МИНИСТЕРСТВО ОБРАЗОВАНИЯ УКРАИНЫ

ОДЕССКАЯ ГОСУДАРСТВЕННАЯ

АКАДЕМИЯ ХОЛОДА

кУРСОВАЯ РАБОТА ПО ДИСЦИПЛИНЕ “тЕОРИЯ ОПЕРАЦИОННЫХ СИСТЕМ”

*“ОРГАНИЗАЦИЯ ФАЙЛОВЫХ СИСТЕМ ОПЕРАЦИОННОЙ СИСТЕМЫ os/2”*

Выполнил студент 323А группы Адамян Аваг.

Принял преподаватель Калмыкова Екатерина Анатольевна.

ОДЕССА

1997

Содержание:

|  |  |
| --- | --- |
| Резюме | 3 |
| FAT | 3 |
| HPFS структура тома | 4 |
| Файлы и Fnodes | 5 |
| Каталоги | 5 |
| Расширенные атрибуты | 7 |
| Значение имени | 7 |
| Инсталлируемые файловые системы | 8 |
| Проблемы эффективности | 8 |
| Отказоустойчивость | 9 |
| Прикладные программы и HPFS | 10 |
|  |  |
| Дополнение: Структура системы файлов FAT | 11 |
| Дополнение: Структура блока управления файлом | 11 |
| Дополнение: B Tree и B+Tree | 12 |
| Дополнение: Расширенный блок управления файлами | 12 |
| Резюме | 13 |
| Список использованной литературы | 14 |

Резюме

 Высокоэффективная Файловая Система (HPFS) для OS/2 решает все проблемы FAT. HPFS работает по методу устройства блока произвольного доступа, и также содержит модуль программного обеспечения, который транслирует файловые запросы из прикладных программ к драйверам устройств. HPFS также является инсталлируемой файловой системой, которая делает возможным обращение к нескольким несовместимым структурам тома системы OS/2 одновременно. Превосходная производительность достигается благодаря использованию продвинутых структур данных, интеллектуального кэширования, предварительного чтения и отложенной записи. Дисковое пространство используется более экономно благодаря использованию разбиения на секторы. HPFS также включает значительно улучшенную отказоустойчивость. При необходимости программы используют расширенные атрибуты и длинные имена файлов.

 Высокоэффективная Файловая Система (далее HPFS), которая впервые появилась в OS/2 systemVersion 1. 2, была разработана Gordon Letwin, главным архитектором операционной системы OS/2.

Файловая Система FAT

 Так называемая файловая система FAT использовалась во всех версиях МСДОС и в первых двух выпусках OS/2 (версии 1.0 и 1.1), имеет двойное наследие. Каждый логический том имеет собственный FAT, который выполняет две важные функции: содержит информацию распределения для каждого файла в томе в форме списка связей модулей распределения (кластеров) и указывает, какие модули распределения свободны.

 Когда FAT был изобретен, это было превосходное решение для дискового управления, главным образом потому что гибкие диски, на которых он использовался редко были большими, чем несколько Mb. FAT был достаточно мал, чтобы находиться в памяти постоянно, позволял обеспечивать очень быстрый произвольный доступ к любой части любого файла. Когда FAT был применен на жестких дисках, он стал слишком большим для резидентного нахождения в памяти и ухудшилась производительность системы. Кроме того, так как информация относительно свободного дискового пространства рассредотачивалась "поперек" большого количества секторов FAT, он был непрактичен при распределении файлового пространства, и фрагментация файла стала препятствием высокой эффективности.

 Кроме того, использование относительно больших кластеров на жестких дисках привело к большому количеству неиспользуемых участков, так как в среднем для каждого файла половина кластера была потрачена впустую.

 Ограничения FAT на наименование файлов и каталогов унаследованы из CP/M. Когда Paterson создавал 86DOS, одной из его первых целей было облегчить импорт из CP/M в его новую операционную систему. А следовательно принятые в CP/M ограничения на имена файлов и расширений перенеслись в 86DOS.

 В течение нескольких лет Microsoft и IBM сделали попытку продлить жизнь файловой системы FAT благодаря снятию ограничений на размеры тома, улучшению cтратегий распределения, кэширования имен пути, и перемещению таблиц и буферов в расширенную память. Но они могут расцениваться только как временные меры, потому что файловая система просто не подходила к большим устройствам произвольного доступа.

 HPFS решает проблемы файловой системы, упомянутые здесь и многие другие, но она не является разновидностью файловой системы FAT. Архитектура HPFS начала создаваться как файловая система, которая может использовать преимущества многозадачного режима.

