Operating Systems

File Systems

Me

March 29, 2016

План лекции

- Устройства хранения. Планирование IO.
- Файловая система из userspace.
- Пример: FAT.
- Пример: FFS.
- Индексные структуры данных. B-деревья и их вариации.
- Надежность файловых систем. Fsck, журналирование и COW.

Устройства харнения

- Есть довольно много различных устройств хранения:
  - магнитные ленты (и да они до сих пор используются);
  - оптические диски;

Устройства харнения

- Есть довольно много различных устройств хранения:
  - магнитные ленты (и да они до сих пор используются);
  - оптические диски;
- Нас будут интересовать два довольно конкретных вида:
  - механические магнитные диски (HDD);
  - flash-based диски (SSD);

Адресация блоков диска

- Обмен данных с диском происходит блоками фиксированного размера
  - поэтому устройства страницы обычно называют блочными устройствами (выполняют запросы на передачу N блоков)
  - типичный размер блока 512 байт (возможно увеличится в ближайшее время)
  - блоки нумеруются последовательно - это называется LBA;

Адресация блоков диска

- Обмен данных с диском происходит блоками фиксированного размера
  - поэтому устройства страницы обычно называют блочными устройствами (выполняют запросы на передачу N блоков)
  - типичный размер блока 512 байт (возможно увеличится в ближайшее время)
  - блоки нумеруются последовательно - это называется LBA;
- Ранее для HDD использовалась другая адресация - CHS
  - CHS отражает физическое устройство HDD
  - C - cylinder
  - H - head
  - S - sector

Время доступа к диску

- Время выполнения запроса к HDD состоит из нескольких параметров:
  - время позиционирования головки диска (2-4 мс);
  - время ротации - ждем пока нужный сектор окажется под головкой (2-8 мс);
  - время передачи данных (30 мкс);

Время доступа к диску

- Время выполнения запроса к HDD состоит из нескольких параметров:
  - время позиционирования головки диска (2-4 мс);
  - время ротации - ждем пока нужный сектор окажется под головкой (2-8 мс);
  - время передачи данных (30 мкс);
- Последовательный доступ быстрее чем случайный:
  - при последовательном доступе происходит меньше позиционирований;
  - для SSD дисков это тоже справедливо, но в силу других причин;

Планирование IO

- Запросы к диску должны быть большими и редкими
  - несколько процессов обращаются к диску независимо;
  - приложения выдают запросы по частям (например, блоками по 4 KB);
  - запросы на чтение и запись отличаются;

Планирование IO

- Запросы к диску должны быть большими и редкими
  - несколько процессов обращаются к диску независимо;
  - приложения выдают запросы по частям (например, блоками по 4 KB);
  - запросы на чтение и запись отличаются;
- Задачу накапливания/слияния запросов к диску выполняет планировщик IO
  - планирование должно быть честным;
  - планировщик не поможет с случайными запросами;
  - иногда параллельность важнее последовательности;

Файловая система

- Файловая система - иерархическая организация чего-либо в файлы и каталоги:
  - файл - последовательность байт;
  - каталог - набор файлов и других каталогов;

Файловая система

- Файловая система - иерархическая организация чего-либо в файлы и каталоги:
  - файл - последовательность байт;
  - каталог - набор файлов и других каталогов;
- Файл - удобная абстракция:
  - чтение/запись последовательности байт очень универсальный интерфейс:
    - работа с сокетом - чтение/запись последовательности байт;
    - устройства могут быть представлены как файлы (/dev/ttyACM0, /dev/input/mouse);
    - порции данных на диске могут быть организованы в файлы;
    - UNIX - все есть файл.

Типичный интерфейс файловой системы

- open/create - открытие и создание файла;
- read/write/seek - чтение/запись/позиционирование файла;
- close - закрытие файла;
- link/unlink - создание и удаление жесткой ссылки на файл;
- mkdir - создание каталога;
- opendir - открытие каталога;
- readdir - чтение содержимого каталога;
- closedir - закрытие каталога;

Расширенный интерфейс файловой системы

- Зачастую современные ФС предоставляют много других возможностей:
  - truncate - задание размера файла и punch hole;
  - дефрагментация - переупорядочивание содержимого ФС на диске;
  - шифрование;
  - списки контроля доступа;
  - RAID;
  - и многое другое...

