришел — первым обслужен»

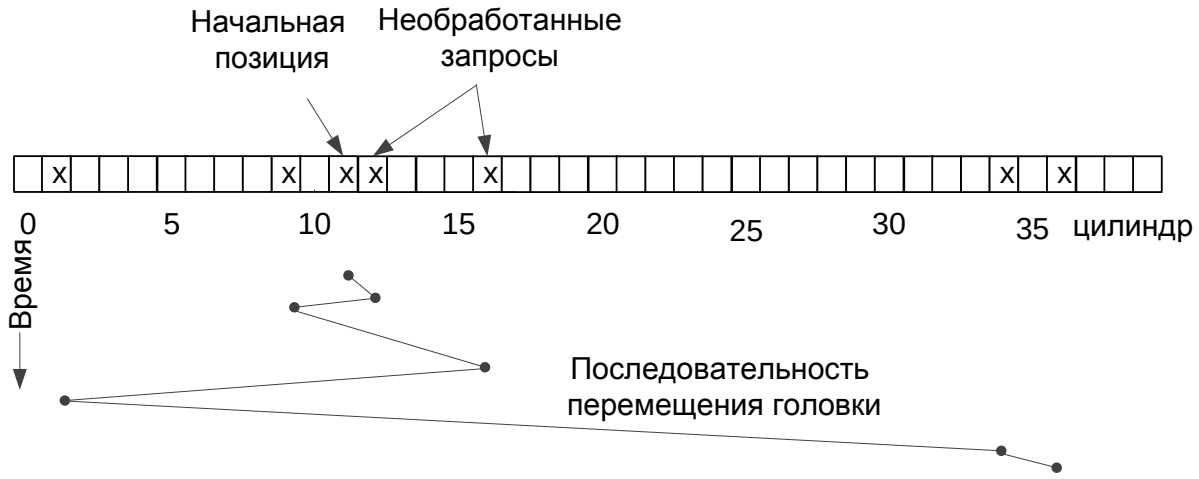


Рис. 7. Алгоритм планирования «ближайший цилиндр первым»

Выполнив первый запрос (обращение к цилиндру 11), драйвер диска должен выбрать следующий запрос. Если обслуживать запросы в порядке поступления, то он должен переместить головку на цилиндр 1, затем на цилиндр 36 и т. д. В результате выполнение этого алгоритма потребует перемещения блока головок на 10, 35, 20, 18, 25 и 3 цилиндра, что в сумме составит 111 цилиндров.

Время перемещения головки можно уменьшить, выбирая каждый раз ближайший цилиндр. При той же серии запросов, показанной на Рис. 7, последовательность их обработки выглядит как ломаная кривая внизу рисунка. При такой последовательности выполнения запросов потребуются перемещения блока головок на 1, 3, 7, 15, 33 и 2

цилиндра, что в сумме составит 61 цилиндр. Применение данного алгоритма, называющегося **SSF (Shortest Seek First — «ближайший запрос первым»)**, снижает суммарные перемещения головок почти вдвое.
Недостатки: Предположим, во время обработки запросов, показанных на Рис. 7, поступили новые запросы. Например, после обработки обращения к цилиндру 16 поступает новый запрос к цилиндру 8. Новый запрос будет иметь приоритет над обращением к цилиндру 1. Затем, если поступит запрос к цилиндру 13, то головка опять будет перемещаться к цилиндру 13 для обработки нового запроса, а запрос к цилиндру 1 останется необработанным. При сильной загрузке диска головка диска будет большую часть времени находиться где-то в районе средних цилиндров, а запросам к крайним цилиндрам диска придется ждать, пока нагрузка диска случайно не снизится и обращений к середине диска не станет меньше. В результате качество обслуживания запросов к цилиндрам, сильно удаленным от середины, может оказаться низким. То есть в конфликт вступают принцип минимизации времени отклика и принцип справедливости.

В высотных зданиях также приходится иметь дело с данной проблемой планирования обслуживания запросов лифта. Вызовы лифта постоянно поступают с разных этажей. Компьютер, управляющий лифтом, должен следить за последовательностью поступления запросов и обслуживать их либо в порядке подачи заявок, либо обслуживая первым ближайший запрос.

Однако в большинстве лифтов применяются различные алгоритмы, пытающиеся примирить конфликтующие цели эффективности и справедливости. Обычно лифт продолжает двигаться в одном направлении до тех пор, пока на этом направлении более не остается запросов. Затем лифт меняет направление движения. Этот алгоритм, называющийся **элеваторным алгоритмом**, требует от ПО следить всего за одним битом, хранящим информацию о текущем направлении движения, ВВЕРХ или ВНИЗ. Выполнив очередной запрос, диск или лифт проверяет значение бита. Если в нем содержится значение ВВЕРХ, кабина лифта или блок головок перемещается вверх к следующему запросу. Если в этом направлении запросов больше нет, состояние бита инвертируется.

Рис. 8 иллюстрирует работу элеваторного алгоритма с теми же семью запросами, что были использованы в качестве примера на Рис. 7. Предполагается, что изначально бит направления движения указывал ВВЕРХ. При этом цилиндры обслуживаются в порядке 12, 16,34,36, 9 и 1, что соответствует перемещению блока головок на 1, 4, 18, 2, 27 и 8 цилиндров и в сумме составляет 60 цилиндров. В данном случае элеваторный алгоритм оказался даже слегка лучше, чем алгоритм SSF, однако на практике он обычно несколько хуже. **Достоинство элеваторного алгоритма** состоит в том, что при любом заданном наборе запросов верхняя граница необходимых перемещений блока головок для выполнения всех запросов фиксирована и ограничена двойным числом цилиндров.

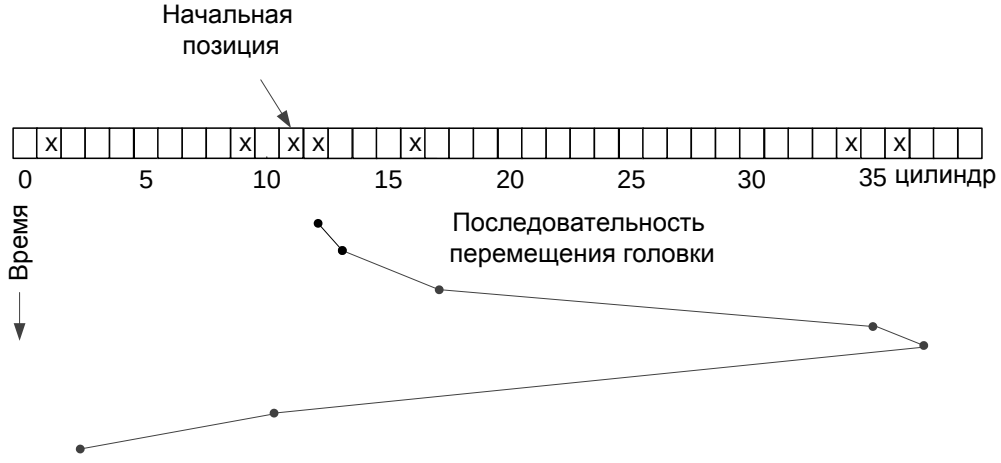


Рис. 8. Элеваторный алгоритм планирования обращений к диску

Незначительная модификация этого алгоритма с меньшим разбросом времени отклика состоит в том, чтобы сканировать цилиндры всегда в одном направлении. После обслуживания обращения к цилиндру с самым высоким номером блок головок перемещается к самому нижнему запрашиваемому цилиндру, после чего продолжает движение вверх. Таким образом, самый нижний цилиндр как бы считается расположенным выше самого верхнего.

