АКАДЕМИЯ НАУК СССР
СИБИРСКОЕ ОТДЕЛЕНИЕ
АВТОМЕТРИЯ

№ 5 1986

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

УДК 681.322.067

С. Д. КУЗНЕЦОВ, В. Н. ЮДИН
(Москва)

УПРАВЛЕНИЕ ФАЙЛАМИ В КЛАСТЕРНОЙ ОПЕРАЦИОННОЙ СИСТЕМЕ

Обязательной принадлежностью современных операционных систем (ОС) являются более или менее развитые системы управления файлами. Файлы используются самой операционной системой (для хранения, например, загрузочных модулей), компиляторами (как рабочие пространства на внешней памяти), пользователями (для хранения текстов программ или документации) и т. д. Над системами файлов строятся другие информационные системы, более высокого уровня, такие, как системы управления базами данных.

Разные виды использования файлов предъявляют, вообще говоря, различные требования как к организации файлов на внешней памяти, так и к структуре системы управления файлами. Кроме того, в последние годы в условиях нарастания объема хранимой информации и в связи с доступностью физических носителей большого объема, таких, как дисковые пакеты емкостью 100 Мбайт и более, стала очевидной необходимость в эффективных методах работы с большими файлами. Это накладывает новые требования на организацию файлов, что касается в первую очередь характера отображения файлов на дисковой памяти, а также структуры и размера обменной информации, определяющих пропускную способность файловой системы.

В данной работе мы не рассматриваем структуризованные методы доступа, а ограничиваемся уровнем базового файла — файла с прямым доступом, состоящим из блоков фиксированного размера. Этот уровень прослеживается во многих системах. По ходу изложения сравниваем некоторые из них по организации файлов на дисковых пакетах, а также по структуре архива, способам именования его объектов, положению томов в архиве и возможности их переноса из одного архива в другой. С этой же точки зрения описываем реализуемую в настоящее время в кластерной операционной системе (КЛОС) [1] систему управления файлами.

Организация файлов на томе. Подход к построению базовых файлов, казалось бы, уже устоялся. При традиционном поблочном методе отображения под файл отводится совокупность блоков тома, расположенных на нем, вообще говоря, случайным образом. Будем пользоваться термином «том», предполагая, что том отображается на дисковый пакет простым способом: соответствует либо пакету, либо фрагменту пакета. Блок тома соответствует сектору на диске. Размер блока файла, как правило, выбирается равным размеру блока тома либо кратным ему.

Поблочный метод отображения файлов на тома применяется, например, в файловой системе ОС UNIX [2]. Том, называемый в терминологии UNIX «файловой системой», занимает раздел дискового пакета, а блок файла совпадает по размеру с блоком тома (512 байт). Каждый том хранит совокупность файлов и описывается своим суперблоком, который

содержит основные характеристики тома: число блоков, максимально допустимое число файлов и указатель начала списка свободных блоков тома. Отдельная часть тома отводится под *i*-список. Это своеобразный индекс, каждый элемент которого (*i*-узел) соответствует файлу на томе, а номер элемента используется для идентификации этого файла в томе.

В *i*-узле размещается стандартная информация, описывающая файл, в частности отображение файла на том в виде массива указателей, которые содержат номера блоков файла. Кроме того, в массив указателей входят один указатель на блок второго уровня косвенности и один указатель на блок третьего уровня косвенности. Такая организация позволяет создавать динамически расширяющиеся файлы достаточно большого размера.

Таблица свободных блоков тома хранится в виде списка блоков. Эта таблица в начале жизни тома упорядочена по номерам блоков, однако по мере создания и ликвидации файлов таблица перемешивается, что приводит к тому, что вновь создаваемые файлы будут состоять из блоков, хаотически разбросанных по тому. Тогда даже при последовательной работе с файлом чтение (запись) очередного блока будет часто требовать подвода головок, что заметно снижает пропускную способность системы.

В файловой системе [3] файлы также отображаются на том поблочно и метка файла, содержащая одноуровневую таблицу отображения, целиком хранится в оперативной памяти для каждого открытого файла. Файл, имеющий после создания заранее объявленный начальный размер, может потом динамически расширяться. Память при этом выделяется в соответствии со шкалой свободных (занятых) блоков тома. При этом, конечно, распределение блоков файла по тому тоже оказывается случайным.

Создаваемые в архиве файлы сравнительно невелики, если учесть, что их отображение должно при открытом файле храниться в ОЗУ. Правда, размер блока в аппаратуре (256 48-разрядных слов) довольно большой, да и создавался архив на 7 мегабайтных дисках.

