Обзорная лекция к госэкзамену ФИИТ

теория вычислений, функциональное программирование, операционные системы

В.Н. Брагилевский

Южный федеральный университет
Институт математики, механики и компьютерных наук им. И. И. Воровича
Направление 02.03.02 — «Фундаментальная информатика и информационные технологии»

3 июня 2015 г.

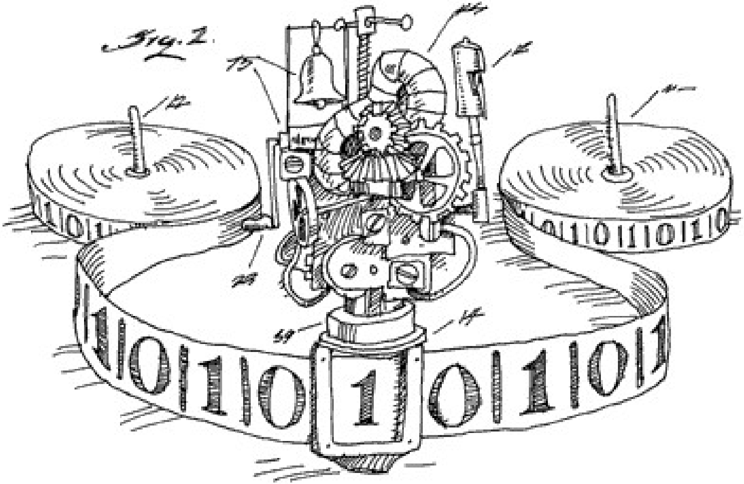
Содержание

Содержание

Модели вычислений

- Модели вычислений служат в качестве формальных определений для понятия вычисления.
- Существует множество эквивалентных друг другу моделей вычислений.
- Примеры: машины Тьюринга, λ-исчисление, рекурсивные функции, нормальные алгорифмы.

Машина Тьюринга



Машина Тьюринга

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

- Q — конечное множество состояний;
- Σ — входной алфавит;
- Γ — ленточный алфавит ($\Sigma \subset \Gamma$);
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ — функция переходов;
- $q_0 (\in Q)$ — начальное состояние;
- $B (\in \Gamma)$ — пробельный символ;
- F — множество допускающих состояний ($F \subset Q$).

Языком машины Тьюринга называется множество всех допускаемых ей слов (то есть слов, на которых она попадает в допускающее состояние).

$$L(M) = \{w \in \Sigma^* : q_0 w \vdash_M^* w' q_i w'', q_i \in F, w', w'' \in \Gamma^*\}$$

Недетерминированная машина Тьюринга

НМТ может находиться одновременно в нескольких состояниях с различным содержимым ленты:

- $\delta : Q \times \Gamma \to \{Q \times \Gamma \times \{L, R\}\}$ — функция переходов;
- НМТ допускает, если хотя бы по одной из ветвей приходит в допускающее состояние.

λ-исчисление

Пусть задано счётное множество переменных x_1, x_2, x_3, ...
Множество выражений, называемых λ-термами, определяется с помощью следующих трёх правил:

- Все переменные являются λ-термами.
- Если M и N — произвольные λ-термы, то (MN) — λ-терм, называемый применением (application).
- Если x — переменная, а M — произвольный λ-терм, то $(\lambda x.M)$ — λ-терм, называемый абстракцией (abstraction).

Вычисление в λ-исчисление

Редексы

Термы вида $(\lambda x.M)N$ называются *редексами* или *редуцируемыми выражениями* (reducible expression).

Подстановка

Выражение $[N/x]M$ означает подстановку терма N вместо переменной x в терм M.

β-редукция

Отношение, которое ставит в соответствие редексу $(\lambda x.M)N$ терм $[N/x]M$ называется *β-редукцией*. Если терм P содержит в качестве своего фрагмента редекс $(\lambda x.M)N$, причём после замены этого редекса на $[N/x]M$ получается терм P', то пишут, что

$$P \to_\beta P'.$$

Рекурсивные функции

Базовые функции

К базовым функциям относятся следующие три группы функций:

1. функция следования $S : \mathbb{N} \to \mathbb{N}$, действующая по правилу: $x \mapsto x + 1$;
2. семейство нуль-функций $C_0^k = C_0^k(x_1, \ldots, x_k) = 0$, $k \geqslant 0$ (это функции-константы, они всегда возвращают ноль независимо от значений аргументов, значение C_0^0 аргументов не имеет);
3. семейство функций-проекторов $P_j^k = P_j^k(x_1, \ldots, x_k) = x_j$, $k \geqslant 1$, $0 < j \leqslant k$ (эти функции возвращают значение аргумента, номер которого совпадает с нижним индексом, например, функция P_1^1 это просто тождественная функция).

Оператор композиции

Пусть заданы функции $f : \mathbb{N}^m \to \mathbb{N}$, $g_1, g_2, \ldots, g_m : \mathbb{N}^k \to \mathbb{N}$. Говорят, что функция $h : \mathbb{N}^k \to \mathbb{N}$ получена из функций $f, g_1, g_2, \ldots, g_m$ с помощью оператора композиции, если её значение может быть вычислено по правилу:

$$h(x_1, \ldots, x_k) = f(g_1(x_1, \ldots, x_k), g_2(x_1, \ldots, x_k), \ldots, g_m(x_1, \ldots, x_k)).$$

Оператор примитивной рекурсии

Пусть заданы две функции: $f : \mathbb{N}^k \to \mathbb{N}$ и $g : \mathbb{N}^{k+2} \to \mathbb{N}$. Говорят, что функция $h : \mathbb{N}^{k+1} \to \mathbb{N}$ получена из функций f и g применением оператора примитивной рекурсии, если её значение может быть вычислено по правилу (схема примитивной рекурсии):

$$\begin{aligned} h(x_1, \ldots, x_k, 0) &= f(x_1, \ldots, x_k); \\ h(x_1, \ldots, x_k, y+1) &= g(x_1, \ldots, x_k, y, h(x_1, \ldots, x_k, y)). \end{aligned}$$

Оператор минимизации

Пусть задана функция $f : \mathbb{N}^{k+1} \to \mathbb{N}$. Говорят, что функция $h : \mathbb{N}^k \to \mathbb{N}$ получена из функции f применением оператора минимизации, если её значение может быть вычислено по правилу:

$$h(x_1, \ldots, x_k) = \mu y[f(x_1, \ldots, x_k, y) = 0],$$

то есть как наименьшее такое y, для которого значение $f(x_1, \ldots, x_k, y)$ равно 0.

Тезис Чёрча—Тьюринга

В настоящее время принято считать, что имеет место *тезис Чёрча—Тьюринга*: *любая вычислимая функция может быть вычислена с помощью соответствующей машины Тьюринга или любой другой эквивалентной ей модели*.

Это утверждение не может быть доказано, поскольку в нём участвует неопределяемое понятие «вычислимой функции», под которым мы понимаем нечто, что может быть принципиально вычислено, какие бы методы при этом ни использовались.

Классификация языков с точки зрения вычислимости

- Рекурсивные (разрешимые) языки — существуют распознающие их всегда останавливающиеся машины Тьюринга.
- Рекурсивно-перечислимые языки — машины Тьюринга, их распознающие, могут не останавливаться на словах, не принадлежащих этим языкам (например, универсальный язык).
- Неперечислимые языки — не существует машин, их распознающих (например, язык диагонализации — язык несамоприменимых машин Тьюринга).