Некоторые контроллеры дисков предоставляют ПО способ узнавать номер текущего сектора под головкой. Такие контроллеры позволяют использовать дополнительный метод оптимизации. Если есть два или более запросов к одному и тому же цилиндру, драйвер может выбрать из них тот, сектор которого пройдет под головкой первым.

Если у диска время перемещения головок значительно ниже задержки, связанной со скоростью вращения диска, тогда следует применять **другую стратегию оптимизации**. Запросы, ожидающие обработки, должны сортироваться по номерам секторов, и как только следующий сектор приблизится к головке, блок головок должен быстро переместиться на нужную дорожку, чтобы считать или записать его.

У современных HDD производительность настолько сильно определяется задержками перемещения головок и вращения диска, что читать по одному или по два сектора крайне неэффективно. Но этой причине многие современные контроллеры дисков читают и сохраняют в кэше по нескольку секторов сразу, даже если запрашивается всего один сектор. Обычно при этом считывается запрашиваемый сектор и все остальные секторы этой дорожки, при условии, что для них достаточно места в кэше контроллера. Например, у диска размер кэша обычно составляет 32-128 Мбайт. Режим использования кэша динамически определяется контроллером. В простейшем режиме кэш разделяется на две секции, одна для запросов чтения, другая для запросов записи. Если последующий запрос на чтение сектора может быть удовлетворен прямо из кэша контроллера, запрашиваемые данные могут быть выданы немедленно.

Следует отметить, что кэш контроллера диска абсолютно никак не связан с кэшем ОС. Кэш контроллера обычно содержит блоки, на которые запрос еще не поступал, но которые было удобно прочитать, так как они случайно оказались под головкой при чтении других блоков. Напротив, любой кэш ОС состоит из блоков, на чтение которых были явно направлены запросы и которые, по мнению ОС, могут понадобиться снова в ближайшем будущем (например, блок диска, содержащий каталог). При одновременном обслуживании контроллером нескольких устройств таблица запросов, ждущих обработки, должна поддерживаться отдельно для каждого устройства. Когда любое из устройств завершает выполнение текущего запроса, ему нужно дать команду на перемещение блока головок на цилиндр для выполнения следующего запроса (при условии, что контроллер позволяет работать в режиме перекрывающегося поиска). По завершении текущей операции переноса данных может быть выполнена проверка правильного позиционирования блоков головок всех устройств. Если хотя бы один блок установлен, может быть начат следующий перенос данных. В противном случае драйвер должен выдать новую команду поиска цилиндра устройству, только что завершившему операцию переноса данных, после чего перейти в состояние ожидания прерывания от диска, установившего блок головок на нужный цилиндр и готового к перемещению данных.

Важно понимать, что во всех вышеизложенных алгоритмах планирования работы дисков молчаливо подразумевается, что физическая геометрия дисков совпадает с виртуальной.

Обработка ошибок

Производственные дефекты проявляются в виде дефектных секторов, то есть секторов, которые не читаются правильно после записи. Если дефект сектора невелик, например всего несколько бит, такой сектор можно использовать, позволив полю ECC исправлять ошибки при каждом чтении сектора. При более серьезных дефектах ошибка уже не может быть скорректирована.

Дефектные блоки могут обрабатываться контроллером или ОС. При первом подходе каждый диск проверяется на фабрике, а список дефектных секторов записывается на диск. **Вместо каждого дефектного блока используется запасной**.

Подобная замена может выполняться двумя способами:

На Рис. 9(а) показана дорожка диска из 30 блоков и 2 запасных. Сектор 7 поврежден. **Контроллер может пометить один из запасных секторов** как сектор 7, как показано на Рис. 9(б).

Другой способ состоит в **сдвиге всех секторов на один сектор**, Рис. 9(в). В обоих случаях контроллер должен знать номера каждого сектора. Он должен хранить эту информацию либо в своих внутренних таблицах (по одной на дорожку), либо перезаписывая заголовки секторов. При перезаписи заголовков метод на Рис. 9(в) требует большей работы (так как для этого нужно перезаписать 23 заголовка), но в

конечном итоге он дает лучшую производительность, так как при этом вся дорожка все так же может быть прочитана за один оборот.

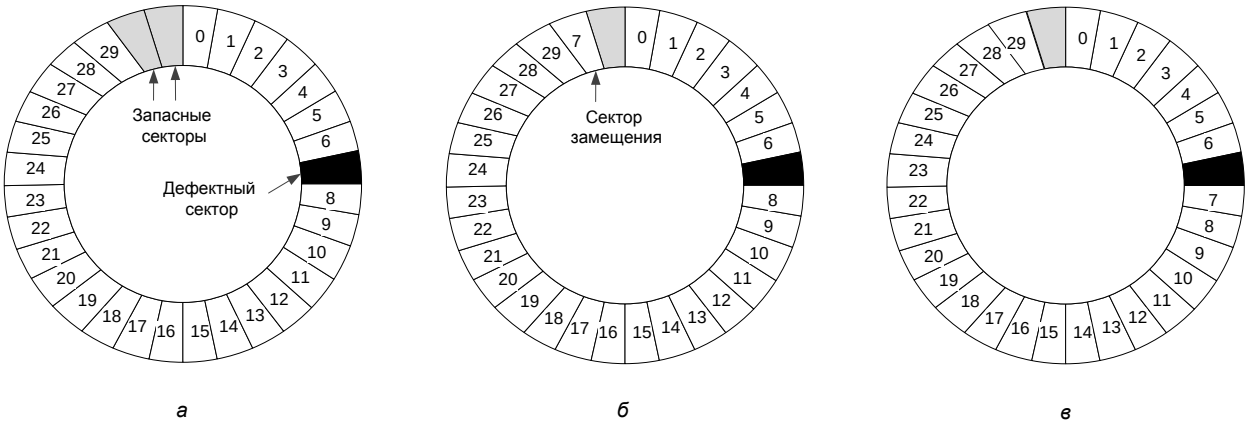


Рис. 9. Дорожка диска с дефектным сектором (а); замена дефектного сектора запасным (б); сдвиг всех секторов (в)

Ошибки могут возникать при выполнении обычных операций после установки диска. **Первая линия обороны при получении ошибки**, которую не может исправить ECC, состоит в простом повторении попытки чтения. Некоторые ошибки чтения вызываются попаданием пылинки на головку и исчезают при повторной попытке. Если контроллер замечает повторяющиеся ошибки на определенном секторе, **он может переключиться с него на запасной сектор раньше**, чем сектор с ошибками выйдет из строя окончательно. При этом данные не будут потеряны, а ОС и пользователь даже не заметят наличия проблемы. Для этого обычно применяется метод, показанный на Рис. 9(б), так как остальные секторы этой дорожки могут уже содержать данные. Использование метода, изображенного на Рис. 9(в), потребует перезаписи не только всех заголовков, но также и копирования всех данных.

Как говорилось ранее, есть два основных подхода к исправлению ошибок: обработка их контроллером или ОС. Если контроллер не может прозрачно преобразовывать секторы, тогда этим должна программно заниматься ОС. Если преобразованием занимается ОС, она должна гарантировать, что дефектные секторы не встречаются в файлах, а также в списке или битовом массиве свободных секторов. Один из способов добиться этого состоит в создании секретного файла, включающего в себя все дефектные секторы. Если этот файл не является частью файловой системы, пользователи не смогут его случайно прочитать или, что еще хуже, удалить.

Есть еще одна проблема, связанная с созданием резервных копий диска. Если архивация диска выполняется по файлам, то важно, чтобы программа архивации не пыталась копировать файл дефектных блоков. Для этого ОС должна спрятать этот файл настолько хорошо, чтобы даже архивирующая программа не смогла его найти. Если же диск архивируется посекторно, а не пофайлово, то будет трудно, если вообще возможно, избежать ошибок чтения во время архивации.