Чтобы показать, что в ряде случаев хранение таблицы отображения в ОЗУ при поблочном отображении файла в том создает серьезную проблему, рассмотрим следующий пример. Для хранения одноуровневой поблочной таблицы отображения файла размером 10 Мбайт потребуется 80 Кбайт ОЗУ. Таблицу такого размера слишком накладно целиком хранить в ОЗУ во время работы с файлом, а это означает, что если запросы на обмен с файлом не локализуются, то реально вместо каждого такого обмена будет выполняться два. Первый обмен потребуется для предварительного считывания блока с нужной частью таблицы отображения.

Наличие второго и третьего уровней косвенности отображения файлов в ОС UNIX показывает, что и здесь эта проблема существует, ее только отчасти сглаживает наличие общесистемного пула буферных страниц, причем в силу глобального характера управления этим пулом число обменов при работе с файловой системой, а тем самым и ее пропускная способность становятся непредсказуемыми.

В силу описанных причин, к которым добавляются малый размер блока и отсутствие асинхронного обращения к файловой системе, ее пропускная способность составила в UNIX 2% от пропускной способности аппаратуры, что побудило разработчиков создать новую, ускоренную версию системы управления файлами [4]. Ее основные отличительные особенности следующие.

Том делится на группы цилиндров, каждая группа содержит свой *i*-список и шкалу свободных и занятых блоков. Файл создается только в пределах своей группы цилиндров (если позволяет его размер). Размер блока увеличен и может варьироваться начиная от 4 Кбайт, оставаясь постоянным для каждой группы цилиндров. Гарантируется, что доступ к любому блоку может быть осуществлен за один физической обмен с устройством.

Реальные данные по размерам часто не кратны блоку файла, поэтому остаток последнего блока не используется. Минимальный размер блока в 4 Кбайт приводит к весьма ощутимым потерям дисковой и оперативной памяти. Для борьбы с потерями было решено делить некоторые блоки на фрагменты, кратные 512 байт, и добавлять их в концы файлов, когда их последние блоки не могут быть заполнены полностью. Если в дальнейшем происходит заполнение последнего блока, фрагмент заменяется на целый блок.

Подводя итог небольшому обзору, можно утверждать, что эффективная работа с таблицами отображения возможна лишь в том случае, когда для каждого открытого файла его таблица целиком размещается в ОЗУ. Для этого таблица отображения должна быть компактной.

Размер таблицы отображения может быть существенно сокращен, если отображать файл на одну или несколько непрерывных областей тома — экстентов. Метод отображения экстентами используется, например, в файловой системе FILES-11, работающей в рамках ОС RSX-11 [5]. Совокупность экстентов, на которых расположен файл, описывается в заголовке файла, размер которого составляет 512 байт, независимо от размера файла.

Опишем теперь более конкретно физический уровень системы управления файлами КЛОС. Понятие тома или логического диска реализуется в КЛОС системой ввода-вывода, скрывающей физические особенности накопителей. Размер тома ограничен размером дискового пакета; том занимает непрерывную область на пакете. Система ввода-вывода оповещает файловую систему о постановке и снятии каждого тома. Если файловая система опознает предложенный ей том как свой, она запрашивает этот том у системы ввода-вывода, после чего выполняет некую процедуру подключения этого тома.

Отображение файла представляется в виде массива записей, каждая из которых описывает размер экстента и его положение на томе и в файле. Эта таблица включается в заголовок файла, который не входит в адресное пространство файла. При преобразовании номера блока в файле в номер блока на томе таблица отображения просматривается до нужной записи. Кроме того, заголовок файла содержит служебную информацию о файле (метку файла), а также метку пользователя, в которой он может хранить небольшой объем «прозрачной» для файловой системы информации.

В томе выделяется специальная область — индекс. В нее помещаются первые блоки всех заголовков всех файлов данного тома. Номер в индексе используется для идентификации файла в томе. Каждый блок индекса, и свободный, и занятый, содержит индексный счетчик, значение которого циклически увеличивается на единицу при каждой ликвидации файла, доступ к которому осуществлялся через данный блок. При инициализации каждый том получает уникальный идентификатор, и тройка ⟨уникальный идентификатор тома, номер по индексу, индексный счетчик⟩ представляет физический идентификатор файла в системе.

Еще одна служебная область тома — область описания свободной памяти (ООСП) — содержит массив записей, описывающих экстенты свободного пространства тома в порядке их расположения. После инициализации тома вся его свободная память образует один экстент. При освобождении экстентов происходит их слияние с соседними, если это возможно.