HPFS Структура Тома

 HPFS-тома используют размер сектора 512 байтов и имеют максимальный размер 2199Gb. HPFS том имеет очень небольшое количество фиксированных структур. Секторы 015 тома (BootBlock, имя тома, 32 бита ID, дисковая программа начальной загрузки). Начальная загрузка относительно сложна (в стандартах МСДОС) и может использовать HPFS в ограниченном режиме

 Сектора 16 и 17 известны как SuperBlock и SpareBlock соответственно. SuperBlock изменяется только при помощи утилит. Он содержит указатели свободного пространства, список плохих блоков, полосу блока каталога, и корневую директорию. Он также содержит дату, соответствующую последней проверке и восстановлению утилитой CHKDSK/F. SpareBlock содержит различные флажки и указатели которые будут обсуждаться позже; Он изменяется, хотя нечасто, при работе системы.

 Остаток диска разделен на 8MB полосы. Каждая полоса имеет собственный список свободного пространства, где биты представляют каждый сектор. Бит 0 если сектор использован 1 если сектор доступен. Списки размещаются в начале или хвосте списка.

 Одна полоса, размещенная в "центре" диска, называется полосой блока каталога и обрабатывается специально. Обратите внимание, что размер полосы зависит от текущей реализации и может изменяться в более поздних версиях файловой системы.

Файлы и Fnodes

 Каждый каталог или файл в HPFS-томе закрепляется за фундаментальным объектом файловой системы, называемым Fnode (произносится "eff node"). Каждый Fnode занимает одиночный сектор и содержит управляющую информацию, хронологию доступа, расширенные атрибуты и списки управления доступом, длину и первые 15 символов имени, и структуру распределения. Fnode всегда находится рядом с каталогом или файлом, который он представляет.

 Структура распределения в Fnode может принимать несколько форм, в зависимости от размера каталога или файлов. HPFS просматривает файл как совокупность одного или более секторов. Из прикладной программы это не видно; файл появляется как непрерывный поток байтов.

Каталоги

 Каталоги, подобно файлам, регистрируются в Fnodes. Для корневой директории Fnodes находится в SuperBlock. Fnodes для некорневых каталогов определяются через входы подкаталога.

 Каталоги могут увеличиваться до любого размера и состоят из блоков каталога 2 КБ, которые распределяются как четыре последовательных сектора на диске. Файловая система делает попытку распределить блоки каталога в полосе каталога, которая размещается около “центра” диска. Если полоса каталога полна, блоки каталога распределяются там, где есть свободное место.

 Каждый блок каталога 2 КБ состоит из большого количества входов каталога. Вход каталога содержит несколько полей, включая штампы времени и даты, Fnode указатель, длина имени каталога или файла, имя непосредственно, и указатель. Каждый вход начинается словом, которое содержит длину.

 Число блоков каталога и входов различно при различной длине имени. Если средняя длина имени файла 13 символов, средний блок каталога будет содержать приблизительно 40 входов. Блоки каталога входов сортируются в двоичном лексическом порядке по полям имени в алфавитном порядке для алфавита США. Последний блок каталога входа - запись, которая отмечает конец блока.

 Когда каталог получает слишком большой файл, который нужно сохранить в одном блоке, он увеличивает размер добавлением блоков 2 КБ, которые организуются как B-Tree[[1]](#footnote-1). При поиске файловая система извлекает указатель B-Tree из входа. Если это не указатель, то поиск неудачен; иначе файловая система следует за указателем в следующий каталог и продолжает поиск.

 Небольшая арифметика дает внушительную статистику. При использование 40 входов на блок, блоки каталога дерева с двумя уровнями могут содержать 1640 входов, каталога и дерева с тремя уровнями могут содержать на удивление 65640 входов. Другими словами, некоторый файл может быть найден в типичном каталоге из 65640 файлов максимум за три обращения. Это намного лучше файловой системы FAT, где в самом плохом случае более чем 4000 секторов нужно прочитать для нахождения файла.

 Структура каталога B-Tree имеет интересные импликации. Создание файла, переименование или стирание может приводить к каскадированию блоков каталогов. Фактически, переименование может терпеть неудачу из-за недостатка дискового пространства, даже если файл непосредственно в размерах не увеличился. Во избежание этого “бедствия”, HPFS поддерживает маленький пул свободных блоков, которые могут использоваться при “аварии”;

 Указатель на этот пул свободных блоков сохраняется в SpareBlock.

Расширенные Атрибуты

 Атрибуты Файла - информация о файле. FAT поддерживает только небольшое количество простых атрибутов (доступный только для чтения, системный, скрытый, архив) которые фактически сохраняются как флажки бита на входе каталога файла; эти атрибуты не доступны, если файл открыт.

 HPFS поддерживает те же самые атрибуты, что и файловая система FAT по историческим причинам, но он также поддерживает и новую форму fileassociated, то есть информацию, называемую Расширенными Атрибутами (EAs). Каждый EA концептуально подобен переменной окружения.