Также возникают **ошибки поиска цилиндра**, вызванные механическими проблемами блока головок. Контроллер следит за положением блока головок. При установке головки на заданный цилиндр он выдает серию импульсов двигателю блока головок, по одному импульсу на цилиндр. Когда блок головок устанавливается в требуемое положение, контроллер считывает истинное значение цилиндра из заголовка первого попавшегося сектора. Если блок головок оказывается не на той дорожке, на которой нужно, возникает ошибка поиска.

Большинство контроллеров HDD **автоматически исправляют ошибки поиска**, но большинство контроллеров FDD просто выставляют бит ошибки и оставляют все остальное драйверу. Драйвер обрабатывает ошибку, издавая команду recalibrate. При этом блок головок отодвигается на самую внешнюю дорожку диска до упора. Это положение принимается контроллером за нулевую дорожку, таким образом, контроллер снова начинает понимать, где находится блок головок. Обычно это решает проблему. Если же нет, то либо нужно сменить диск, либо починить дисковод.

Стабильное запоминающее устройство

Для некоторых данных крайне важно, чтобы данные никогда не были потеряны или испорчены, даже по причине ошибок диска или CPU. В идеале диск должен просто всегда работать без ошибок. К сожалению, это недостижимо. Однако можно создать дисковую подсистему, обладающую следующими свойствами: получив команду записи, такое устройство либо корректно записывает данные, либо не делает ничего, не изменяя хранящиеся на нем данные. Подобная система называется **стабильным запоминающим устройством** и реализуется программно. Ниже будет описана разновидность оригинальной идеи.

Прежде чем создать алгоритм, важно иметь ясную модель всех возможных ошибок. **Модель предполагает**:

1. При записи блока (одного или нескольких секторов) эта операция записи либо корректна, либо некорректна, и эта ошибка может быть обнаружена с помощью последующего чтения и изучения полей ECC. Гарантированное обнаружение ошибки невозможно в принципе, так как 512-байтовый сектор, способный хранить 2^{4096} различных комбинаций данных, защищается 16-байтовым полем ECC, способным принимать всего 2^{128} значений, из которых допустимыми являются далеко не все. Таким образом, существуют комбинации (из которых всего одна верная), соответствующих одному и тому же значению поля ECC. Вероятность случайного совпадения значения 16-байтового поля ECC составляет около 2^{-128}, то есть настолько мало, что можем называть это число нулем, хотя в действительности это не так.

2. Правильно записанный сектор может внезапно стать дефектным и перестать читаться. Однако предполагается, что такие события происходят столь редко, что вероятность выхода из строя одинаковых секторов на основном и резервном дисках за небольшой интервал времени (например, один день) можно смело считать равной нулю.

3. Модель также допускает выход из строя CPU. В этом случае он просто останавливается. Любая операция записи, совершающаяся в этот момент, также останавливается, это приводит к неверным данным в одном секторе и неверному значению поля ECC, что может быть обнаружено позднее. При всех вышеперечисленных условиях возможно создать 100% надежное стабильное запоминающее устройство, то есть либо записывающее данные корректно, либо оставляющее все хранящиеся данные так, как есть.

Стабильное запоминающее устройство использует пару идентичных дисков, в которых соответствующие блоки работают совместно, образуя блоки, защищенные от ошибок. При отсутствии ошибок соответствующие блоки на обоих дисках идентичны. Для получения одинакового результата может быть прочитан любой из дисков. Для достижения этой цели определены три следующие операции:

1. Стабильная операция записи.

- Сначала на диск 1 записывается блок данных, который затем считывается для проверки корректности выполнения этой операции.
- Если обнаруживается ошибка, операции записи и чтения повторяются в цикле до тех пор, пока при чтении блока не будет ошибки или пока этот цикл не будет выполнен определенное число раз. Если после выполнения заданного числа циклов операций записи и чтения успешного результата добиться не удается, блок диска помечается как дефектный и вместо него используется резервный блок. После этого вся процедура повторяется до тех пор, пока блок не будет записан, независимо от того, сколько резервных блоков придется задействовать.
- Когда записан блок на диск 1 с использованием той же процедуры, записывается и проверяется соответствующий ему блок диска 2.

2. Стабильная операция чтения.

- Считывается блок с диска 1.
- Если проверка значения поля ECC не дает верного результата, считывание блока и проверка контрольной суммы повторяются в цикле определенное число раз.
- Если все попытки чтения оказываются неуспешными, соответствующий блок читается с диска 2.

Поскольку операция стабильной записи создает две проверенные копии блока, а также предполагается, что вероятность внезапного выхода из строя сразу двух соответствующих блоков за короткий интервал времени пренебрежимо мала, операция стабильного чтения всегда завершается успешно.

3. Восстановление от сбоев.

- После сбоя программа восстановления сканирует оба диска и сравнивает соответствующие блоки.
- Если оба блока успешно читаются и совпадают, то никаких действий не выполняется. Если у одного из блоков при чтении обнаруживается ошибка контрольной суммы (ECC), то на место дефектного блока записывается правильный блок.
- Если оба блока читаются без ошибки, но не совпадают, тогда блок с диска 1 пишется поверх блока на диске 2.

При отсутствии сбоев CPU эта схема всегда работает, так как при операции стабильной записи блок всегда записывается дважды, а также предполагается, что сразу два соответствующих блока никогда не выходят из строя одновременно. Как может повлиять сбой CPU на операцию стабильной записи? Результат зависит от того момента, когда произойдет сбой. Возможны пять вариантов, показанные на Рис. 10.

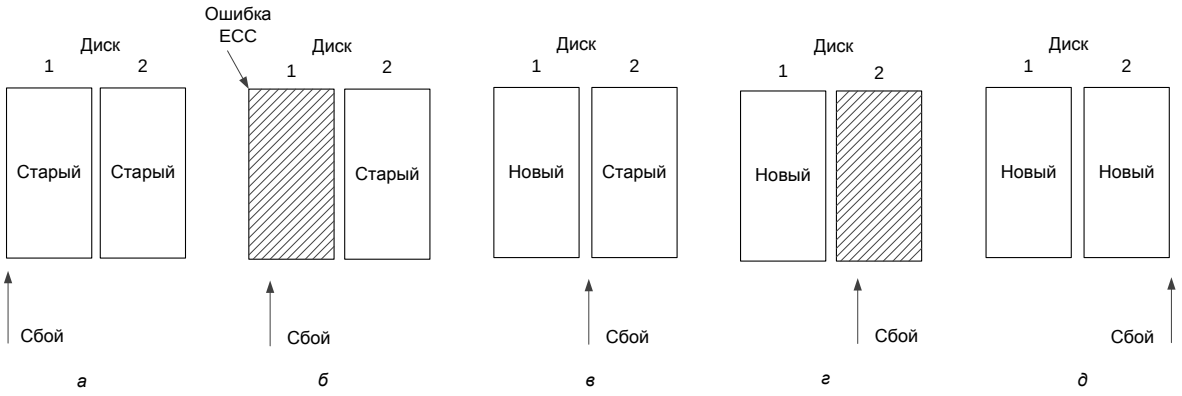


Рис. 10. Анализ влияния сбоя процессора на стабильность записи

На Рис. 10(а) сбой CPU происходит прежде, чем записывается какая-либо копия блока. При восстановлении ничего не меняется и сохраняется старое значение, что является допустимым.

На Рис. 10(б) сбой CPU происходит во время записи блока на диск 1, в результате чего разрушается содержимое этого блока. ПО восстановления обнаруживает эту ошибку и восстанавливает блок на диске 1 с диска 2. Таким образом, результат сбоя устраняется и восстанавливается старое значение.