Примеры неразрешимых задач

- Проблема останова.
- Универсальный язык.
- Распознавание свойств рекурсивно-перечислимых языков (теорема Райса).
- Проблема соответствий Поста.

Содержание

Сложность вычислений

Ресурсы машины Тьюринга

- Время (зависимость необходимого числа шагов от длины входного слова).
- Память (зависимость длины используемого фрагмента ленты от длины входного слова).

Виды зависимостей

- Полиномиальная.
- Экспоненциальная.

Классы P и NP

- Класс P — задачи, решаемые некоторой детерминированной машиной Тьюринга за полиномиальное время.
- Класс NP — задачи, решаемые некоторой недетерминированной машиной Тьюринга за полиномиальное время.
- Ясно, что $P \subset NP$, но неизвестно, является ли вложение строгим.

Сводимость и NP-полнота

Сводимость по Карпу (для задач разрешения)

Задача P_1 сводится к P_2, если существует работающий за полиномиальное время алгоритм, который преобразует экземпляр задачи P_1 к экземпляру задачи P_2 с тем же ответом.

NP-полные задачи

NP-полными называются такие задачи из класса NP, к которым сводятся любые другие задачи из класса NP.

Утверждение

Если некоторую NP-полную задачу можно решить за полиномиальное время, то P=NP.

Примеры NP-полных задач

- SAT (проверка выполнимости булевой формулы), CSAT, 3SAT.
- Задача о существовании гамильтонова цикла в ориентированном и неориентированном графах.
- Задача коммивояжёра.

Приёмы доказательства NP-полноты

- Прямое доказательство (сведение произвольной машины Тьюринга).
- Построение сведения уже известной NP-полной задачи к данной.

Пространственная сложность

- Класс PSPACE — класс задач, разрешимых в полиномиальном пространстве.
- У машин Тьюринга для задач из класса PSPACE существует не более чем экспоненциальное количество различных конфигураций, поэтому для их решения достаточно экспоненциального времени (достаточно перебрать все конфигурации и можно останавливаться без допускания, поскольку повтор конфигурации означает зацикливание).
- Теорема Сэвитча: PSPACE=NPSPACE (недетерминированная машина Тьюринга может быть смоделирована детерминированной, работающей с не более чем квадратичным объёмом памяти по сравнению с исходной).

Содержание

Основные принципы ФП

- Вычислительный процесс — это вычисление значения функции по заданным аргументам.
- В вычислении отсутствует состояние (состояние неизменяемо).
- Отсутствие присваиваний и циклов (цикл предполагает изменение состояния во время итераций).
- Основная единица программы — функция, основная операция — вызов функции (применение функции к аргументам).
- Функции являются «чистыми» — значение функции зависит исключительно от значений её аргументов, повторный вызов с теми же значениями гарантированно возвращает тот же результат.

Списки и рекурсия

- Рекурсия — вызов функцией самой себя с другими значениями аргументов.
- Список — рекурсивная структура данных (состоит из головы и хвоста, который в свою очередь также является списком).
- База рекурсии — пустой список.

Функции высших порядков

Определение

Функция высшего порядка (*higher-order function*) — функция, принимающая в качестве аргументов функции или возвращающая функцию в качестве результата.

Важные примеры ФВП

- map — преобразование каждого элемента списка заданной функцией;
- filter — выбор элементов списка, удовлетворяющих предикату;
- fold (reduce) — свёртка списка (вычисление значения на основе прохода по списку и накопления результата некоторой функцией);
- unfold — развёртка списка (построение списка на основе некоторого значения и правила построения следующего элемента).

Алгебраические типы данных

- Типы-перечисления (суммы): элементом является одно из перечисленных значений:
  Boolean = True + False
- Типы-контейнеры (произведения): элементом является набор значений:
  Person = Name * Age * Weight
- Общий случай (например, контейнер, полями которого являются перечисления).
- Параметризованные типы — имеется типовый параметр, от которого зависят реальные значения:
  Vector3D t = t * t * t
- Рекурсивные типы — элемент может содержать в качестве своего компонента элемент того же типа, например, бинарное дерево:
  BinTree = EmptyTree + BinTree * BinTree

Содержание

Содержание

Понятие процесса

- Процесс как программа, запущенная на выполнение.
- Процесс как единица работы (задача) и как единица управления ресурсами.
- Многозадачность на одном процессоре и переключение контекста.

Понятие потока

- Процесс как единица управления ресурсами — поток как единица работы.
- Thread (fiber), multithreading.
- Разделяемые потоками данные.
- Данные, специфичные для потока.
- Многоядерность как причина распространения многопоточного программирования.
- Потоки в языках программирования реализуются либо языковыми конструкциями, либо библиотечными функциями.
- Потоки в системе реализуются либо на уровне ядра, либо в библиотеках пользовательского уровня, либо и там, и там.

Взгляд на потоки и процессы в Windows и Linux

- Linux: многопоточный процесс — это группа процессов с общими данными.
- Windows: поток — это самостоятельная сущность, единица планирования.

Функции ОС по управлению процессами и потоками

- чередование на процессоре (планирование);
- распределение ресурсов (процессорное время, память, доступ к устройствам);
- поддержка межпроцессного взаимодействия и синхронизации.

Операции над потоками

- Создание (create) и запуск (start).
- Аннулирование (cancel) и остановка (stop).
- Присоединение потока (join).
- Уступка (yield).

Содержание

Состояние гонок (race conditions)

```
x := x + 1;
```

```
LOAD x
STORE x, x + 1
```

x *— разделяемая (shared) переменная*

Последовательное выполнение

Процесс 1	*Процесс 2*
	x = 0
`LOAD, x`	
`STORE x, x + 1`	
	`LOAD x`
	`STORE x, x + 1`
	x = 2

Состояние гонок (race conditions)

Выполнение с переключением контекста

Процесс 1	*Процесс 2*
x = 0	
`LOAD, x`	
	`LOAD x`
`STORE x, x + 1`	
	`STORE x, x + 1`
x = 1	

