Лекция 14:
Первая файловая система UNIX
BSD UNIX Fast File System

Алексей Линёв
Александр Мощук
Кирилл Погорельский

somes slides are adapted from the OS course at the University of Washington

Таким образом…

- Вы можете подключить диск к нерабочей системе…
- Загрузиться с него…
- Искать, создавать, изменять файлы на диске нерабочей системы
  - поскольку суперблок расположен в фиксированном месте диска, а в нем указано размещение таблицы i-node и блоков данных
  - по соглашению, второй i-node используется для описания корневого каталога тома ФС

Первая файловая система UNIX

- Dennis Ritchie & Ken Thompson, Bell Labs, 1969
- “UNIX rose from the ashes of a multi-organizational effort in the early 1960s to develop a dependable timesharing operating system” – Multics
  (UNIX восстал из пепла совместного проекта создания надежной ОС разделения времени в начале 1960-х – Multics)
- Разработан для рабочих групп, разделяющих одну ЭВМ
- Выполнял свою работу на удивление хорошо
  - Хотя в процессе развития из-за параллельного расширения по многим направлениям он стал достаточно уродливым
- Замечательный пример множества инженерных компромиссов

Формат i-node

- UID (User IDentifier) – пользователь-владелец
- GID (Group IDentifier) – группа-владелец
- Биты прав доступа
- Временны́е характеристики (время выполнения последних операций чтения данных, записи данных, записи i-node)
- Тип объекта файловой системы: обычный файл, каталог, файл устройства и т.д.
- Размер: длина файла в байтах
- Список блоков данных: определяет размещение данных файла в блоках на диске
  - этот вопрос мы рассмотрим немного позже
- Число жестких ссылок - число записей в каталогах, ссылающихся на данный i-node

На всех дисках есть…

- Блок начальной загрузки
  - можно запустить систему, начав загрузку с этого блока
- Суперблок
  - устанавливает границы следующие 4 областей, также содержит заголовки списков свободных i-node и свободных блоков данных
- Таблица i-node
  - содержит описатели (**i-nodes**) всех объектов файловой системы на диске; все i-node имеют одинаковый размер; заголовок списка свободных i-node расположен в суперблоке
- Область блоков данных
  - множество блоков фиксированного размера; заголовок списка свободных блоков расположен в суперблоке
- Пространство подкачки
  - предназначено для хранения процессов, выгруженных из оперативной памяти

"Плоская" (flat) файловая система, основанная на i-node

- Именами файлов являются числа, которые одновременно являются номерами i-node
  - файлы с именами "1", "2", "3" и т.д.
  - почему такая файловая система называется "плоской"?
- При создании файл считается пустым и растет при выполнении операций записи

Древовидная файловая система (иерархическая ФС или ФС с каталогами)

- Каталог – обычный плоский файл, содержащий записи фиксированного размера
- Каждая запись содержит номер i-node и имя файла

i-node number	File name
152	.
18	..
216	my_file
4	another_file
93	oh_my_god
144	a_directory

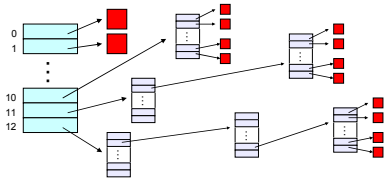
- Так же просто, как и в предыдущем случае!

Максимальный размер файла…

- Использования косвенной адресации 3-ого уровня даст нам еще 128*128*128*512 = 1 Гб
  - 13-ый адрес указывает на блок размером 512б в области данных, который содержит 128 4-байтных адресов блоков, которые содержат по 128 4-байтных адресов блоков, которые содержат по 128 4-байтных адресов блоков, в которых размещены данные файла
- Максимальный размер файла равен 1 Гб + 8 Мб + 64 К + 5 К

"Список блоков данных" в i-node (Unix Version 7)

- Указывает на блоки данных диска, в которых размещаются данные файла
- Должен быть организован таким образом, чтобы описывать и очень маленькие, и очень большие файлы. Как?
- Каждый i-node содержит 13 адресов блоков
  - первые 10 – это "прямые адреса" (адреса блоков данных размером 512 б)
  - 11, 12, 13 – "косвенные адреса" первого, второго и третьего уровней
    - single indirect, double indirect, triple indirect pointers



Максимальный размер файла

- Более поздние версии UNIX из Bell Labs использовали 12 прямых адресов вместо 10
- Berkeley Unix использовал блоки размером 1 Кб
  - Каков максимальный размер файла?
    - 256x256x256x1Кб = 17 Гб +
  - Какова цена подобного изменения?
- Предположим, вы используется блоки размером 4 Кб
  - 1Кб * 1Кб * 1Кб * 1Кб = 4 Тб +