На Рис. 10(в) сбой CPU происходит после записи на диск 1, но до записи на диск 2. ПО восстановления копирует блок с диска 1 на диск 2. Операция записи считается успешной.

Ситуация на Рис. 10(г) похожа на ситуацию на Рис. 10(б). При восстановлении дефектный блок заменяется правильным блоком. Окончательным значением обоих блоков будет новое значение.

Наконец, на Рис. 10(д), как и в ситуации на Рис. 10(а), ПО восстановления видит, что оба блока успешно читаются и совпадают. Поэтому тут также ничего не изменяется.

К данной схеме применимы различные методы оптимизации и усовершенствования. Прежде всего, сравнение всех блоков, конечно, выполнимо, но все же занимает слишком много времени. Значительно ускорить этот процесс можно, записывая номера записываемых блоков перед операцией стабильной записи. В результате после сбоя нужно будет проверить состояние только этих блоков. Это может быть реализовано несколькими различными способами. На некоторых компьютерах имеется небольшое количество энергонезависимого ОЗУ, представляющего собой специальную CMOS-память (Complementary Metal-Oxide Semiconductor — комплементарный металло-оксидный полупроводник), питающуюся от отдельной литиевой батареи. В отличие от ОЗУ, содержимое которой теряется после сбоя, содержимое энергонезависимого ОЗУ сохраняется. В энергонезависимом ОЗУ обычно хранится время и дата (изменяющиеся специальной микросхемой). В результате в компьютерах часы идут даже тогда, когда питание компьютера выключено.

Оптические диски

CD

В 1980 году фирма Philips совместно с Sony разработала компакт-диск (CD, Compact Disc), который быстро вытеснил вращающиеся со скоростью 33 1/3 оборотов в минуту виниловые пластинки. Точные технические детали компакт-диска были опубликованы в виде Международного стандарта ISO 10149 - Красной книгой (из-за цвета обложки). Каждому международному стандарту обычно соответствует какой-либо национальный стандарт, как, например, ANSI (American National Standards Institute — Национальный институт стандартизации США) или DIN (Deutsche Industrie Norm — промышленная норма Германии), применяемые в Европе.

Для производства компакт-дисков используется мощный инфракрасный лазер, которым прожигаются отверстия диаметром 0,8 мкм в металлическом покрытии стеклянного диска-оригинала, называемого также **мастер-диском**. С мастер-диска снимается слепок с выпуклостями на месте прожженных лазером отверстий. С помощью этого слепка из поликарбонатной смолы печатаются компакт-диски с тем же рисунком выемок, что и на стеклянном оригинале. Затем застывший поликарбонат покрывается очень тонким слоем отражающего свет алюминия, поверх которого диск покрывается защитным лаком, а на него затем наносится этикетка. Углубления в поликарбонате называют **питами** (pit — впадина), а нетронутые лазером участки между питами называют **«землей» или «сушей»** (land).

При воспроизведении полупроводниковый лазер низкой мощности освещает питы и участки между ними лучом с длиной волны 0.78 мкм, в то время как они прокручиваются с постоянной линейной скоростью. Лазер освещает алюминиевое покрытие диска сквозь прозрачный поликарбонатный диск толщиной в 1,2 мм. Питы выглядят с этой стороны как выступы высотой в четверть длины волны луча лазера. В результате интерференции свет, отраженный от питов, имеет меньшую яркость, чем отраженный от участков между ними, что и регистрируется фотодетектором как последовательность нулей и единиц. На самом деле для большей надежности за 1 считается переход пит/земля или земля/пит, а отсутствие перехода означает 0.

Последовательность питов и промежутков между ними записывается в виде единой непрерывной спиральной дорожки, начинающейся недалеко от центра диска и заканчивающейся у края диска. Максимальная ширина спирали может составлять 32 мм, а число оборотов — 22 188 (около 600 на мм). Длина спирали, показанной на Рис. 11, достигает 5,6 км в длину.

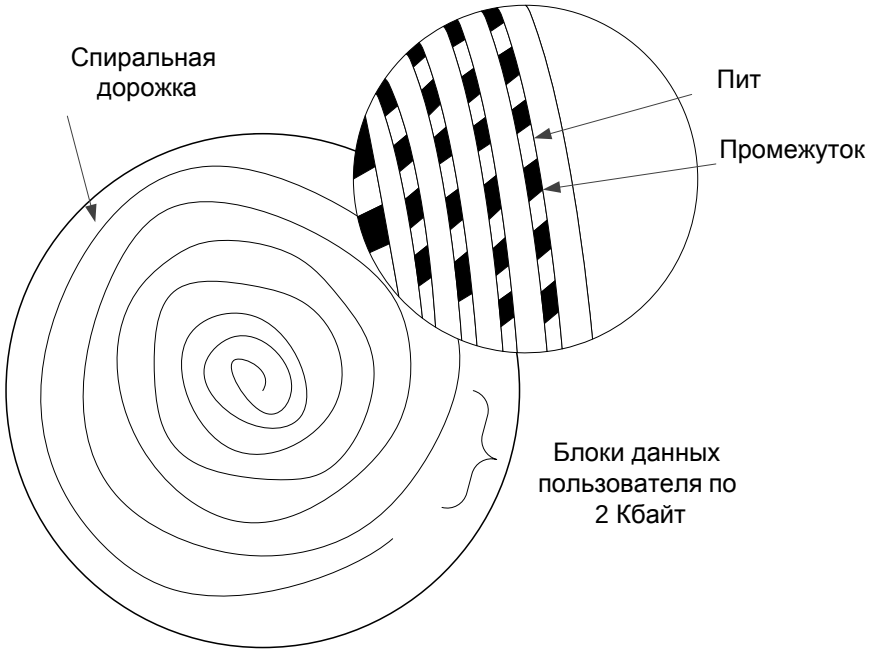


Рис. 11. Структура записи компакт диска

В 1984 году фирмы Philips и Sony осознали потенциал использования CD для хранения данных, поэтому они опубликовали Желтую книгу, в которой определили точный

стандарт того, что теперь называется **CD-ROM** (Compact Disk Read Only Memory). Поскольку рынок аудио CD к тому моменту был уже весьма широк, было решено сделать CD-ROM точно того же размера, что и аудио CD, а также механически и оптически совместимыми с ними. Помимо возможности проигрывать музыку на устройствах чтения CD-ROM, это также давало возможность использовать для производства CD-ROM те же технологические линии, на которых производились аудио CD. В результате себестоимость производства одного CD-ROM оказалась намного ниже одного доллара. **Недостатком такого решения явилась необходимость использования в устройствах чтения CD-ROM медленных двигателей, вращающихся с переменной скоростью**.

В Желтой книге был определен формат компьютерных данных. Новый стандарт также улучшал способность системы исправлять ошибки, что было очень важно, так как если на слух почти не заметно искажение одного бита то тут, то там, для хранения информации требуется значительно более высокая надежность. **Основой формата CD-ROM является кодирование одного байта 14-разрядным числом**. Как уже было сказано, 14 разрядов достаточно для кодирования одного байта (8 бит) кодом Хэмминга с запасом в два бита. В действительности используется более мощная система помехоустойчивого кодирования. При чтении преобразование 14-в-8 осуществляется аппаратно при помощи таблицы.

На более высоком уровне группа из 42 последовательных 14-разрядных символов образуют 588-разрядный кадр. Каждый кадр содержит 192 бит данных (24 байт). Оставшиеся 396 бит используются для коррекции ошибок и управления. До сих пор используемая схема одинакова для звуковых компакт-дисков и CD-ROM.