Файловый дескриптор

- В UNIX есть два понятия: File Descript*or* и File Descript*ion*:
  - File Descript*or* с точки зрения программиста - просто число идентифицирующее открытый файл;
  - можете смотреть на него как на индекс в таблице открытых файлов процесса;
  - дескриптор имеет смысл только в рамках процесса;

Файловый дескриптор

- В UNIX есть два понятия: File Descript*or* и File Descript*ion*:
  - File Descript*or* с точки зрения программиста - просто число идентифицирующее открытый файл;
  - можете смотреть на него как на индекс в таблице открытых файлов процесса;
  - дескриптор имеет смысл только в рамках процесса;
- File Descript*or* - это ссылка на File Descript*ion*:
  - дескриптор у каждого процесса свой, но они могут ссылаться на один и тот же открытый файл;
  - File Descript*ion* в частности хранить текущее смещение в файле - могут быть неприятности после fork;
  - чтобы создать новый File Descript*ion* нужно заново открыть файл.

Задача файловой системы

- Основная задача ФС сопоставить имени файла и смещению в файле смещение на диске
  - ФС должна поддерживать информацию о свободном/занятом месте на диске и выделять/освобождать место по требованию;
  - ничего не напоминает?

Задача файловой системы

- Основная задача ФС сопоставить имени файла и смещению в файле смещение на диске
  - ФС должна поддерживать информацию о свободном/занятом месте на диске и выделять/освобождать место по требованию;
  - ничего не напоминает?
- ФС - это еще один алокатор, но со своей спецификой:
  - все запросы к диску тоже проходят через ФС;
  - мы не можем полагаться на надежность процессов;
  - выделенное место не непрерывно на диске;
  - от ФС часто ожидают надежного хранения;

Файловая система FAT

- FAT простая и очень популярная ФС появившаяся в 1977
  - Microsoft, NCR, SCP, IBM, Compaq, Digital Research, Novell;
  - FAT расшифровывается как File Allocation Table;
  - она неплохо работает при небольших объемах диска;
  - не предоставляет средств обеспечения надежности;

Разметка диска в FAT

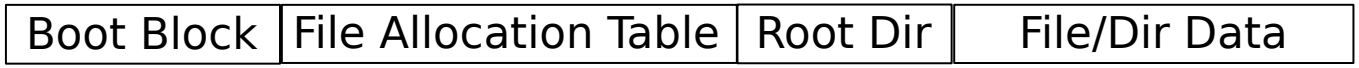

Figure : FAT Layout

- Boot Block - содержит общую информацию (параметры диска, размер блока ФС и прочее), размер фиксирован;
- File Allocation Table - таблица блоков файловой системы (по одно записи на каждый блок), размер определяется при форматировании;
- Root Dir - содержимое корневого каталога ФС;
- File/Dir Data - содержимое остальных каталогов и файлов;

File Allocation Table

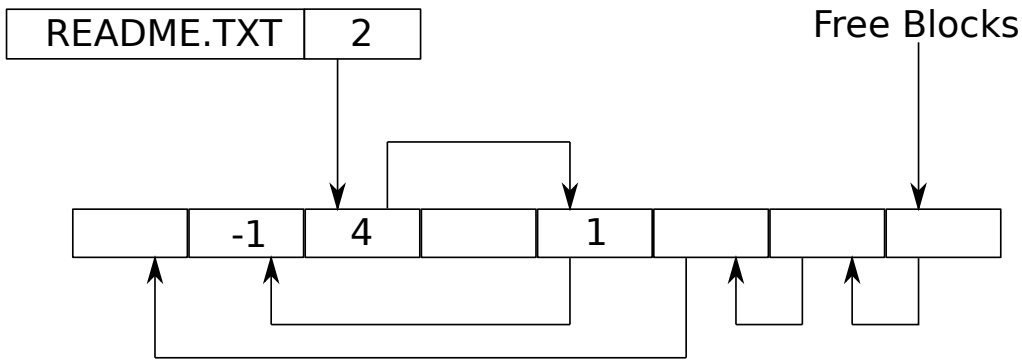

Figure : File Allocation Table

- Главная структура для FAT - File Allocation Table:
  - File Alocation Table хранит связные списки блоков файлов/каталогов;
  - File Allocation Table хранит номер следующего блока или маркер конца списка;
  - зная первый блок легко получить все остальные;