Значение имени

 В OS/2 1.2 каждый каталог или файл может иметь максимум 64 КБ присоединенных EAs. Это ограничение снимается в более поздних версиях OS/2.

 Метод хранения для EAs может изменяться. Если одиночный EA становится слишком большим, он может помещаться снаружи Fnode.

 Ядро API функции DosQFileInfo и DosSetFileInfo расширено новыми информационными уровнями, которые позволяют прикладным программам управлять расширенными атрибутами файлов. Новые функции DosQPathInfo и DosSetPathInfo используются для чтения или записи EAs, связанных с произвольными именами пути.

 Поддержка EAs является существенным компонентом в объектно - ориентированных файловых системах. Информация о почти любом типе может сохраняться в EAs. Так как HPFS развивается, средства для управления EAs становятся еще более сложным.

 Можно предположить, например, что в будущем версии API могут расширяться функциями EA, которые являются аналогичным DosFindFirst и DosFindNext и EA - данные могут быть организованы в B-Tree.

Инсталлируемые Файловые Системы

 Поддержка для инсталлируемой файловой системы является очень ценной особенностью OS/2. Она дает возможность обращаться к многочисленным несовместимым томам структур FAT, HPFS, CD ROM, и UNIX. Система OS/2 упрощает жизнь и открывает путь быстрому развитию файловых систем и новшеств. Инсталлируемые файловые системы, однако, релевантны по отношению к HPFS, поскольку они используют HPFS факультативно. Файловая система FAT вложена в ядро OS/2 и будет оставаться там как файловая система совместимости в течение некоторого времени.

 Драйвер инсталлируемой файловой системы (FSD) аналогичен драйверам устройств. FSD постоянно находится на диск как структурный файл, который является подобным библиотеке динамических компоновок (DLL). Обычно этот файл имеет системное расширение, и загружается во время инициализации системы из файла CONFIG. SYS.

 Когда FSD установлен и инициализирован, ядро передает ему логические запросы на открытие, чтение, запись, поиск, закрытие, и так далее. FSD транслирует эти управляющие запросы в запросы на сектор чтения или записи.

Проблемы эффективности

 Здесь рассмотрим лишь часть методик повышения эффективности. Например, когда файл открывается, файловая система прочитает и кэш Fnode и первое небольшое количество секторов содержания файла. Если информация хронологии в Fnode файла выполнимой программы показывает, что операция открытия обычно сопровождается непосредственным последовательным чтением всего файла, файловая система будет читать в кэш еще больше содержания файла.

 В заключение, поддержка операционной системы OS/2 для многозадачного режима делает возможным для HPFS “ленивые записи” (иногда называемый отсроченными записями или запись позади) для улучшения эффективности.

Отказоустойчивость

 Первичный механизм для коррекции ошибок записи называется hotfix. Если обнаружена ошибка, файловая система берет свободный блок вне резервного пула hotfix, записывает данные в этот блок, и модифицирует карту hotfix. (Карта hotfix представляет собой двойные слова, содержащие номер плохого сектора связанный с номером замены. Указатель на hotfix находится в SpareBlock). Выдается предупредительное сообщение, и пользователь знает, что все хорошо не с дисковым устройством.

 Каждый раз когда файловая система запрашивает сектор, дисковый драйвер просматривает карту hotfix и заменяет любые плохие номера секторов на соответствующие хорошие.

 CHKDSK освобождает карту hotfix. Он добавляет плохой сектор в список плохих блоков, выпускает сектор замены обратно в пул hotfix, удаляет hotfix вход из карты hotfix, и записывает модифицированный hotfix.

 HPFS защищает себя от пользователей, слишком любящих Красный\_Переключатель и поддерживает флажок Dirty FS, помечающий в SpareBlock каждый HPFS том. Флажок очищается только когда все файлы тома закрылись и все буферизованные в кэше данные были записаны.

 В течение загрузки OS/2, файловая система осматривает DirtyFS и, если флажок установлен, не будет позволять дальнейший доступ к тому до тех пор, пока CHKDSK не закончит работу. Если флажок DirtyFS установлен на томе начальной загрузки, система не будет загружена; пользователь должен загружать OS/2 в режиме с дискеты и выполнить CHKDSK для проверки и возможно ремонта тома начальной загрузки.

 При потере SuperBlock или корневой директории, HPFS имеет возможность успешно восстановить их. Каждый тип, определяющий файл, включая объект Fnodes, секторы распределения, и каталоги блоков компонуются так, что и хозяин и дочерние записи и содержит уникальную сигнатуру с 32 битами. Fnodes также содержат начальную часть имени каталога или файлов. Следовательно, CHKDSK может восстановить весь том методом просмотр диска для Fnodes, секторов распределения, и блоки каталога, использовав их для восстановления файлов, каталогов и freespace - списков.