Предусмотрена программная выбраковка дефектных блока или группы блоков. При этом группа «плохих» блоков выделяется в отдельный экстент и изымается из файла. Для замены плохих блоков через ООСП выделяется новый экстент, а номера плохих блоков записываются в специальную область.

Специфичным видом файлов являются такие, которые существуют лишь от создания до первого закрытия — так называемые временные файлы. В файловой системе КЛОС временные файлы создаются в рам-

ках свободной памяти тома, причем экстенты таких файлов не изымаются из ООСП, а лишь помечаются как временно занятые. Заголовки этих файлов хранятся только в ОЗУ; при запуске системы все экстенты, помеченные как временно занятые, этой пометки лишаются.

По характеру выделения памяти файлы подразделяются на статические и динамические. Статический файл имеет непрерывное адресное пространство, размер которого задается при создании файла и не меняется за время его существования. При создании файла для него выбирается минимально возможное число экстентов, в идеале — один. Для таких файлов допускаются операции чтения и записи группы смежных блоков файла.

Динамические файлы при создании также получают пространство требуемого размера, но для них возможно дальнейшее выделение пространства по явной команде запроса с указанием нужного числа блоков. Система выбирает из ООСП экстент нужного размера, располагает его в свободном месте адресного пространства файла и сообщает в ответ на запрос номер первого блока выделенного экстента. Разрешен также отказ от группы блоков файла.

В процессе работы с файлом его отображение может изменяться. Каждое изменение отображения должно сопровождаться фиксацией (сбросом на том) заголовка файла и ООСП. Поскольку процедура не атомарная, внутри нее возможны остановки процессора. Порядок фиксации определяет, произойдет ли в этом случае дублирование экстента в файле и ООСП, или же он будет вовсе потерян. Как обычно бывает, проще реализуется сборка ранее потерянной памяти, чем поиск дважды использованной, который требуется после каждого сбоя. Поэтому при запросе блоков сначала фиксируется ООСП, а затем заголовок файла, а при возврате — наоборот.

Если заголовок файла занимает более одного блока, то остановка процессора в момент, когда начата, но не закончена запись заголовка, приводит к рассогласованию информации на томе, к потере файла и необходимости восстанавливать том. Во избежание этого та часть заголовка файла, которая не входит в индексный блок, дублируется и каждый экземпляр отображается на отдельный экстент. При считывании заголовка сначала читается его первый блок, а затем остаток, при записи — наоборот.

В различных файловых системах применяются разные методы поддержки логической целостности файлов. Например, один из методов — запоминание согласованного состояния в отдельной версии файла. Открытие файла в режиме модификации автоматически порождает новую версию файла, куда и помещаются модифицированные блоки. Этот метод просто и легко реализуем, но неэкономен и требует большого внимания со стороны пользователей, которые должны вовремя уничтожать ненужные и испорченные версии.

Файловая система [3] использует более экономичный механизм рабочей копии, который поддерживает для открытого файла два отображения — теневое (для хранения согласованной версии) и текущее. Теневое отображение содержится во внешней памяти, текущее — в ОЗУ. Блоки, которые не модифицировались с момента открытия файла, являются общими для обоих состояний. При попытке записи блока запрашивается свободный блок на томе; он помещается в текущее отображение; запись идет в новый блок; старый блок остается в теневом отображении.

Для файлов с рабочей копией имеются команды фиксации состояния (запись текущего отображения на том) и восстановления (чтение с тома теневого отображения на место текущего). В случае сбоев процессора внутри сеанса работы с таким файлом на томе останется согласованный вариант этого файла, к которому и будет представлен доступ при следующем открытии.

Механизм рабочей копии очень жестко ориентирован на блочные отображения, однако по сравнению с простым удвоением памяти при соз-

дании новой версии файла он кажется весьма заманчивым. В КЛОС применяется расширение механизма рабочей копии, суть которого в следующем.

Файлы с теневым механизмом имеют два непересекающихся отображения; обозначим их 0 и 1. Заголовок файла, кроме таблиц отображений, содержит шкалы теневого и текущего состояний файла. Каждый разряд шкалы указывает, в каком из отображений (0 или 1) находится блок соответствующего состояния.

Когда файл закрыт, его текущее состояние совпадает с теневым. После открытия теневое состояние остается неизменным, а вся работа производится по шкале текущего состояния, причем запись блока происходит через отображение, противоположное тому, в котором находится соответствующий теневой блок. Команда восстановления записывает теневую шкалу на место текущей, команда фиксации — наоборот. Выполнение команды фиксации вызывает запись заголовка файла на том.