Запись каталога

- Каталог - просто набор записей;
- все записи в каталоге имеют одинаковую структуру:
  - 8 байт на имя + 3 байта на расширение;
  - различные атрибуты файла (права, размер в байтах);
  - номер первого блока файла;

Fast File System

- Fast File System - ФС разработанная для замены ФС созданной Кеном Томпсоном:
  - ФС от Томпсона даже не пыталась поддерживать локальность и бороться с фрагментацией;
  - ее производительность была ужасна и со временем становилась хуже;
  - группа исследователей в Berkely разработала ФС решающую эти проблемы в 1984;
- Основная идея Fast File System учитывать специфику дискового доступа:
  - располагать файлы последовательно;
  - избегать позиционирований головки диска;
  - Marshall Kirk McKusick, William N. Joy, Samuel J. Leffler, Robert S. Fabry "A Fast File System for UNIX"

Разметка диска в FFS

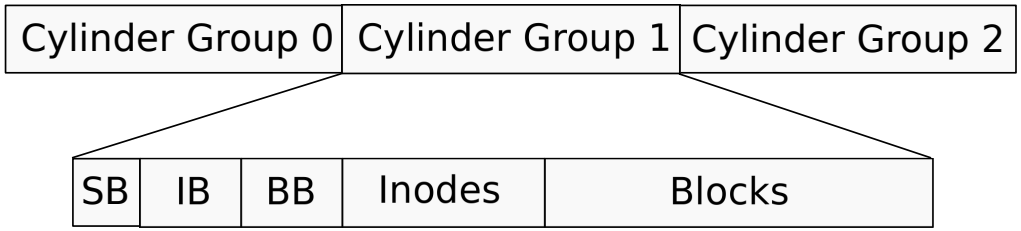

Figure : FFS Layout

- Все пространство разбито на цилиндровые группы для поддержки локальности;
- каждая группа содержит копию суперблока (SB) с параметрами ФС;
- IB и BB - битовая карта свободных/занятых inode и блоков внутри группы;
- inodes - таблица индексных узлов (Index Nodes);

Индексный узел

- Inode представляет сущности ФС (файлы/каталоги/специальные файлы):
  - inode хранит информацию о блоках файла;
  - inode хранит атрибуты файла;
  - inode не хранит имя файла - один и тот же inode может соответствовать нескольким именам;
- Количество индексных узлов определяется в момент форматирования;

Политика выделения ресурсов

- В FFS есть два ресурса, которые нужно выделять: Inode и блоки диска;
- ФС старается выделять inode-ы файлов одного каталога в одной группе цилиндров
  - команда ls требует обращения к inode, таким образом ускоряем ls;
- ФС распределяет inode-ы каталогов равномерно по группам;
- ФС старается выделять блоки одного файла в одной группе
  - для больших файлов стараемся выделять большие порции файла в одной группе;
  - порции файлов распределяются по разным группам;