Максимальный размер файла…

- В самом i-node используются 13 * 4 = 52 б
- Можно работать с файлами размером до 10 * 512 = 5 Кб, используя только прямые адреса
  - 10 прямых адресов, размер блока в области данных = 512б
- Можно увеличить размер файла на 128 * 512 = 64 Кб за счет использования косвенной адресации 1-ого уровня
  - 11-ый адрес указывает на блок размером 512б в области данных, который содержит 128 4-байтных адресов блоков, в которых размещены собственно данные файла
- Можно увеличить размер файла еще на 128*128*512 = 8 Мб за счет использования косвенной адресации 2-ого уровня
  - 12-ый адрес указывает на блок размером 512б в области данных, который содержит 128 4-байтных адресов блоков, которые содержат по 128 4-байтных адресов блоков, в которых размещены данные файла

Целостность файловой системы

- i-nodes и блоки данных кэшируются в памяти
- Команда "sync" инициирует немедленную запись содержимого кэша на диск
  - ОС вызывает "sync" каждые несколько секунд
- Сбой ОС или отключение электричества между операциями "sync" может привести к нарушению целостности содержимого диска
- Можно снизить частоту возникновения таких проблем, уменьшив степень кэширования, но при этом сильно пострадает производительность

i-check: целостность плоской файловой системы

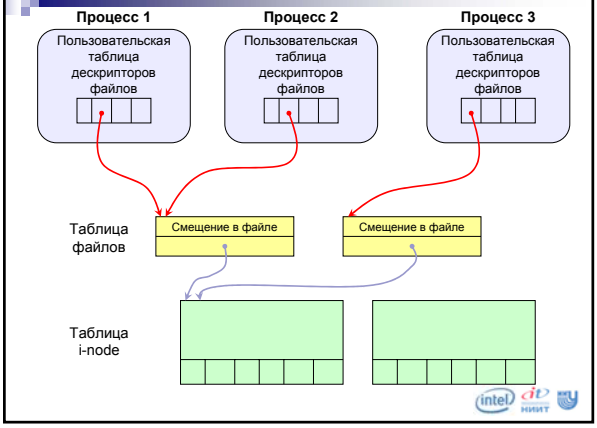
- Принадлежит ли каждый блок в точности одному списку?
  - создаем битовый массив, в котором столько элементов, сколько у нас блоков данных
  - просматриваем список свободных блоков и списки блоков каждого i-node
  - для каждого блока из списков проверяем соответствующий элемент в битовом массиве
    - если значение бита = 0, устанавливаем в 1
    - если значение бита равно 1
      - если блок одновременно принадлежит файлу и списку свободных, удаляем его из списка свободных и двигаемся дальше
      - если блок принадлежит двум файлам, зовем администратора!
  - если по окончании просмотра в битовом массиве остались нули, помещаем соответствующие блоки в список свободных

Разделение файлов

- Каждый процесс имеет "пользовательскую таблицу дескрипторов файлов"
- Каждая запись в пользовательской таблице дескрипторов файлов – это указатель на запись в общесистемной "таблице файлов"
- Каждая запись в таблице файлов содержит текущее смещение в файле (указатель позиции в файле) и указатель на запись в "таблице i-node"
- Когда процесс открывает уже открытый файл, создается новая запись в таблице файлов (с новым значением смещения в файле), указывающая на ту же самую запись в таблице i-node
- Если процесс вызывает fork(), потомок получает копию пользовательской таблицы дескрипторов файлов (и, соответственно, разделяет с родителем контекст файлового ввода-вывода, например – смещения во всех открытых файлах)

d-check: целостность файловой системы с каталогами

- Формируют ли каталоги дерево?
- Совпадает ли число жестких ссылок у каждого файла с числом записей в каталогах, ссылающихся на него?
  - не будем рассматривать детально
    - используется массив счетчиков (по одному на каждый i-node), массив инициализируется нулями
    - сначала обходим дерево, затем просматриваем все i-node



Защита

- Объекты: отдельные файлы
- Субъекты: владелец/группа-владелец/все
- Операции: чтение/запись/исполнение

- Выглядит просто и негибко, но опыт показывает, что подобной схемой можно успешно пользоваться!

Современные реализации файловых систем

- Мы сделали обзор жестких дисков
- Мы сделали обзор основных концепций, связанных с системами управления файлами
- Мы подробно рассмотрели организацию первой файловой системы UNIX (Bell Labs)
  - одновременно простая и практичная структура
  - отлично иллюстрирует инженерные компромиссы, часто встречающиеся при проектировании
- Теперь посмотрим на более сложную файловую систему
  - Berkeley Software Distribution (BSD) UNIX Fast File System (FFS)
    - более высокая производительность, чем у оригинальной UNIX ФС
    - лежит в основе большинства современных файловых систем UNIX