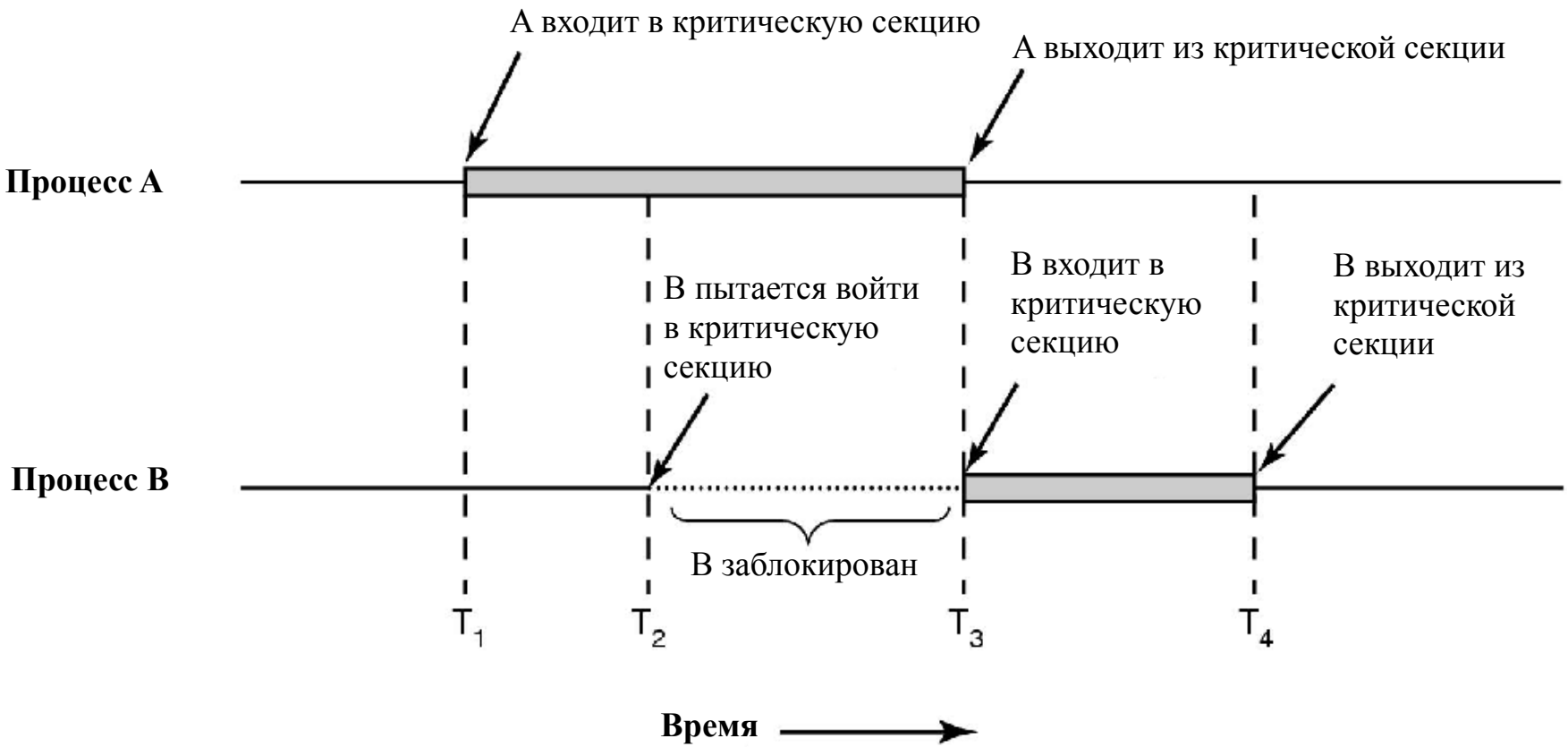
Взаимное исключение

mutual exclusion
critical section (critical region)

Общая схема процесса:

```
while (true)
{
  enter_critical_section();
  /* критическая секция */
  exit_critical_section();
  /* операторы вне критической секции */
}
```

Взаимное исключение



Примитивы синхронизации

1) Семафоры (Дейкстра) и мьютексы.
2) Условные переменные.
3) Мониторы (Хоар–Хансен, Лэмпсон–Ределл).

Семафоры

```
struct semaphore {
  int count;
  queue q;
}
```

down

up

процесс P:

```
down(s) {
  s.count--;
  if (s.count < 0){
    P блокируется
    P → s.q
  }
}
```

любой другой процесс:

```
up(s) {
  s.count++;
  if (s.count <= 0) {
    s.q → P
    P освобождается
  }
}
```

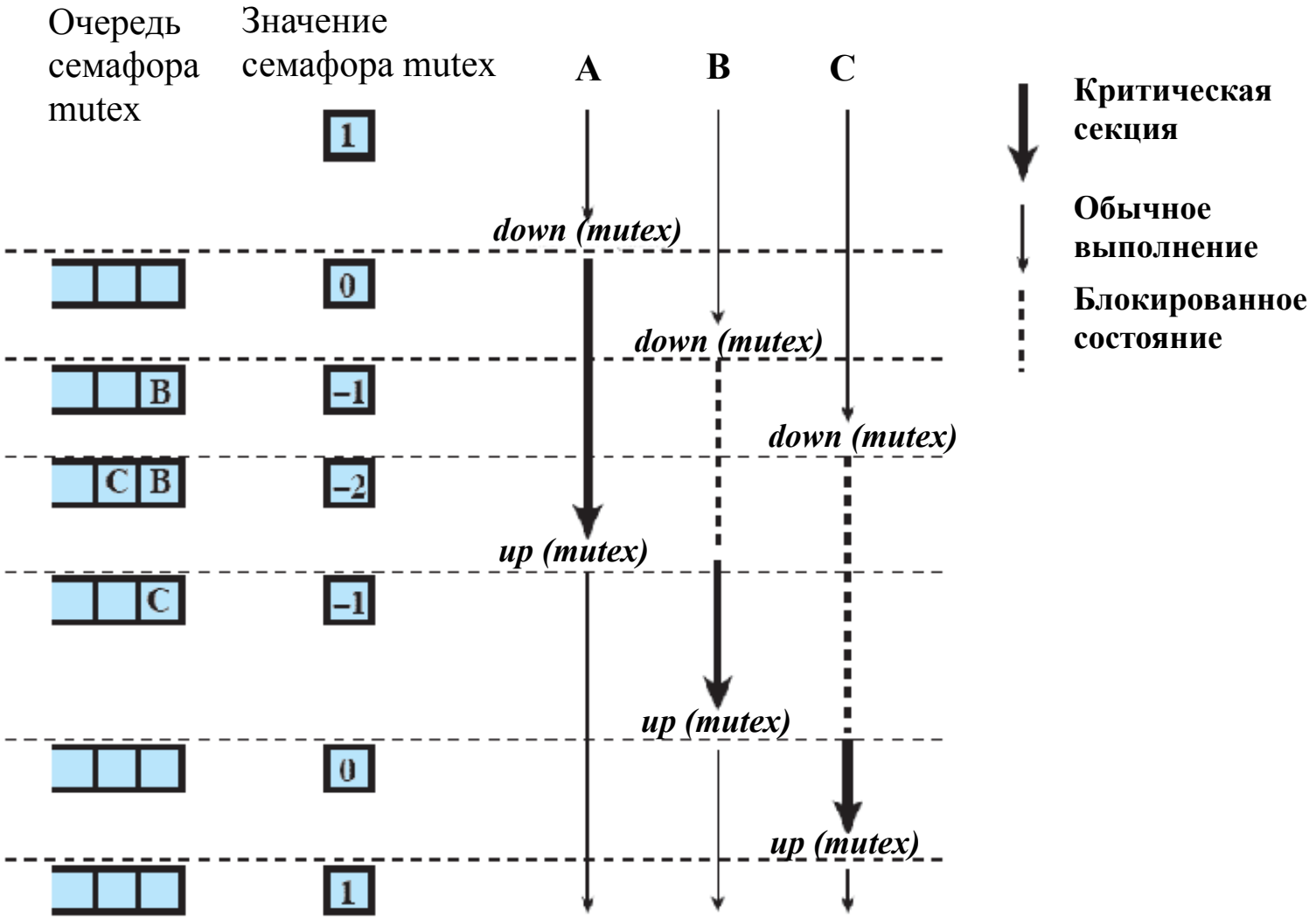
Семафоры и взаимное исключение

```
// Разделяемая переменная — мьютекс
semaphore mutex = 1;

void process (int i)
{
  while(true)
  {
    down(mutex);
    / * критическая секция */
    up(mutex);
    /* остальные операторы */
  }
}
```

Операции над мьютексом (альтернативная формулировка):

- захват блокировки (down=lock);
- освобождение блокировки (up=unlock).