Блоки файла

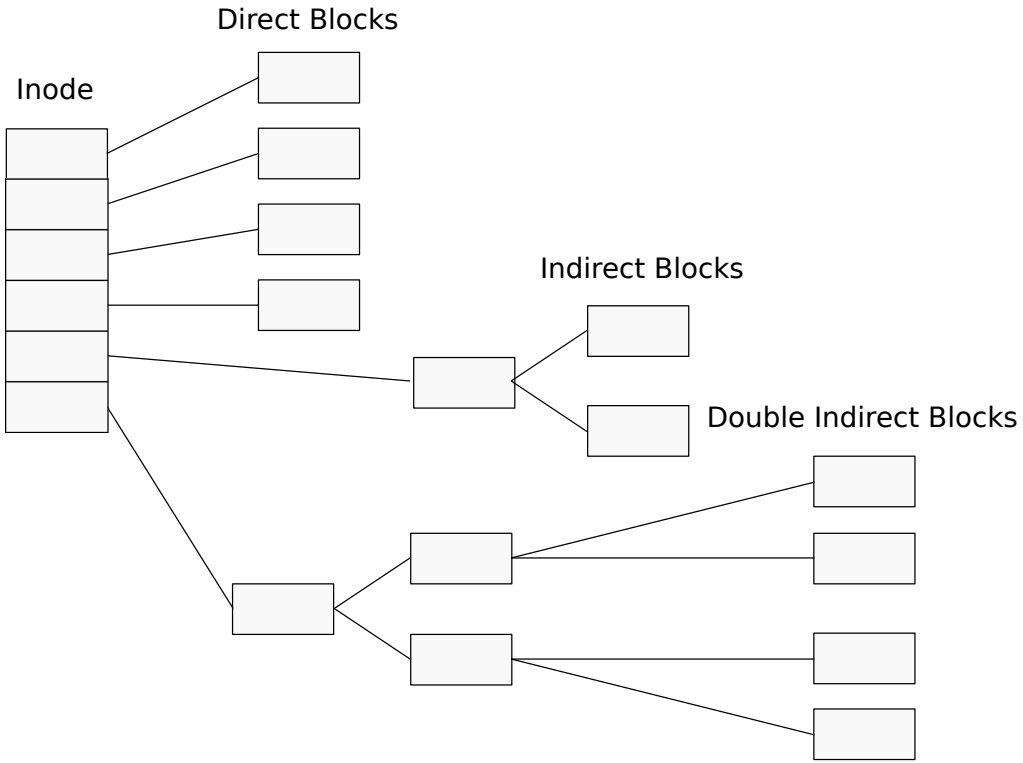

Figure : FFS Inode Blocks

Индексные структуры данных

- ФС отображает различные ключи на значения (возможно пустые):
  - имя файла и inode родительского каталога на inode файла;
  - inode файла/каталога и смещение на смещение на диске;
  - поддерживает множество свободных блоков диска;

Индексные структуры данных

- ФС отображает различные ключи на значения (возможно пустые):
  - имя файла и inode родительского каталога на inode файла;
  - inode файла/каталога и смещение на смещение на диске;
  - поддерживает множество свободных блоков диска;
- Все что нам нужно - правильная индексная структура данных:
  - предназначенная для работы с диском;
  - по возможности защищенная от ошибок;
  - по возможности concurrent-friendly;

B-деревья

- B-деревья - сильно ветвистые идеально сбалансированные деревья поиска;
  - каждый узел (кроме возможно корня) хранит от B до $2 \times B$ ключей, где B параметр дерева;
  - B обычно определяется размером ключа и размером блока ФС;

B-деревья

- B-деревья - сильно ветвистые идеально сбалансированные деревья поиска;
  - каждый узел (кроме возможно корня) хранит от B до $2 \times B$ ключей, где B параметр дерева;
  - B обычно определяется размером ключа и размером блока ФС;
- На практике чаще используются B+-деревья:
  - внутренние узлы хранят только ключи и указатели на узлы;
  - значения хранятся только в листьях;
  - листья дерева могут быть связаны в список;

Синхронизация B-деревьев

- У всех деревьев поиска (включая B-дереьвя) есть он

Синхронизация B-деревьев

- У всех деревьев поиска (включая B-дереьвя) есть он - корень:

Синхронизация B-деревьев

- У всех деревьев поиска (включая B-дереьвя) есть он - корень:
  - все операции начинаются от корня, как поиск, так и вставка и удаление;
  - даже если корень меняется редко, вам все равно нужна синхронизация в корне;
  - деревья поиска - очень не concurrent-friendly;

Синхронизация B-деревьев

- У всех деревьев поиска (включая B-дереьвя) есть он - корень:
  - все операции начинаются от корня, как поиск, так и вставка и удаление;
  - даже если корень меняется редко, вам все равно нужна синхронизация в корне;
  - деревья поиска - очень не concurrent-friendly;
- Леман и Яо предложили довольно эффективную схему синхронизации B-деревьев:
  - читателям не требуется никакая синхронизация (в модели оригинальной статьи);
  - писатели в каждый момент времени держат константное количество блокировок;
  - оригинальная статья: Efficient Locking for Concurrent Operations on B-Trees

Синхронизация B-деревьев

- Нам потребуется изменить структуру B-дерева немного:
  - все узлы хранят ссылку на следующий узел того же уровня;
  - после разделения узел и сосед работают "как один" узел благодаря ссылке;
  - почти на все узлы есть две ссылки: ссылка из родителя и ссылка из соседа;
  - ссылка из соседа на новый узел всегда устанавливается первой, а затем мы будем изменять родителя;

Синхронизация B-деревьев

- Поиск ключа в дереве:
  - мы выполняем обычный поиск по дереву поиска, до тех пор пока не найдем нужную пару ключ/значение;
  - во время поиска мы могли прийти в узел, максимальный ключ которого меньше искомого;
    - это могло произойти в результате разделения узла (не успели обновить родителя);
    - просто воспользуемся ссылкой на соседа и заглянем в соседний узел;