Желтая книга добавляет к этому стандарту группирование 98 кадров в так называемый CD-ROM-сектор, как показано на Рис. 12. Каждый CD-ROM-сектор начинается 16-байтовым заголовком, первые 12 байт которого содержат OOFFFFFFFFFFFFFFFFFFFFOO (шестнадцатеричное), чтобы считывающее устройство могло распознать начало CD-ROM-сектора. Следующие три байта содержат номер сектора, что необходимо, так как поиск данных на единственной спирали диска значительно сложнее, чем на магнитном диске, состоящем из концентрических дорожек. При поиске ПО устройства приблизительно вычисляет, куда переместить головку, а после перемещения головки считывает первый заголовок, проверяя, насколько точно удалось приблизиться к требуемым данным. Последний байт заголовка содержит код режима.

Желтой книгой определяется два режима:

В режиме 1 используется формат, показанный на Рис. 12, с 16-байтовым заголовком, 2048 байт данных и 288-байтовым кодом исправления ошибок ECC (Error Correction Code), для которого используется перемежающийся код Рида—Соломона.

В режиме 2 поле данных объединяется с полем ECC в 2336-байтное поле данных. Этот режим может использоваться для тех приложений, которым не требуется (или у них нет на это времени) коррекция ошибок, например аудио или видео. Обратите внимание, что коррекция ошибок осуществляется на трех уровнях:

- внутри символа,
- в кадре
- в CD-ROM-секторе.

Однобитовые ошибки исправляются на нижнем уровне, короткие пакеты ошибок корректируются на уровне кадров, а все остальные ловятся на уровне сектора. В результате 98 кадров по 588 бит (7203 байт) содержат всего лишь 2048 байт полезной нагрузки, что соответствует эффективности около 28 %. Такую цену приходится платить за надежность хранения информации на оптическом диске.

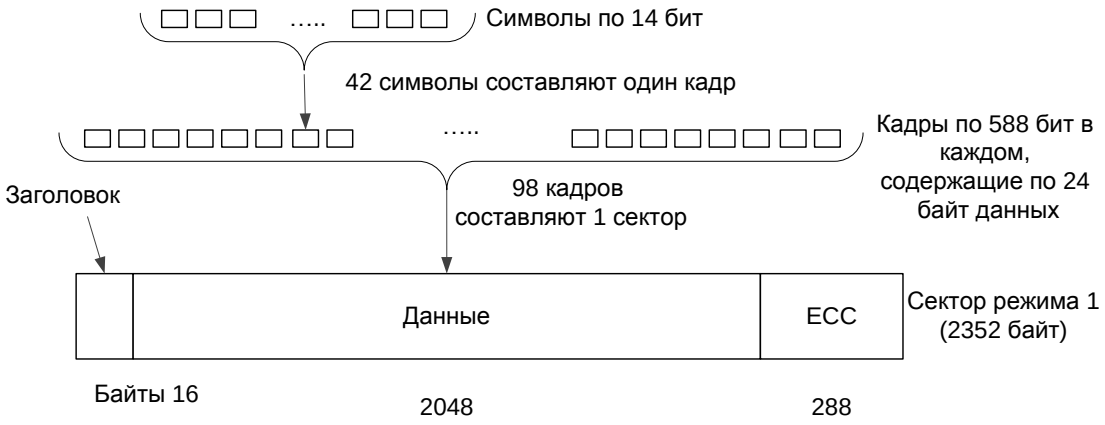


Рис. 12. Логическое расположение данных на CD-ROM

Односкоростные CD-ROM-приводы считывают данные со скоростью 75 секторов в секунду, что соответствует скорости передачи данных 153 600 байт/с в режиме 1 и 175 200 байт/с в режиме 2. Двухскоростные приводы в два раза быстрее и т. д. Таким образом, 40-скоростной CD-ROM-привод может поставлять данные со скоростью до 40x153 600 байт/с (6000 Кбайт/с), при условии, что интерфейс устройства, шина и ОС могут обработать такой поток данных. На стандартном аудио CD помещается до 74 мин музыки, что при использовании его для хранения данных в режиме 1 дает емкость в 681 984 000 байт. Эта величина обычно обозначается как 650 Мбайт, так как 1 Мбайт равен 2^{20} байт (1 048 576 байт), а не 1 000 000 байт.

В 1986 году фирма Philips снова нанесла удар, выпустив Зеленую книгу, добавив к стандарту графику и возможность совмещения в одном секторе аудио, видео и данных, что существенно для мультимедийных CD.

Наконец, следует рассмотреть ФС, используемую на CD-ROM. Чтобы один и тот же CD-ROM можно было использовать на различных компьютерах, необходимо добиться соглашения по вопросу ФС для CD-ROM в виде Международного стандарта ISO 9660.

Эта ФС состоит из трех уровней:

На первом уровне файлы имеют имена формата, схожего с MS-DOS — до 8 символов имя файла плюс до— 3 символов расширение. В имени файла могут использоваться только прописные символы, цифры и символ подчеркивания. Глубина вложенности каталогов ограничена восемью. Имена каталогов не могут содержать расширений. Все файлы уровня 1 должны быть непрерывными, чего не сложно добиться на носителе, на который информация записывается всего один раз. Таким образом, любой CD-ROM, соответствующий уровню 1 стандарта ISO 9660, может быть прочитан в системе MS-DOS, на компьютере Apple, в системе UNIX, то есть практически на любом компьютере. Производители CD-ROM считают это свойство большим плюсом.

Уровень 2 стандарта ISO 9660 разрешает использовать имена файлов длиной до 32 символов.

Уровень 3 позволяет использовать сегментированные файлы.

Расширения стандарта Rock Ridge разрешают использовать очень длинные имена (для UNIX), а также универсальные идентификаторы UID, глобальные идентификаторы GID и связывание файлов.

CD с возможностью записи

Эти носители известны под названием CD-R (CD-Recordable — компакт-диски с возможностью записи).

Физически CD-R, так же как и обычные CD-ROM, состоят из 120-мм поликарбонатных пластин, с той разницей, что на них нанесена спиральная дорожка глубиной 0,6 мм для направления луча лазера при записи. Дорожка выполнена в виде синусоиды с амплитудой в 0,3 мм и частотой 22,05 кГц (при одинарной скорости вращения диска), что обеспечивает постоянную обратную связь, позволяющую отслеживать и корректировать скорость вращения диска. CD-R выглядят как обычные CD-ROM. В отличие от обычных CD, на поверхности которых пресс-формой нанесены углубления, на CD-R-дисках отличия в отражающей способности питов и промежутков между ними

достигается при помощи добавления специального слоя красителя между поликарбонатом и отражающей золотой поверхностью, как показано на Рис. 13. Используются два типа красителя: синевато-зеленый цианин и желтовато-оранжевый фталоцианин.