Семафоры и взаимное исключение



Условные переменные

- Операции:
  - ожидание условия (wait);
  - информирование об изменении условия (signal).
- Задача об ограниченном буфере:
  - два потока: производитель и потребитель;
  - чтение и запись в критической секции;
  - ситуация пустого буфера;
  - ситуация полного буфера.

Условные переменные

```
semaphore mutex = 1;
condition cond;
```

wait
signal
signalAll

потребитель (чтение):

```
down(mutex);
while(буфер пуст)
  wait(cond, mutex);
// чтение
if(освободилось место)
  signal(cond);
up(mutex);
```

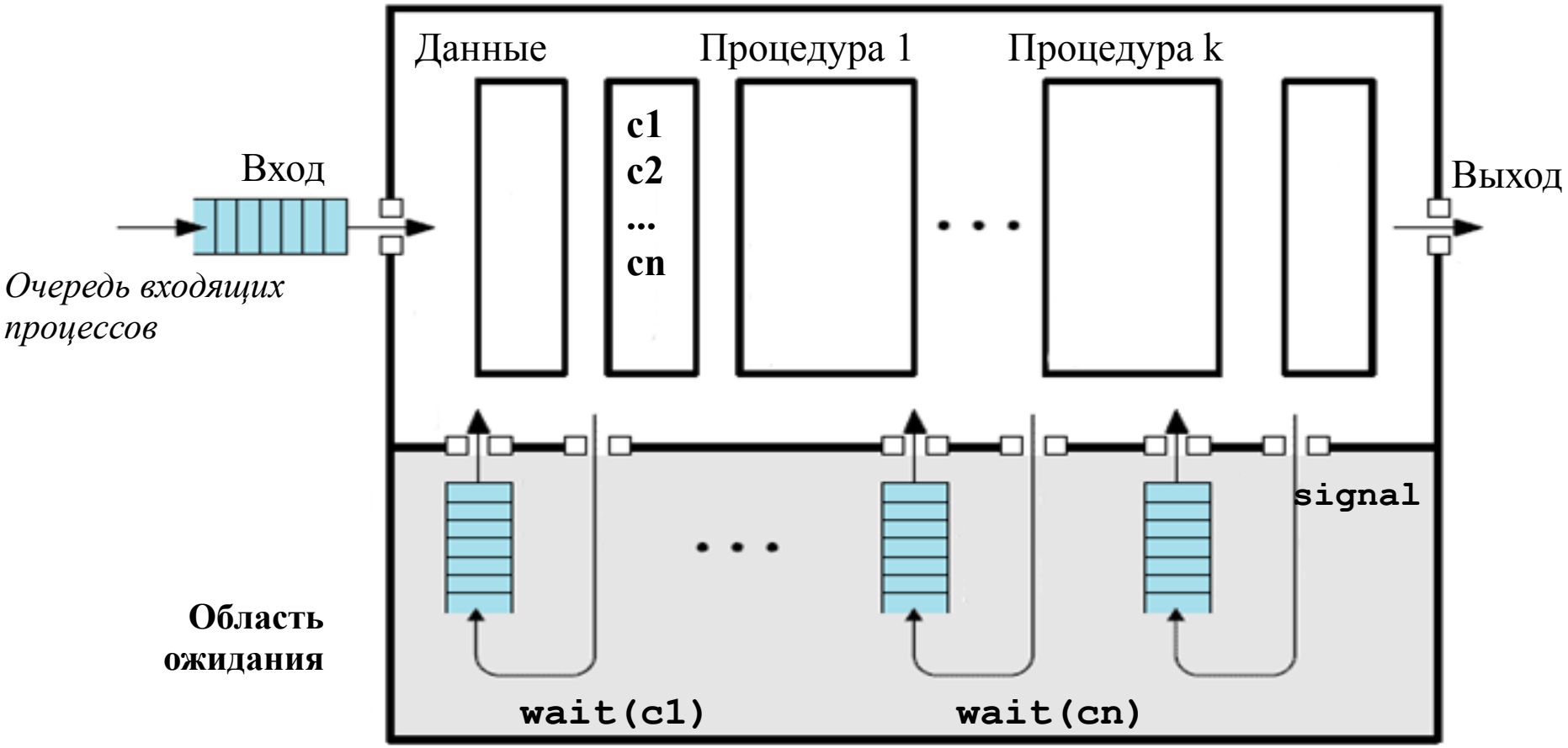
производитель (запись):

```
down(mutex);
while(буфер полон)
  wait(cond, mutex);
// запись
if(появились данные)
  signal(cond);
up(mutex);
```

Мониторы Хоара—Хансена

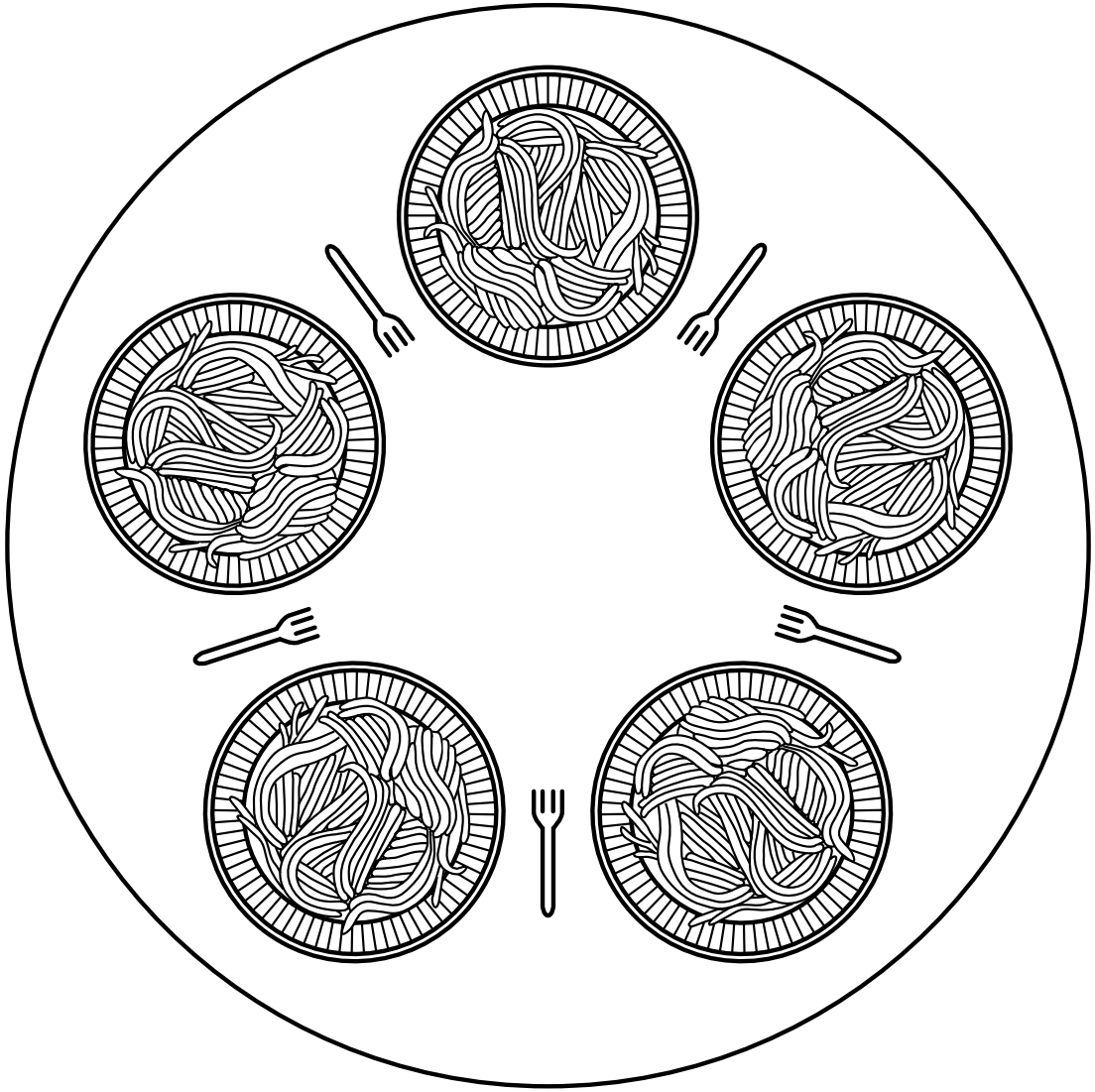
```
monitor example;
  data: DataType;

  c1, c2, ... : condition;

  procedure proc1;
  begin
    ...
  end;

  procedure proc2;
  begin
    ...
  end;
end monitor;
```