Теневой механизм можно создать для любого динамического файла. После этого его второе отображение будет постепенно расти по мере увеличения числа модифицированных экстентов первого отображения, достигая в пределе такого же размера, что и первое. По затратам памяти это эквивалентно созданию второй версии. Реализуется соответствующая команда для ликвидации теневого механизма файла. При этом все блоки текущего состояния собираются в одно отображение, а все оставшиеся экстенты возвращаются в ООСП.

Структура архива. В развитых файловых системах архив представляется в виде иерархии именованных объектов, конечными узлами которой являются файлы, а промежуточными — справочники, через которые происходит доступ к узлам вниз по иерархии.

Путь к файлу или справочнику от выделенного узла адекватен составному имени объекта как совокупности элементарных имен узлов или связей, составляющих этот путь. Часто возникает потребность использовать укороченные имена внутри некоторого подархива, характеризуемого своим справочником, например справочником пользователя или справочником тома, в пределах которого идет работа. Тогда этот справочник — база для образования короткого составного имени.

В ОС RSX-11 архив представляет собой совокупность томов, каждый из которых имеет трехуровневую древовидную структуру. Том соответствует дисковому пакету и становится доступным системе после установки его на конкретное устройство по идентификатору этого устройства. Каждый том представляет собой изолированную структуру, и перенос тома из одной файловой системы в другую не создает никаких проблем.

В архивах с объединенной структурой система по имени объекта сама организует поиск, при необходимости требуя установить нужный том. Таким архивом является, например, архив в [3]. В нем допускается включение в справочники ссылок на объекты, уже присутствующие в других местах архива. Для этого используется аппарат альтернативных имен, когда ссылка на объект в справочнике заменяется полным составным именем объекта, для поиска которого необходимо вновь двигаться от корня дерева архива. Перенос томов из одного архива ОС ЦП в другой затруднителен.

Архив ОС UNIX — это тоже единая структура, ориентированный граф с выделенным корневым справочником. Допускаются связи от справочников к файлу внутри одного тома. Возможно короткое именование объектов архива от любого справочника. Подключение томов к архиву происходит на основе использования аппарата специальных файлов. В ускоренной версии файловой системы введена возможность использовать альтернативные имена для ссылок между томами.

При разработке логической структуры архива КЛОС мы руководствовались следующими целями: достаточная гибкость связей (допускаются связи с любыми объектами архива, в том числе на других томах);

возможность применения коротких имен при базировании на разных справочниках; физическая идентификация объектов; обеспечение автономности томов.

Структура хранения — ориентированный граф, узлы которого являются справочниками и файлами. Из справочника может выходить любое количество именованных связей к другим справочникам и файлам. Если от справочника к объекту ведет цепочка связей, то обобщенное имя объекта относительно этого справочника состоит из физического идентификатора справочника и логического имени — цепочки имен связей на этом пути. Для пользователя структура физического идентификатора прозрачна.

Операция настройки на справочник по его обобщенному имени определяет и выдает физический идентификатор справочника, который в дальнейшем может быть использован для короткого именования объектов, подчиненных этому справочнику.

Существование объектов архива не связано с существованием связей к ним. Уничтожение последней связи к объекту не влечет автоматического уничтожения объекта, к которому по-прежнему возможен доступ через физический идентификатор. Уничтожение объекта не влечет автоматического уничтожения всех связей с этим объектом. Корректность ссылки обеспечивается уникальностью физического идентификатора объекта.

Подключение справочников новых томов к централизованному архиву производится с помощью явного проведения связей к ним из справочников архива. Связи с другими томами осуществляются косвенно, через таблицу внешних томов данного тома. Это обеспечивает достаточную гибкость при переименовании томов и при других действиях уровня администратора архива.

Архитектура файловой системы КЛОС. Коротко напомним основные понятия КЛОС. Кластер — это объединение объекта с программой управления этим объектом. Операции над объектом можно интерпретировать как обращение на входы его программы обработки. Синхронизацию операций над объектом производит сам кластер-объект, выбирая в том или ином порядке обращения от других кластеров. Каждый кластер выполняется асинхронно по отношению к другим кластерам как независимый процесс.

Файловая система в кластерном исполнении содержит кластеры, соответствующие основным объектам архива — томам (по числу подключенных томов) и файлам (по числу открытых файлов). Функцию коммутации обращений осуществляет кластер-администратор.