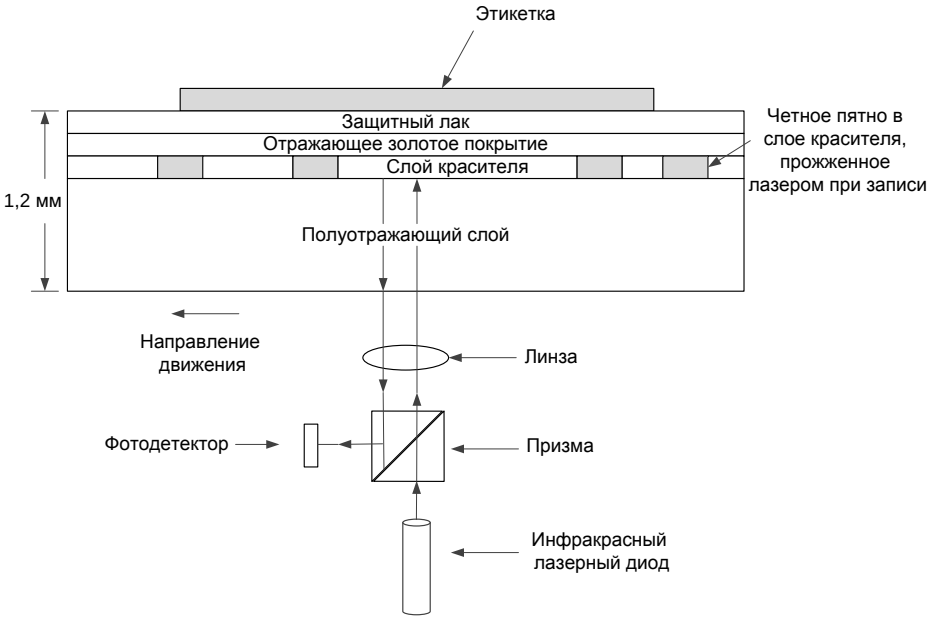


Рис. 13. Поперечный разрез CD-R и лазера. Серебристые CD-ROM имеют сходную структуру с той разницей, что у них нет слоя красителя, а вместо золотого покрытия — алюминиевое с углублениями

В изначальном состоянии уровень красителя прозрачен и позволяет лучу лазера проходить сквозь него и отражаться от золотого покрытия. Во время записи лазер переводится в режим высокой мощности (8-16 мВт). Когда луч лазера попадает на краситель, он нагревает его, разрушая химические связи. При этом образуется темное пятно. При чтении лучом лазера с мощностью 0,5 мВт фотодетектор замечает разницу между темными пятнами, прожженными лазером, и нетронутыми прозрачными областями.

В 1989 году была опубликована Оранжевая книга, посвященная CD-R, а также новому формату CD-ROM XA, позволяющему посекторно дописывать информацию на CD-R. Группа последовательных секторов, записываемых за один раз, называется CD-ROM-дорожкой.

До выхода в свет Оранжевой книги у всех CD-ROM было единое оглавление тома **VTOC** (Volume Table of Contents) в начале диска. При использовании инкрементной (многодорожечной) записи такая схема перестает работать. Решение, опубликованное в Оранжевой книге, состоит в том, что для каждой CD-ROM-дорожки создается собственное оглавление тома VTOC. В нем могут перечисляться некоторые или все файлы предыдущих дорожек. Когда CD-R вставляется в дисковод, ОС перебирает все CD-ROM-дорожки в поисках самого последнего оглавления тома, соответствующего текущему состоянию диска. Включением в новое оглавление тома не всех файлов из предыдущего оглавления создается иллюзия удаления файлов с диска. Дорожки могут объединяться в сеансы, образуя **диски с многосеансовой записью**. Стандартные проигрыватели CD не поддерживают многосеансовые CD, так как они рассчитаны на единственное оглавление тома в начале диска.

Каждая дорожка должна быть записана на CD за одну непрерывную (без остановок) операцию. В результате HDD, с которого производится запись, должен быть достаточно быстрым, чтобы успевать предоставлять записываемые данные вовремя. Если копируемые на диск файлы разбросаны по всему HDD, время поиска может оказаться слишком долгим, что приведет к опустошению буфера и «пересыханию» потока данных, поступающих на CD-R. В результате вместо диска с записями получится блестящая подставку под стаканы. ПО для устройств записи CD-R обычно предлагает собрать все записываемые файлы в виде единого непрерывного образа CD-ROM, размером в 650 Мбайт, прежде чем прожигать CD-R, но эта процедура удваивает общее время записи диска и требует 650 Мбайт свободного места на жестком диске. К тому же

даже такая мера не спасает от случайностей, как, например, перегрев HDD, в результате которого он вдруг начинает выполнять операцию термальной калибровки.

Многократно перезаписываемые компакт-диски

Такая технология появилась под названием CD-RW (CD-ReWritable). CD-RW по размерам ничем не отличаются от обычных компакт-дисков, CD-ROM или CD-R. Однако вместо цианина или фталоцианина в CD-RW в качестве записывающего слоя используется сплав серебра, индия, сурьмы и теллура. У этого сплава два устойчивых состояния: кристаллический и аморфный, с различной отражающей способностью.

В устройствах записи CD-RW используются лазеры с тремя различными уровнями мощности. При высокой мощности лазер расплавляет сплав, преобразуя его из кристаллического состояния с высокой отражающей способностью в аморфное состояние с низкой отражающей способностью, соответствующее питу. При средней мощности лазер плавит сплав, превращая его в кристаллическое состояние, соответствующее промежуткам между питами. При низкой мощности лазер считывает информацию с диска, не вызывая фазовых переходов сплава.

DVD

Базовый формат компакт-дисков (CD/CD-ROM) не менялся с 1980 года. Однако технология с тех пор ушла вперед, в результате чего стало экономически возможным создание оптических дисков большей емкости. Спрос на такие диски уже весьма велик и продолжает возрастать.

В результате объединения технологических усилий и интересов трех богатейших и мощнейших отраслей промышленности и была создана система DVD. (DVD, Digital Video Disk).

Таймеры

Таймеры (также называемые часами) очень важны для работы любой многозадачной системы по ряду причин. Среди многих других задач, они следят за временем суток и не позволяют одному процессу надолго занять CPU. ПО таймера может принимать форму драйвера устройства, несмотря на то, что таймер не является ни блочным, ни символьным устройством.

Аппаратная часть таймеров

В компьютерах широко применяются два типа таймеров:

- Наиболее простые компьютерные часы привязываются по частоте к линии питания переменного напряжения 110 или 220 В и вызывают прерывания при каждом цикле напряжения с частотой 50 или 60 Гц. Такие часы очень широко применялись ранее, но сейчас являются редкостью.
- Другой тип часов состоит из трех компонентов: кварцевого генератора, счетчика и регистра хранения, как показано на Рис. 14. Если взять кусок кристалла кварца правильного размера и установить его в оправу под давлением, то можно заставить его колебаться и выдавать электрический сигнал с частотой в несколько сот мегагерц. Частота зависит от конкретного кристалла, но каждый кристалл выдерживает эту частоту с достаточно высокой точностью. С помощью электроники эту частоту можно поднять до 1 ГГц или даже до еще более высокой частоты. По крайней мере, одна такая схема обязательно присутствует в каждом компьютере, обеспечивая сигнал синхронизации для различных цепей компьютера. Этот сигнал подается на вход декрементного счетчика. Когда содержимое счетчика достигает нуля, он вызывает прерывание CPU.

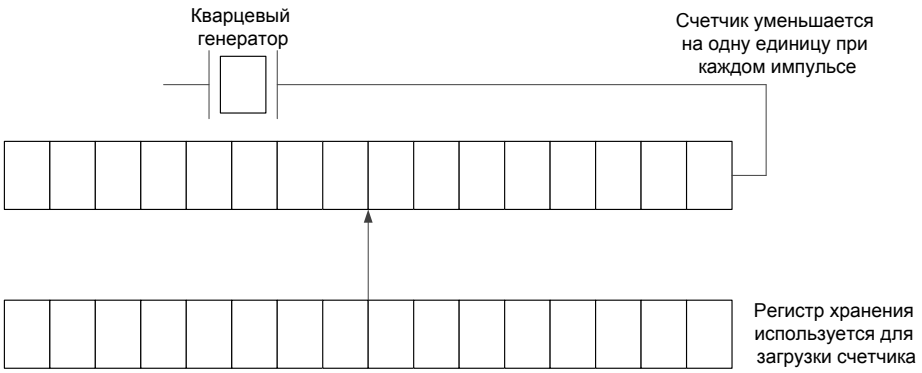


Рис. 14. Программируемый таймер

У программируемого таймера обычно есть несколько режимов работы:

В режиме одновибратора при запуске таймера содержимое регистра хранения копируется в счетчик. Затем содержимое счетчика уменьшается на единицу при каждом импульсе от кристалла. Когда счетчик достигает нуля, он вызывает прерывание и останавливается до тех пор, пока он не будет снова явно запущен программным обеспечением.