Структура монитора



Содержание

Обед философов



Неверное решение

```
const int N = 5;

void philosopher(int i) {
  while (true) {
    think();
    take_fork( LEFT(i) );
    take_fork( RIGHT(i) );
    eat();
    put_fork( LEFT(i) );
    put_fork( RIGHT(i) );
  }
}
```

Неэффективное решение

```
const int N = 5;
mutex table;

void philosopher(int i) {
  while (true) {
    think();
    lock(table);
    take_fork( LEFT(i) );
    take_fork( RIGHT(i) );
    eat();
    put_fork( LEFT(i) );
    put_fork( RIGHT(i) );
    unlock(table);
  }
}
```

Известные подходы к решению и их проблемы

Подходы к решению

- Ограничение числа философов в комнате.
- Асимметричность философов.
- Атомарное взятие обеих вилок и откаты.

Проблемы

- Отсутствие голодания и живые блокировки.
- Эффективность.
- Доказательство корректности.

Содержание

Содержание

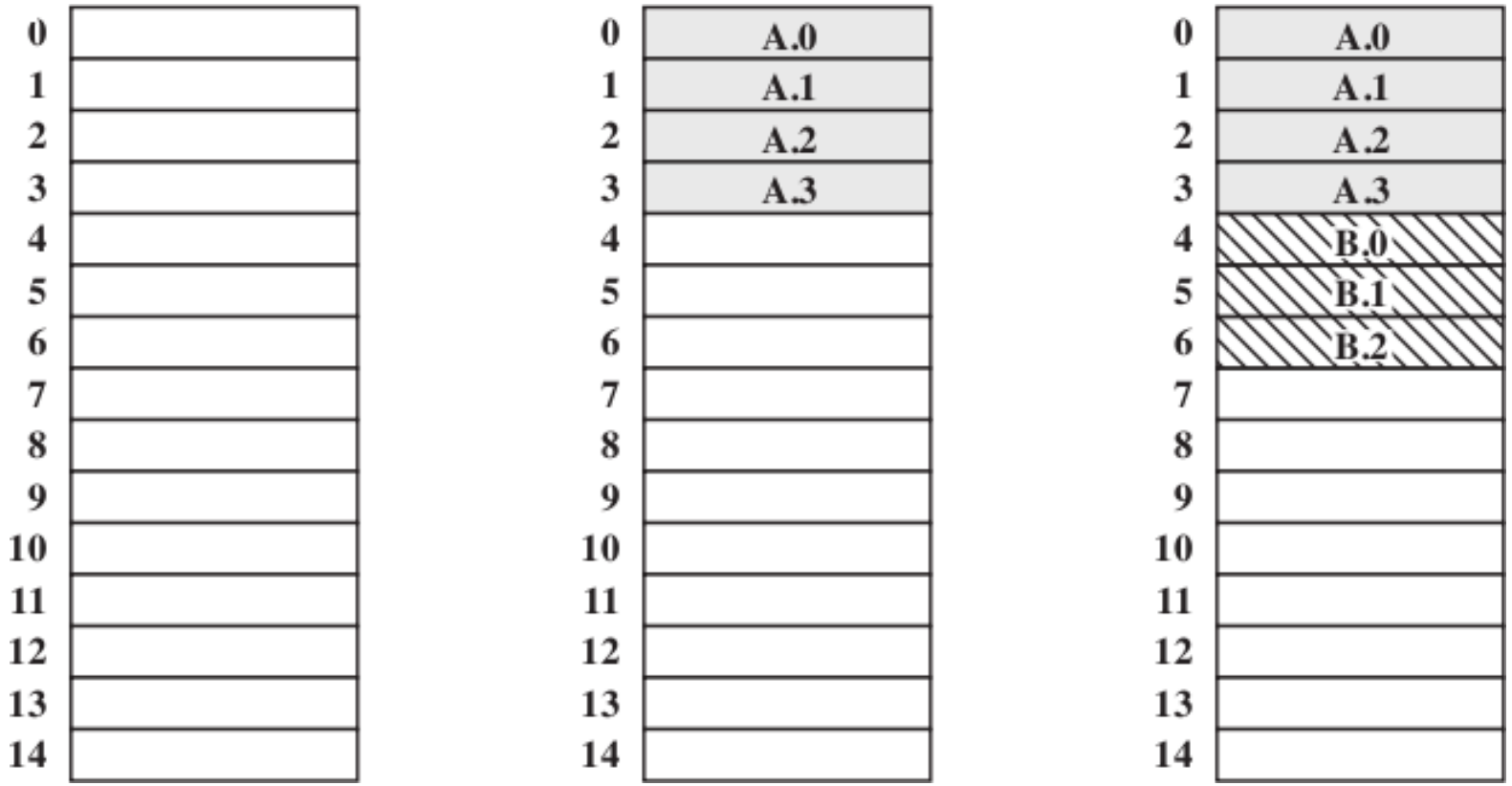
Модели памяти

- Плоская.
- Страничная.
- Сегментная.