Прикладные Программы и HPFS

 Новые функции API, DosCopy помогают прикладным программам создавать копии, по существу дублируя существующий файл вместе с EAs. EAs может также управляться явно с DosQFileInfo, DosSetFileInfo, DosQPathInfo, и DosSetPathInfo.

 Другие изменения в API не будут воздействовать на среднюю прикладную программу. Функции DosQFileInfo, DosFindFirst, и DosFindNext.

 DosQFsInfo используется для получения метки тома или дисковых характеристик точно как прежде, и использование DosSetFsInfo для меток тома не изменяется. Имеются a немногие полностью новые функции API типа DosFsCtl (аналогичный DosDev IOCtl но используемый для связи между прикладной программой и FSD), и DosQFsAttach (определяет, какой FSD имеет том); они предназначаются главным образом для использования дисковыми сервисными программами.

Дополнения

Структура системы файлов FAT

Файл Д

Файл В

Файл Б

Каталог

Файл Г

Каталог

Каталог

Файл А

Идентификатор

дисковода

Корневой каталог

(метка тома)

|  |  |  |
| --- | --- | --- |
| 00Н | 01Н | Идентификатор дисковода |
| 01Н | 09Н | Имя файла (8 символов) |
| 09Н | 0СН | Расширение (3 символа) |
| 0СН | 0ЕН | Номер текущего блока |
| 0ЕН | 10Н | Размер записи |
| 10Н | 14Н | Размер файла |
| 14Н | 16Н | Дата создания (обновления) |
| 16Н | 18Н | Время создания (обновления) |
| 18Н | 20Н | Зарезервировано |
| 20Н | 21Н | Номер текущей записи |
| 21НТипичный блок управления файломЭлементы структурыОтносительный конечный адрес в байтахОтносительный начальный адрес в байтах | 25Н | Номер относительной записи |

B Tree и B+Tree

 Многие программисты не знакомы со структурой данных, известной как двоичное дерево. Двоичные деревья это методика для логического упорядочивания совокупности элементов данных.

 В простом двоичном дереве каждый узел содержит некоторые данные, включая значение ключа, которое определяет логическую позицию узла в дереве, и указатели на левые и правые поддеревья узла. Узел который начинает дерево известен как корень; узлы которые сидят на конце ветви дерева иногда называются уходами.

 Такие простые двоичные деревья, хотя просты в понимании и применении, имеют недостатки (неудобства), обнаруженные практикой. Если ключи распределяются не оптимально или добавляются к дереву в непроизвольном режиме, дерево может становиться совершенно асимметричным, что приводит к большим различиям между временами обхода дерева.

 Поэтому большое количество программистов предпочитают в использовании сбалансированные деревья известные как B-Tree.

|  |  |  |
| --- | --- | --- |
| 00Н | 01Н | 0FFH |
| 01Н | 06Н | Зарезервировано |
| 06Н | 07Н | Байт атрибута |
| 07Н | 08Н | Идентификатор дисковода |
| 08Н | 10Н | Имя файла (8 символов) |
| 10Н | 13Н | Расширение (3 символа) |
| 13Н | 15Н | Номер текущего блока |
| 15Н | 17Н | Размер записи |
| 17Н | 1ВН | Размер файла |
| 1BН | 1DН | Дата создания (обновления) |
| 1DН | 1FН | Время создания (обновления) |
| 1FН | 27H | Зарезервировано |
| 27H | 28H | Номер текущей записи |
| 28H | 2CH | Номер относительной записи |

Расширенный блок управления файлом

Относительный начальный адрес в байтах

Элементы структуры

Относительный конечный адрес в байтах

Резюме

Резюме.

 HPFS решает все исторические проблемы файловой системы FAT. Он достигает превосходной производительности даже при экстремальных ситуациях, например когда очень маленькие файлы и их очень много или немногие очень большие файлы. Это свидетельствует о продуманных структурах данных и преимуществах техники типа интеллектуального кэширования, предварительного чтения, отложенной записи. Дисковое пространство используется экономно. Существующие прикладные программы нуждаются в модификации для использования преимуществ поддержки HPFS для расширенных атрибутов и длинных имен файлов, но эти изменения не сложны. Все прикладные программы будут извлекать пользу из высокой эффективности HPFS и уменьшенной загрузки CPU.

Список использованной литературы:

1. Microsoft Systems Journal, Sept 1989. Получен из Computer Library Periodicals, Jan 1990, Doc #14753
2. OS/2 Warp, К. Данильченко, BHV Київ, 1995
3. Профессиональная работа в MS-DOS, Р.Данкан, Мир, 1993
1. О двоичных деревьях B-Tree смотри в дополнении [↑](#footnote-ref-1)