В режиме генератора прямоугольных импульсов при достижении счетчиком нуля инициируется прерывание, а содержимое регистра хранения автоматически копируется в счетчик, и весь процесс повторяется снова бесконечно.

Преимущество программируемого таймера состоит в том, что частота прерываний от него может управляться программно. Микросхемы программируемых таймеров обычно содержат два или три независимо программируемых счетчика и помимо этого обладают целым рядом других функций (например, могут увеличивать, а не уменьшать значение счетчика, не инициировать прерываний и т. д.).

Чтобы показания таймера не терялись, пока питание компьютера выключено, часы компьютеров питаются от аккумулятора. Показания часов считываются при загрузке ОС. Если таких часов у компьютера нет, ОС может запросить дату и время при запуске. Кроме того, система может узнать эти сведения по сети от удаленного хоста. В любом случае эти время и дата транслируются в количество интервалов таймера с какого-либо момента, например полуночи 1 января 1970 года по всеобщему скоординированному времени (UTC, Universal Coordinated Time), как это делает, например, система UNIX. До 1928 года время UTC называлось средним временем по Гринвичу (GMT, Greenwich Mean Time). В системе Windows время отсчитывается от 1 января 1980 года, При каждом прерывании от таймера счетчик времени увеличивается на единицу. В ОС обычно присутствуют ПО, позволяющие скорректировать показания системных часов.

Программное обеспечение таймеров

Все, что делает таймер, аппаратно — он инициирует прерывания через определенные интервалы времени. Все остальное, связанное со временем, должно выполняться программно драйвером часов. Обязанности драйвера часов варьируются в зависимости от ОС, но обычными являются следующие функции:

1. Следят за временем суток.

2. Не позволяют процессам работать дольше, чем им разрешено.

3. Ведут учет использования CPU.

4. Обрабатывают системный вызов alarm, инициированный процессом пользователя.

5. Поддерживают следящие таймеры для ОС.

6. Ведут наблюдение, анализ и сбор статистики.

Первая функция часов, поддерживающая время суток (также называемое истинным временем), не сложна. Она просто требует увеличения счетчика на единицу при каждом импульсе сигнала времени часов, Рис. 15(а). Нужно только следить за количеством битов в счетчике времени суток. При частоте импульсов сигнала времени 60 Гц 32-разрядный счетчик переполнится уже за два года. Очевидно, система не может хранить значение истинного времени в тиках с 1 января 1970 года в 32 бит.

Для данной проблемы возможны три решения:

Во-первых, можно использовать 64-разрядный счетчик, хотя это потребует больших затрат, так как увеличивать значение счетчика придется помногу раз в секунду, Рис. 15(б).

Второй способ, хранение времени суток не в тиках (количестве импульсов сигнала времени), а в секундах, переводя импульсы сигнала времени в секунды при помощи дополнительного счетчика, Рис. 15(б). Поскольку 2^{32} с — это больше, чем 136 лет, такой метод будет работать вплоть до 22-го века.

Третий способ, учитывать импульсы сигнала времени, но относительно того момента, в который была загружена машина, а не от фиксированного внешнего момента. При этом система во время загрузки узнает текущее время, которое сохраняет в памяти в любом удобном виде. Позднее, при запросе времени, система складывает хранящееся время загрузки со значением счетчика, чтобы получить текущее время, Рис. 15(в).

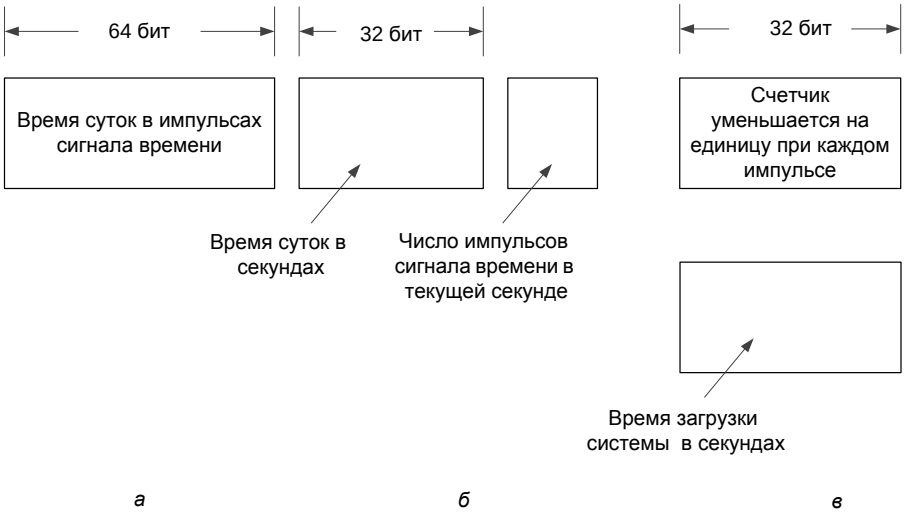


Рис. 15. Три способа реализации времени суток

Вторая функция часов состоит в недопущении слишком долгой работы процесса. При запуске процесса планировщик инициализирует счетчик, записывая в него выделенное этому процессу количество импульсов сигнала времени. При каждом прерывании от таймера драйвер таймера уменьшает значение счетчика на 1. Когда значение счетчика достигает нуля, драйвер таймера вызывает планировщик, чтобы тот запустил другой процесс.

Третья функция часов состоит в учете использования CPU. Наиболее точно это может быть сделано, если при каждом запуске нового процесса запускать второй таймер, независимый от основных системных часов. Когда процесс останавливается, значение таймера считывается, чтобы определить, сколько времени работал процесс. Чтобы все было правильно, значения второго таймера должны сохраняться на время прерываний.

Не столь точный, но более простой метод учета состоит в создании указателя текущего процесса в таблице процессов в виде глобальной переменной. При каждом импульсе сигнала времени поле текущего процесса в таблице увеличивается на 1. Таким образом, каждый импульс сигнала времени «заботится» о текущем процессе. Недостаток этого метода состоит в том, что в случае частых прерываний во время работы процесса ему все равно будет засчитана работа в течение полного импульса сигнала времени.

Во многих системах процесс может попросить ОС выдать ему сигнал предупреждения после определенного интервала времени. Предупреждение может быть сигналом, прерыванием, сообщением и т. п. Такие предупреждения нужны, например, для работы в сети, при которой пакет, не получивший подтверждения в течение определенного интервала времени, должен быть передан повторно. Другим приложением может быть обучающее ПО, ожидающее ответа на вопрос в течение установленного интервала времени.

Если драйвер часов управляет достаточным количеством таймеров, он может установить таймер для каждого запроса. Если физических таймеров недостаточно, они

легко могут быть смоделированы программно. Один из способов реализации большого числа виртуальных таймеров состоит в создании таблицы, хранящей все времена сигналов для обрабатываемых таймеров, а также переменная, в которой хранится время срабатывания ближайшего таймера. При каждом обновлении времени суток драйвер проверяет, не пора ли подавать сигнал от ближайшего таймера. При этом ищется следующий по времени таймер.

Если ожидается много сигналов, то более эффективным считается реализовать их в виде сортированного связного списка, как показано на Рис. 16. Каждый элемент списка содержит число импульсов сигнала времени относительно предыдущего таймера. В данном примере сигналы должны быть поданы в моменты времени 4203, 4207, 4213, 4215 и 4216.