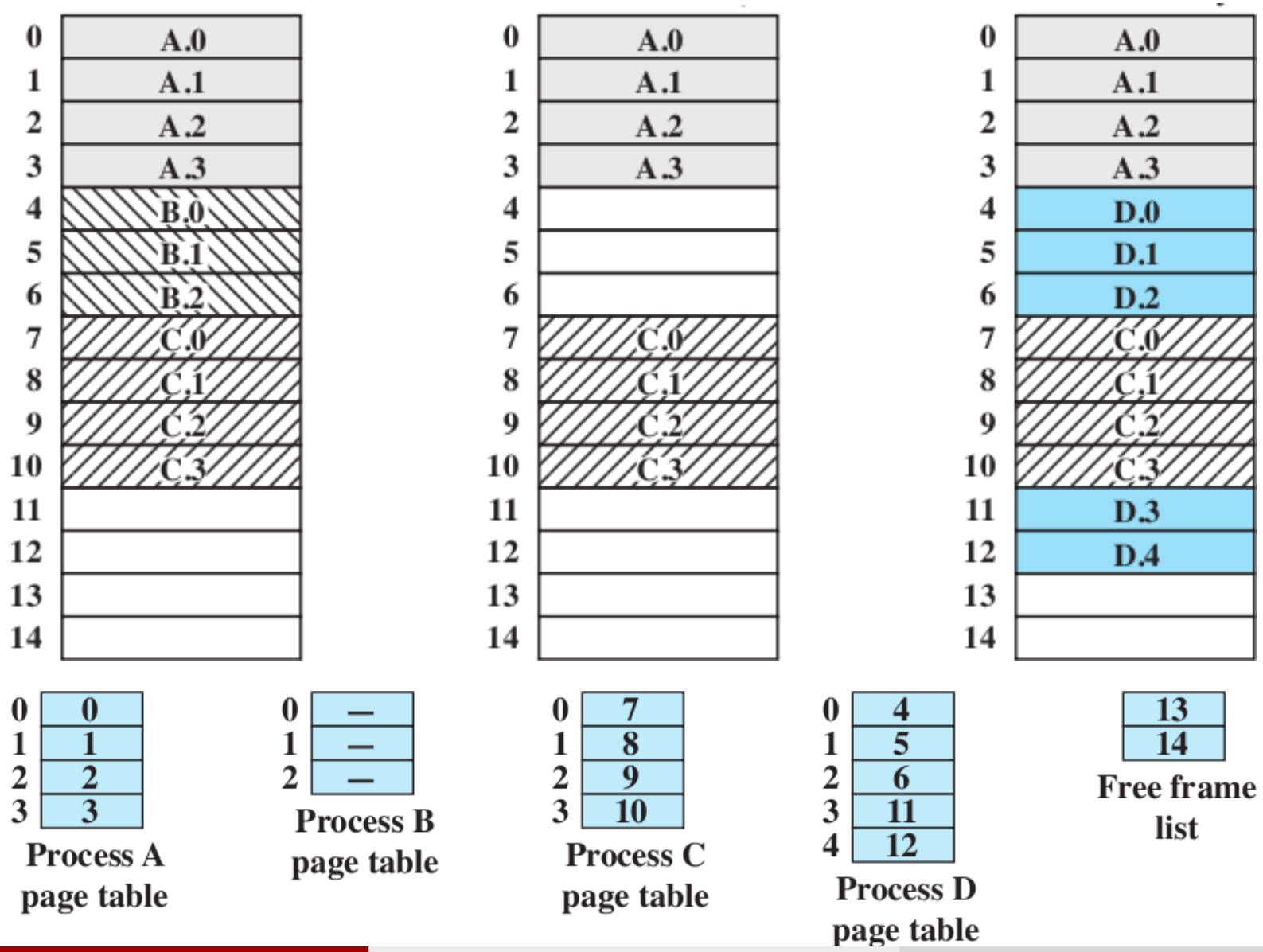
Страничная организация памяти

- Одинаковые блоки небольшого размера.
- Память: кадры (frames).
- Адресное пространство процесса: страницы (pages).
- Таблица страниц: номер страницы $\mapsto$ номер кадра.