BSD UNIX FFS

- Оригинальная UNIX ФС была изящной, но медленной
  - низкая производительность при работе с диском
    - в среднем – слишком большое количество операций поиска
- В середине 1980-х в рамках Berkeley UNIX была представлена новая структура
  - McKusick, Joy, Fabry, and Leffler
  - улучшена производительность, уменьшено среднее время обработки запроса
  - основная идея FFS – использование при работе информации о структуре диска
    - размещение связанных данных на близко расположенных цилиндрах для уменьшения времени поиска

FFS: увеличенный размер блока, фрагменты

- Оригинальная UNIX ФС использовала блоки размером 512б
  - больше операций поиска
  - малый максимальный размер файла (~1 Гб)
- Если использовать версию с размером блока 1 Кб
  - все еще очень слабо
- FFS использует блоки размером 4 Кб
  - максимальный размер файла ~4 Тб
  - однако, появляется внутренняя фрагментация
    - в среднем, на каждый файл теряется 2 Кб
      - Почему?
    - еще хуже, средний размер файла составляет около 1 Кб!
      - Почему?
  - решение: введение "фрагментов"
    - 1-Кб части блока

Размещение таблицы i-node и блоков данных в UNIX ФС

- Оригинальная UNIX ФС имела 2 основных проблемы с производительностью
  - в ФС, существующей долгое время, размещение блоков данных файла на диске произвольно/случайно
    - в только что созданной ФС блоки данных одного файла расположены последовательно
    - в процессе заполнения и "старения" ФС, необходимо повторно выделять блоки освобожденные при удалении файлов
      - как правило, расположение удаляемых файлов совершенно произвольно
      - таким образом, при создании новых файлов их блоки будут разбросаны по всему диску
  - i-nodes расположены далеко от блоков данных
    - все i-node размещены в начале диска, далеко от данных
    - разбор полного имени файла, операции с файлами и каталогами требуют множества перемещений между таблицей i-node и блоками данных
- Обе указанные причины генерируют множество "медленных" операций поиска!

FFS: использование информации о характеристиках аппаратного обеспечения

- Оригинальная UNIX ФС не использовала информацию о параметрах диска
- FFS устанавливает параметры ФС в зависимости от параметров диска и характеристик ЦП
  - например, оценка времени обработки прерываний центральным процессором и параметры жесткого диска учитываются при выборе места размещения последовательных блоков файла (для уменьшения латентности вращения)

FFS: группы цилиндров

- FFS решает перечисленные проблемы, введя понятие группы цилиндров
  - жесткий диск разделяется на группы цилиндров
  - блоки данных файла располагаются в одной группе цилиндров
  - файлы из одного каталога размещаются в одной группе цилиндров
  - i-node файла располагается в той же группе, что и данные
- Вводит требования к объему свободного пространства
  - для обеспечения возможности выделения свободных блоков в нужной группе, на диске должны быть свободные блоки во всех группах цилиндров
  - в FFS, 10% дискового пространства резервируется для данной цели!
    - отличная мысль: постоянно держать часть диска свободной!
    - по этой причине "df" может показывать заполненность диска > 100%!

FFS: производительность

- Это было очень давно – обращайте внимание на относительную производительность

Type of File System	Processor and Bus Measured	Read Speed	Read Bandwidth	Read % CPU
old 1024	750/UNIBUS	29 Kbytes/sec	29/983 3%	11%
new 4096/1024	750/UNIBUS	221 Kbytes/sec	221/983 22%	43%
new 8192/1024	750/UNIBUS	233 Kbytes/sec	233/983 24%	29%
new 4096/1024	750/MASSBUS	466 Kbytes/sec	466/983 47%	73%
new 8192/1024	750/MASSBUS	466 Kbytes/sec	466/983 47%	54%

Table 2a – Reading rates of the old and new UNIX file systems.

Type of File System	Processor and Bus Measured	Write Speed	Write Bandwidth	Write % CPU
old 1024	750/UNIBUS	48 Kbytes/sec	48/983 5%	29%
new 4096/1024	750/UNIBUS	142 Kbytes/sec	142/983 14%	43%
new 8192/1024	750/UNIBUS	215 Kbytes/sec	215/983 22%	46%
new 4096/1024	750/MASSBUS	323 Kbytes/sec	323/983 33%	94%
new 8192/1024	750/MASSBUS	466 Kbytes/sec	466/983 47%	95%

Table 2b – Writing rates of the old and new UNIX file systems.

ЦП полностью занят выделением блоков!)

FFS: быстрее, но не так изящна (дополнения сделали ее быстрее, но и некрасивее)

- Множество групп цилиндров
  - эффективно, трактует единственный большой диск как множество маленьких дисков
  - дополнительное требование о постоянном наличии свободного пространства (хотя, сейчас оно дешево)
- Увеличенные блоки
  - но введение фрагментов с целью уменьшения внутренней фрагментации
- Использование информации о параметрах аппаратного обеспечения
  - хорошо ли это?