Синхронизация B-деревьев

- Вставка узла в дерево:
  - Начинаем с поиска позиции для вставки
    - после того как мы нашли нужный лист и перед тем как мы его запишем кто-то может его обновить;
    - поэтому нужно захватить блокировку на узле и двигаться по правым ссылкам уже с блокировками;
  - Мы нашли и заблокировали нужный узел - теперь нужно вставить ключ
    - вставка может потребовать разделения - сначала нужно записать правый узел;
    - потом нужно записать левый узел с правильной ссылкой на соседа;
  - После разделения нужно обновить родителя, а для этого нужно его найти
    - опять же проходом по правым ссылкам;

LSM дерево

- Обычные B-деревья (и их вариации) обладают рядом недостатков:
  - расположение на диске - узлы дерева разбросаны как попало;
  - concurrent-unfriendliness - требуются нетривиальные схемы синхронизации, вступающие в противоречия с другими требованиями;

LSM дерево

- Обычные B-деревья (и их вариации) обладают рядом недостатков:
  - расположение на диске - узлы дерева разбросаны как попало;
  - concurrent-unfriendliness - требуются нетривиальные схемы синхронизации, вступающие в противоречия с другими требованиями;
- Проблемы B-деревьев возникают из-за удаления и вставки:
  - без операций вставки и удаления деревья становятся очень простыми (очень пустыми);
  - мы можем заменить операции вставки и удаления операцией слияния;

LSM дерево

- LSM (Log-Structured Merge) дерево состоит из нескольких деревьев:
  - C_0 - дерево в памяти, единственное дерево в которое можно вставлять или удалять;
  - C_1 - C_k - набор неизменяемых деревьев на диске (например, B+-деревьев);

LSM дерево

- LSM (Log-Structured Merge) дерево состоит из нескольких деревьев:
  - C_0 - дерево в памяти, единственное дерево в которое можно вставлять или удалять;
  - C_1 - C_k - набор неизменяемых деревьев на диске (например, B+-деревьев);
- Операция слияния сливает два соседних дерева в одно новое:
  - в результате слияния C_1 и C_2 получим два дерева: C_1' - пустое, и C_2' - дерево объединения;
  - слияние - просто проход по двум упорядоченным деревьям с выписыванием значений;

LSM дерево

- Для удаления ключа из LSM дерева используем вставку и слияние:
  - вставьте ключ со специальным маркером означающем удаленный узел;
  - при слиянии деревьев удаляйте отбрасывайте этот ключ в старых деревьях;
  - при слиянии в последнее дерево, просто отбросим ключ с маркером;

LSM дерево

- Для удаления ключа из LSM дерева используем вставку и слияние:
  - вставьте ключ со специальным маркером означающем удаленный узел;
  - при слиянии деревьев удаляйте отбрасывайте этот ключ в старых деревьях;
  - при слиянии в последнее дерево, просто отбросим ключ с маркером;
- Непосредственные модификации происходят только над C_0, который находится в памяти
  - слияние, очевидно, не требует модификации, т. е. все деревья на диске неизменны;
  - после объединения деревьев нужно просто поправить указатель на корень;

LSM дерево

- Ключевой вопрос - когда нужно выполнять слияние?
  - обычно, слияние определяется размером дерева - сливаем дерево со следующим, если оно стало большим;
  - соответственно, размеры деревьев растут с индексом дерева;

LSM дерево

- Ключевой вопрос - когда нужно выполнять слияние?
  - обычно, слияние определяется размером дерева - сливаем дерево со следующим, если оно стало большим;
  - соответственно, размеры деревьев растут с индексом дерева;
- Какими должны быть размеры деревьев?
  - если размеры деревьев образуют геометрическую прогрессию вы получите амортизировано $O(1)$ на слияние;
  - количество деревьев, обычно, фиксировано, а последнее дерево растет неограниченно;

Надежность файловых систем

- От файловых систем часто ожидают надежного хранения:
  - толерантность к ошибкам диска;
  - возможность переживать fail-stop-ы (например, сбои питания);