Распределение страниц по кадрам



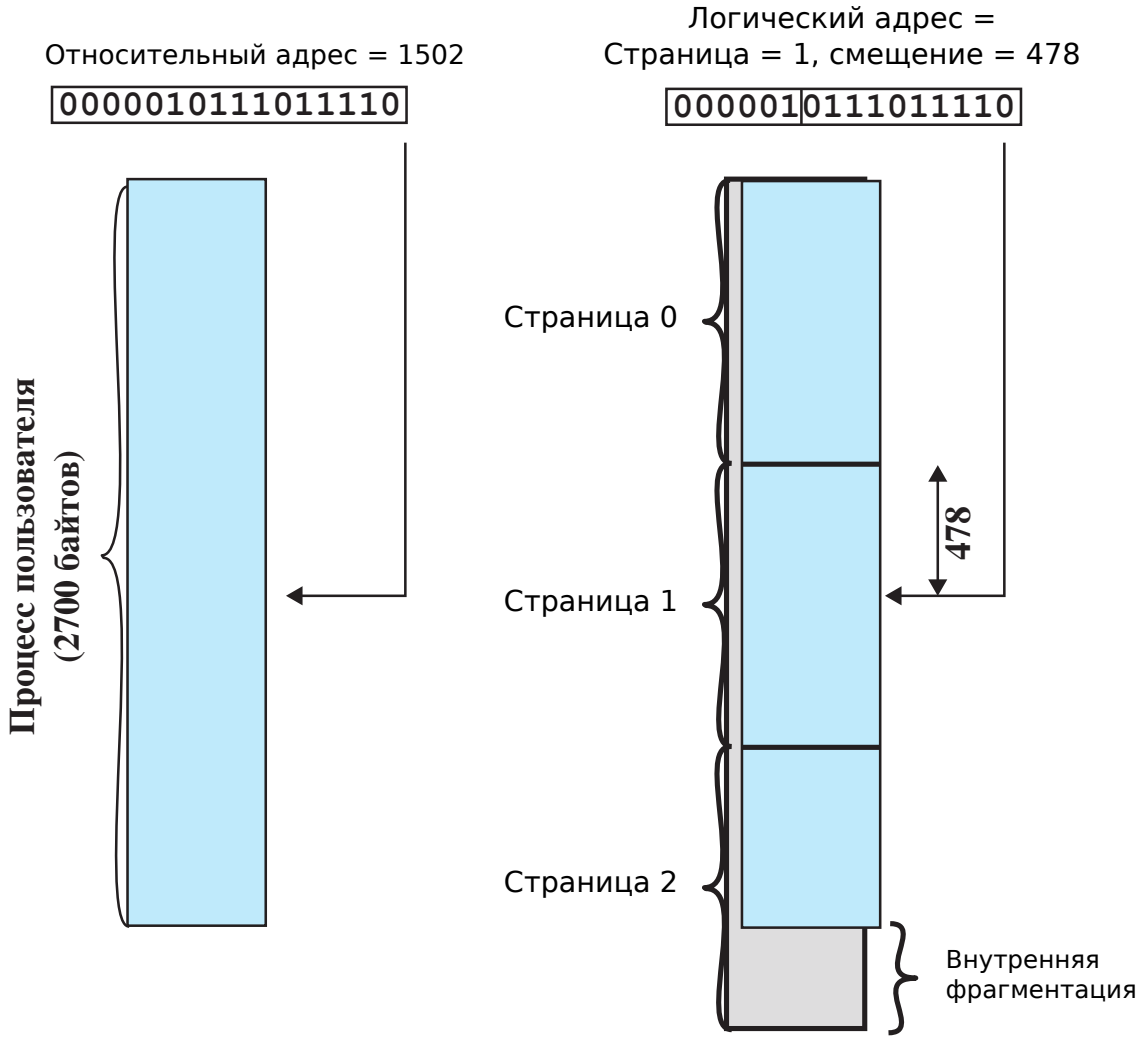
Распределение страниц по кадрам — 2



Страницы и адреса

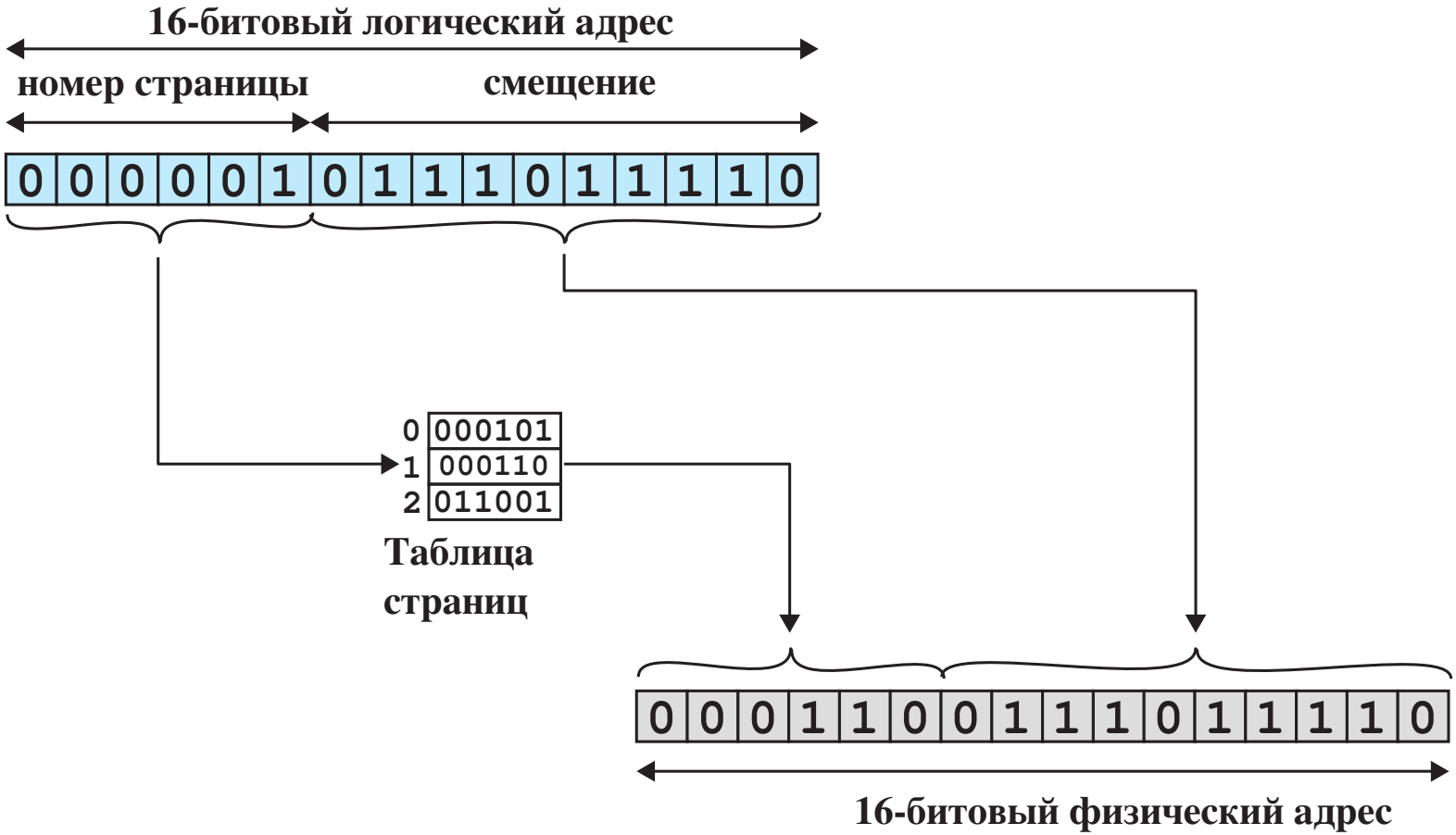
- Размеры страниц: 2^n байтов (1024, 2048, 4096).
- Относительный адрес = номер байта от начала адресного пространства.
- Логический адрес = (номер страницы, смещение).
- Физический адрес = (номер кадра, смещение).

Относительные и логические адреса



Размер страницы: 1Кб

Вычисление физического адреса



Сегментная организация памяти

Сегмент — раздел адресного пространства процесса:

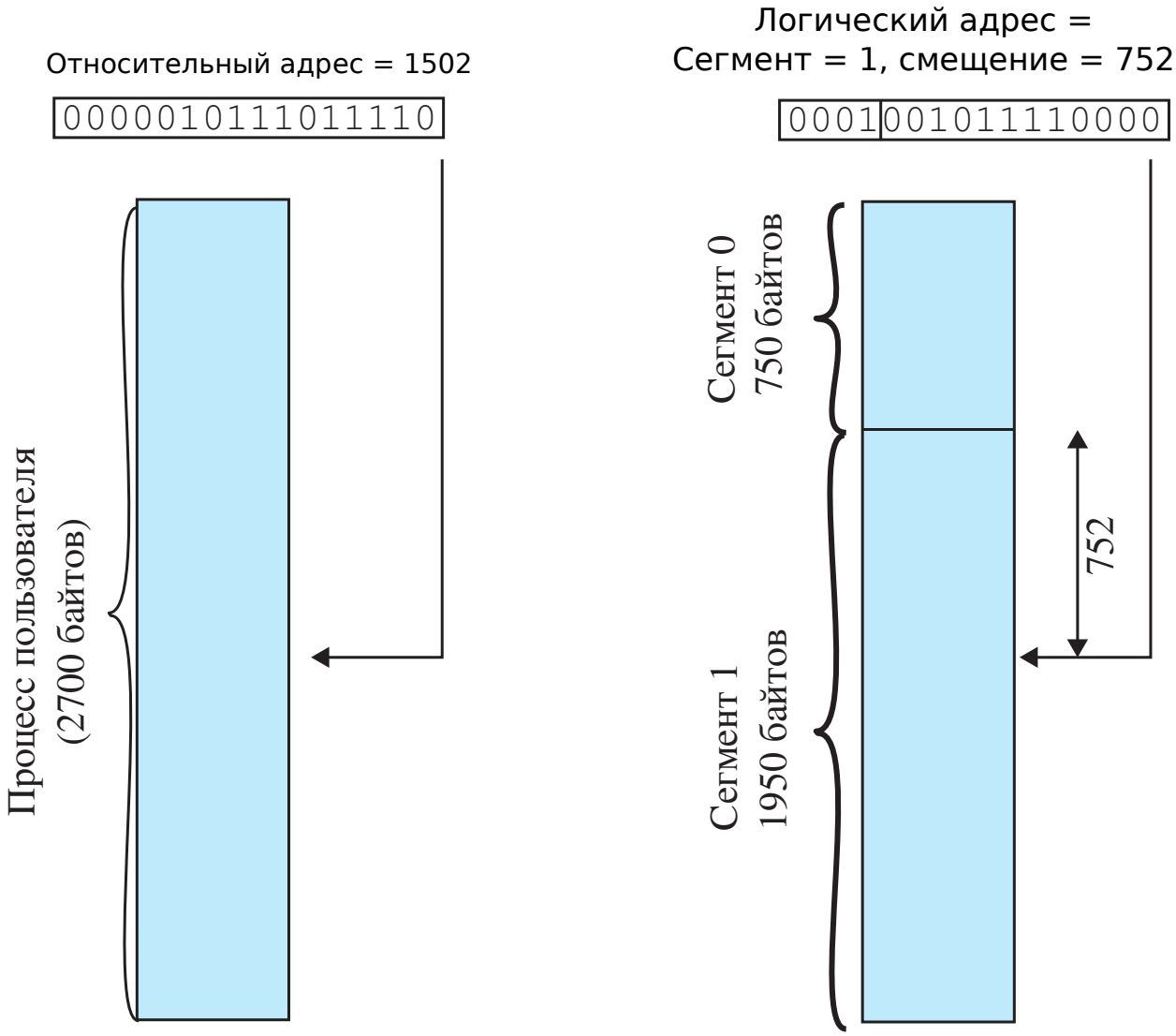
- сегмент кода;
- сегмент данных;
- сегмент стека.