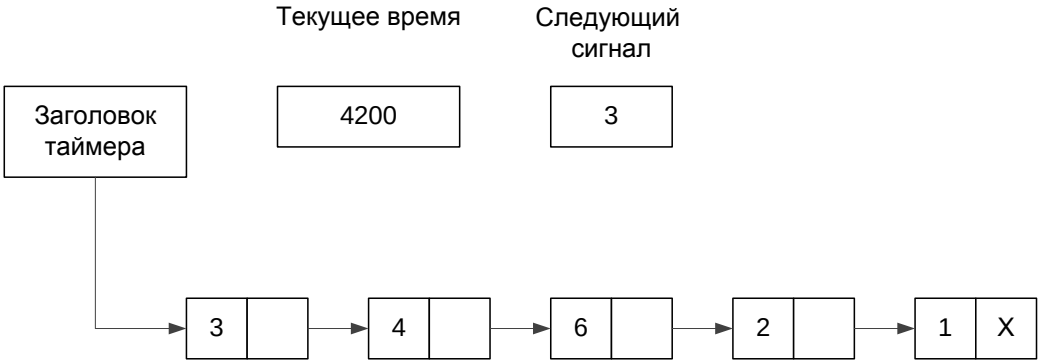


Рис. 16. Моделирование нескольких виртуальных таймеров

На Рис. 16 следующее прерывание произойдет через 3 тика. На каждом тике значение переменной **Next signal**, хранящей число тиков, оставшееся до подачи следующего сигнала, уменьшается на 1. Когда значение этой переменной достигает нуля, подается сигнал в соответствии с первым элементом списка, который затем удаляется из списка. После этого переменной **Next signal** присваивается значение следующего элемента списка, то есть 4 в нашем примере.

Обратите внимание, что за время прерывания от таймера драйвер часов должен выполнить несколько действий: увеличить показания часов истинного времени, уменьшить значение кванта времени, выделенного текущему процессу, и сравнить его с нулем, выполнить операцию учета использования CPU и уменьшить счетчик таймера тревоги.

Следующая функция таймеров в ОС - сторожевые таймеры. Например, гибкие диски не вращаются, пока ими не пользуются, чтобы избежать слишком быстрого изнашивания носителей и головок дисковода. Когда требуются данные с гибкого диска, следует запустить двигатель. Только если гибкий диск вращается с полной скоростью, может начаться операция ввода-вывода. Когда процесс пытается читать данные с находящегося в состоянии покоя гибкого диска, драйвер FDD запускает двигатель и устанавливает сторожевой таймер, чтобы тот инициировал прерывание спустя время, достаточное для разгона диска. Сторожевой таймер необходим, так как FDD не умеют формировать прерывания, сообщающие о том, что FDD достаточно разогнался.

Механизм обработки сторожевых таймеров, используемый драйвером часов, тот же, что применяется для сигналов пользователя. Единственное отличие состоит в том, что когда таймер срабатывает, вместо подачи сигнала драйвер часов вызывает процедуру, предоставляемую обратившимся к нему процессом. Эта процедура является частью процесса. Она может сделать все, что нужно, даже вызвать прерывание, хотя внутри ядра прерывания часто неудобны, а сигналов не существует. Вот почему предоставляется механизм сторожевых таймеров. Следует заметить, что этот механизм работает, только если драйвер таймера и вызываемая им процедура находятся в одном адресном пространстве.

Последняя функция таймеров — это сбор статистики. В некоторых ОС предоставляется механизм построения гистограммы, показывающей положение счетчика команд программы пользователя. Таким образом, пользователь может видеть, какие процедуры его программы какой процент CPU потребляют (в Windows – Task Manager). Для этого на каждом тике драйвер часов должен проверить, собирается ли статистика по текущему процессу, и если да, то определяет, в каком диапазоне адресов

находится счетчик команд. После этого значение счетчика, соответствующее этому диапазону, увеличивается на единицу.

«Мягкие» таймеры

У большинства компьютеров есть второй программируемый таймер, который может быть установлен для формирования прерываний с той частотой, какая требуется ПО. Этот таймер представляет собой добавление к основному системному таймеру, описанному ранее. До тех пор пока частота прерываний невелика, никаких проблем, связанных с использованием второго таймера для прикладных целей, не возникает. Трудности появляются, когда частота прерываний прикладного таймера становится очень высокой. Ниже опишем схему программного таймера, хорошо работающую в различных обстоятельствах, даже на высоких частотах.

Обычно есть два способа управления вводом-выводом:

- Прерывания.
- Опрос.

Прерывания обладают низким временем задержки, то есть они происходят немедленно после самого события или с минимальной задержкой. С другой стороны, в современных CPU прерываниям сопутствуют значительные накладные расходы, связанные с необходимостью переключения контекста, а также с их влиянием на конвейер, кэш и буфер быстрого преобразования адреса TLB.

Вместо прерываний может использоваться опрос приложением какого-либо порта или слова памяти при ожидании события. Этот метод позволяет избежать прерываний, но может привести к значительным задержкам, то есть к замедленной реакции приложения на ожидаемое им событие.

Для некоторых приложений ни накладные расходы прерываний, ни задержка опроса неприемлемы. Возьмите, к примеру, такую высокоскоростную сеть, как гигабитная сеть Ethernet. Эта сеть способна принимать или доставлять пакет полного размера каждые 12 мкс. Для поддержания оптимальной производительности на выходе надо посылать новый пакет каждые 12 мкс.

Один из способов достижения такой скорости состоит в том, что по завершении передачи каждого пакета происходит прерывание, либо устанавливается таймер, инициирующий прерывания каждые 12 мкс. Недостаток этого метода — одно прерывание занимает 4,45 мкс. Этот показатель накладных расходов вряд ли улучшился с 70-х годов. Например, у большинства мини-компьютеров прерывание занимает всего четыре цикла шины, необходимых для помещения в стек счетчика команд и слова состояния CPU, и для загрузки новых значений PC и PSW.

Идея «мягких» таймеров позволяет избежать лишних прерываний. Вместо этого ядро, вызываемое по какой-либо другой причине, перед тем как вернуться в режим пользователя, проверяет значение часов реального времени, чтобы проверить, не истек ли период ожидания «мягкого» таймера. Если время ожидания истекло, выполняется планируемое событие (например, передача пакета или проверка, не пришел ли пакет). При этом отпадает необходимость специального переключения в режим ядра, так как система и так уже находится в режиме ядра. Когда необходимые действия выполнены, «мягкий» таймер снова устанавливается для ожидания следующего события. Все, что для этого требуется — это взять текущее значение часов, прибавить к нему интервал ожидания и сохранить сумму в ячейке таймера.

«Мягкие» таймеры устанавливаются и срабатывают с той скоростью, с которой выполняются входы в ядро по другим причинам. К этим причинам относятся:

1. Системные вызовы,

2. Ошибки преобразования адреса TLB.

3. Отсутствие страницы памяти.

4. Прерывания ввода-вывода.

5. Временное отсутствие работы для CPU.

Литература

1. Э. Таненбаум. Современные операционные системы. 2-ое изд. –СПб.: Питер, 2002. – 1040 с.

2. А. Шоу. Логическое проектирование операционных систем. Пер. с англ. –М.: Мир, 1981. –360 с.

3. С. Кейслер. Проектирование операционных систем для малых ЭВМ: Пер. с англ. –М.: Мир, 1986. –680 с.

4. Э. Таненбаум, А. Вудхалл. Операционные системы: разработка и реализация. Классика CS. –СПб.: Питер, 2006. –576 с.

5. Microsoft Development Network. URL: http://msdn.com