Рассмотрим для примера, как происходит обработка команды «Открыть файл» по обобщенному имени файла. Команда поступает к администратору, который определяет нужный том и обращается к соответствующему кластеру-тому. Кластер-том осуществляет поиск по цепочке имен внутри своего тома и, если обнаруживает связь с другим томом, передает остаток логического имени и физический идентификатор объекта, на который указывает связь, соответствующему кластеру-тому через посредство кластера-администратора. Кластер-том, который обнаруживает на своем томе искомый файл, создает кластер-файл, через обращения к которому и будет происходить взаимодействие пользователя с файлом.

Применяемая технология позволила резко упростить синхронизацию по сравнению, например, с архивом [3] при поиске объектов в архиве. Каждый кластер-том производит поиск в пределах своего тома, не принимая других обращений, пока либо не найдет файл, либо не встретит связь с другим томом.

Предложенная организация файловой системы в КЛОС основана на анализе и сравнении ряда файловых систем. Надеемся, что файловая система КЛОС будет удовлетворять требованиям как системных, так и прикладных программистов.

ЛИТЕРАТУРА

1. **Иванников В. П., Бурдонов И. Б., Кузнецов С. Д. и др.** Технологический подход к организации информационных систем.— В кн.: Труды III Всесоюз. конф. «Банки данных». Таллин: Политехн. ин-т, 1985.
2. **Ritchie D. M., Thompson K.** The UNIX time-sharing system.— Commun. of the ACM, 1974, v. 17, N 7, p. 365—375.
3. **Бяков А. Ю., Иванников В. П., Кузнецов С. Д.** Архив центрального процессора.— М., 1980. (Препринт/ИТМиВТ; 5).
4. **McKusick M. K. e. a.** A fast file system for UNIX.— ACM Trans. on Computer Systems, 1984, v. 2, N 3, p. 181—197.
5. **RSX-11** Input/Output Operations Reference Manual, 1979, N 12.

Поступила в редакцию 4 ноября 1985 г.

УДК 519.682

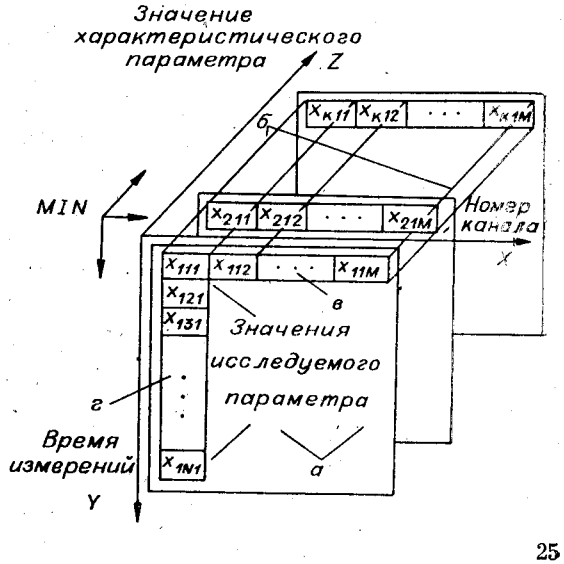
В. А. ВИТТИХ, П. О. СКОБЕЛЕВ

(Куйбышев)

ФУНКЦИОНАЛЬНЫЕ ЯЗЫКОВЫЕ СРЕДСТВА ИНТЕРАКТИВНОГО ВЗАИМОДЕЙСТВИЯ В АСНИ

Одним из путей повышения эффективности применения автоматизированных систем научных исследований (АСНИ) является разработка языковых средств для интерактивного взаимодействия исследователей с ЭВМ на этапе проведения экспериментов.

Необходимость применения языковых средств для АСНИ вызывается обычно требованиями обеспечить возможность оперативного изменения последовательности выполнения специализированных операций по сбору, обработке и отображению данных. Особенности построения указанных средств в значительной степени связаны с технологией проведения эксперимента. В ряде случаев эта технология требует сохранения информации о параметрах эксперимента и порядке их изменения, что предопределяет многомерность данных, поступающих в ЭВМ (рисунок). Операции при этом выполняются, как правило, для «сечений» (по времени и характеристическому параметру, отдельному каналу и т. п.) или фрагментов «сечений» (например, для заданного диапазона изменений параметра и интервала времени) данных. Число возможных направлений для построения таких «сечений» (и применения программ обработки, например *MIN*, на рисунке) обычно велико для многопараметрических объектов исследования, характеризуемых однородными (по физической природе) последовательностями данных измерений. При этом в процессе проведения экспериментов (особенно на ранних стадиях исследований) состав задаваемых (переменных) параметров и последо-



Представление данных эксперимента в координатах задаваемых параметров и «сечения» данных:

а — при выбранных значениях характеристического параметра; *б* — времени измерений; *в* — параметра и времени; *г* — параметра и номера канала