Надежность файловых систем

- От файловых систем часто ожидают надежного хранения:
  - толерантность к ошибкам диска;
  - возможность переживать fail-stop-ы (например, сбои питания);
- Решение первой проблемы - дублирование
  - и возможно детектировать ошибки, без этого дублирование не работает;
  - мы не будем останавливаться на этом подробнее;

Надежность файловых систем

- От файловых систем часто ожидают надежного хранения:
  - толерантность к ошибкам диска;
  - возможность переживать fail-stop-ы (например, сбои питания);
- Решение первой проблемы - дублирование
  - и возможно детектировать ошибки, без этого дублирование не работает;
  - мы не будем останавливаться на этом подробнее;
- Вторая проблема имеет несколько "решений":
  - fsck;
  - журналирование;
  - COW;

Пример

- Рассмотрим как происходит создание файла в ФС:
  - необходимо выделить место под данные (если он есть);
  - необходимо выделить inode (например, выставить бит в битовой карте);
  - добавить имя файла в родительский каталог;

Пример

- Рассмотрим как происходит создание файла в ФС:
  - необходимо выделить место под данные (если он есть);
  - необходимо выделить inode (например, выставить бит в битовой карте);
  - добавить имя файла в родительский каталог;
- В зависимости от порядка записи в случае fail-stop-а может случиться плохое или очень плохое:
  - очень плохое: inode хранящий мусор и ведущий в никуда;
  - плохое: inode занят, но не доступен (утечка);

fsck

- fsck - утилита, сканирующая ФС в поисках ошибок и исправляющая их;
  - например, мы можем легко найти и исправить следующие ошибки:
    - неправильный супер блок (boot block);
    - утечки inode и блоков диска;

fsck

- fsck - утилита, сканирующая ФС в поисках ошибок и исправляющая их;
  - например, мы можем легко найти и исправить следующие ошибки:
    - неправильный супер блок (boot block);
    - утечки inode и блоков диска;
- fsck - прямолинейный и неэффективный способ:
  - fsck - сканирует всю ФС, что не быстро;
  - некоторые ошибки не легко исправить, даже если вы их нашли;
  - из личного опыта - однажды fsck спас меня там, где не смогли справиться более продвинутые методы;

Журналирование

- Журналирование (write-ahead logging) - логирование операций перед их выполнением
  - перед выполнением операции мы записываем ее описание в специальное место диска - журнал;
  - после выполнения операции мы удаляем запись из журнала (помечаем как выполненную);
  - после fail-stop нужно "проиграть" записи из журнала;

Журналирование

- Журналирование (write-ahead logging) - логирование операций перед их выполнением
  - перед выполнением операции мы записываем ее описание в специальное место диска - журнал;
  - после выполнения операции мы удаляем запись из журнала (помечаем как выполненную);
  - после fail-stop нужно "проиграть" записи из журнала;
- Записи в журнале должны быть идемпотентными:
  - например, запись "выделить inode" не идемпотентна;
  - запись "записать в блок B данные X" идемпотентна;
  - т. е. запись нужно уметь проиграть несколько раз;

Журналирование

- Фактически журналирование приводит к дублирование каждой записи
  - зачастую журналируются только обновления метаданных, но не данных;
  - запись в журнал всегда происходит последовательно;

Журналирование

- Фактически журналирование приводит к дублирование каждой записи
  - зачастую журналируются только обновления метаданных, но не данных;
  - запись в журнал всегда происходит последовательно;
- Журнал - маленькая fail-stop безопасная ФС:
  - мы можем записать что-то в журнал, а потом прочитать (не без сложностей);
  - а не можем ли мы всю файловую систему построить как журнал?

COW

- Copy On Write - никогда не перезаписывать данные:
  - вместо перезаписи на месте мы можем создать копию и обновить ее;
  - нужно атомарно обновить ссылку с оригинала на обновленную копию;
  - запись одного блока диска атомарна;

COW

- Copy On Write - никогда не перезаписывать данные:
  - вместо перезаписи на месте мы можем создать копию и обновить ее;
  - нужно атомарно обновить ссылку с оригинала на обновленную копию;
  - запись одного блока диска атомарна;
- Деревья дружественная, обычно, к COW структура данных:
  - используя проактивные разделения и слияния узлов сравнительно легко реализовать COW B+-дерево;
  - нужно копировать только путь от корня, до измененного листа;
  - подмена оригинала копией - подмена указателя на корень;
  - Btrfs - ФС построенная вокруг COW B+-деревьев;