Сегменты могут отражать модульную структуру программы.

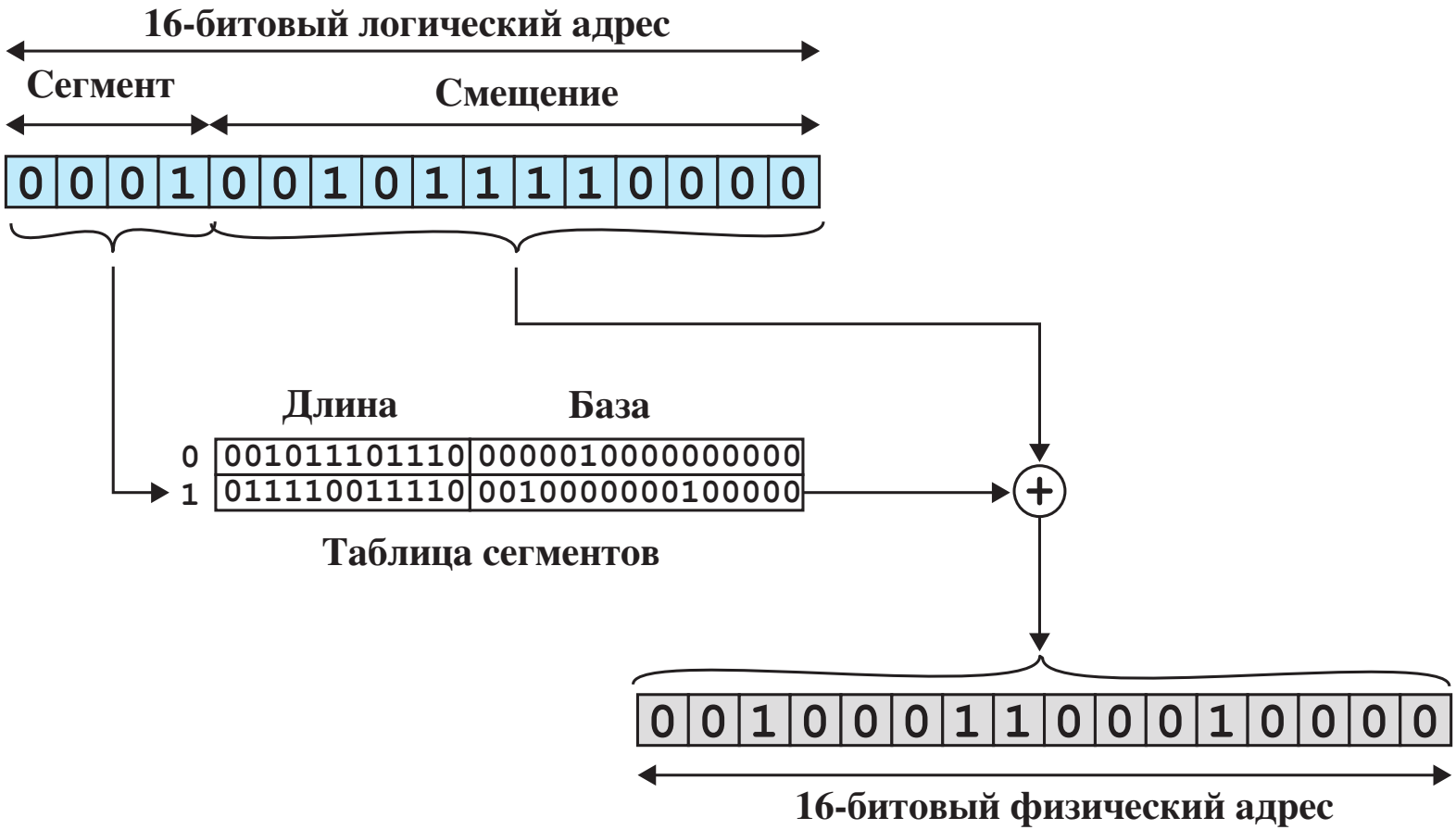
Сегменты и адреса

- Размер сегмента произвольный (есть ограничение сверху).
- Логический адрес = (номер сегмента, смещение).
- Таблица сегментов = номер сегмента $\mapsto$ (длина, база).

Относительные и логические адреса



Вычисление физического адреса



Содержание

Идея виртуальной памяти

1. Логическая и физическая адресация
2. Преобразование адреса
3. Перемещение
4. Разбиение адресного пространства на части

Идея виртуальной памяти

1. Логическая и физическая адресация
2. Преобразование адреса
3. Перемещение
4. Разбиение адресного пространства на части
5. **Должны ли все части процесса находиться в памяти?**

Виртуальная память — эффекты

1. Большее количество процессов в памяти.
2. Потенциально большое адресное пространство.

Виртуальная память: основные понятия

- Резидентное множество процесса.
- Вторичная память: файл подкачки / swap-раздел.
- Большие таблицы страниц.
- Биты присутствия и модификации в таблицах страниц.
- Страничная ошибка (page fault).

Алгоритмы управления виртуальной памятью

- Выборка (по требованию и предварительно)
- Размещение (UMA и NUMA)
- Замещение
- Управление резидентным множеством
- Управление загрузкой

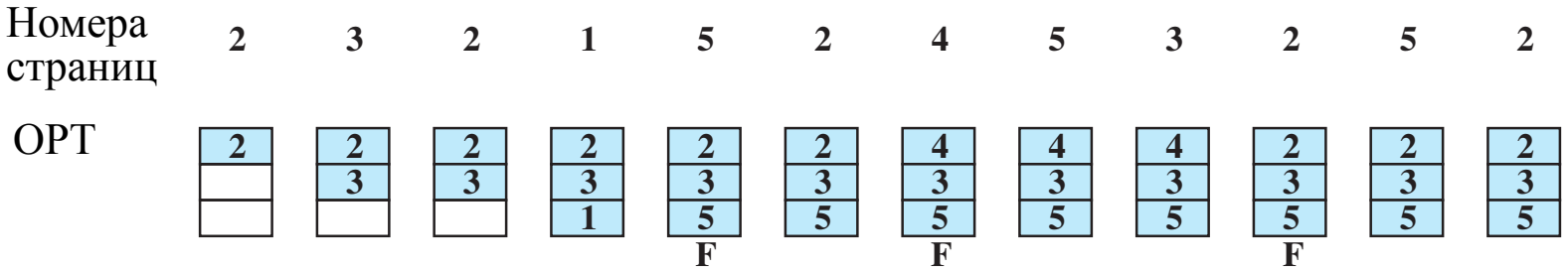
Алгоритмы замещения

Замещение

Выбор страницы, которая будет замещена страницей, загружаемой из вторичной памяти.

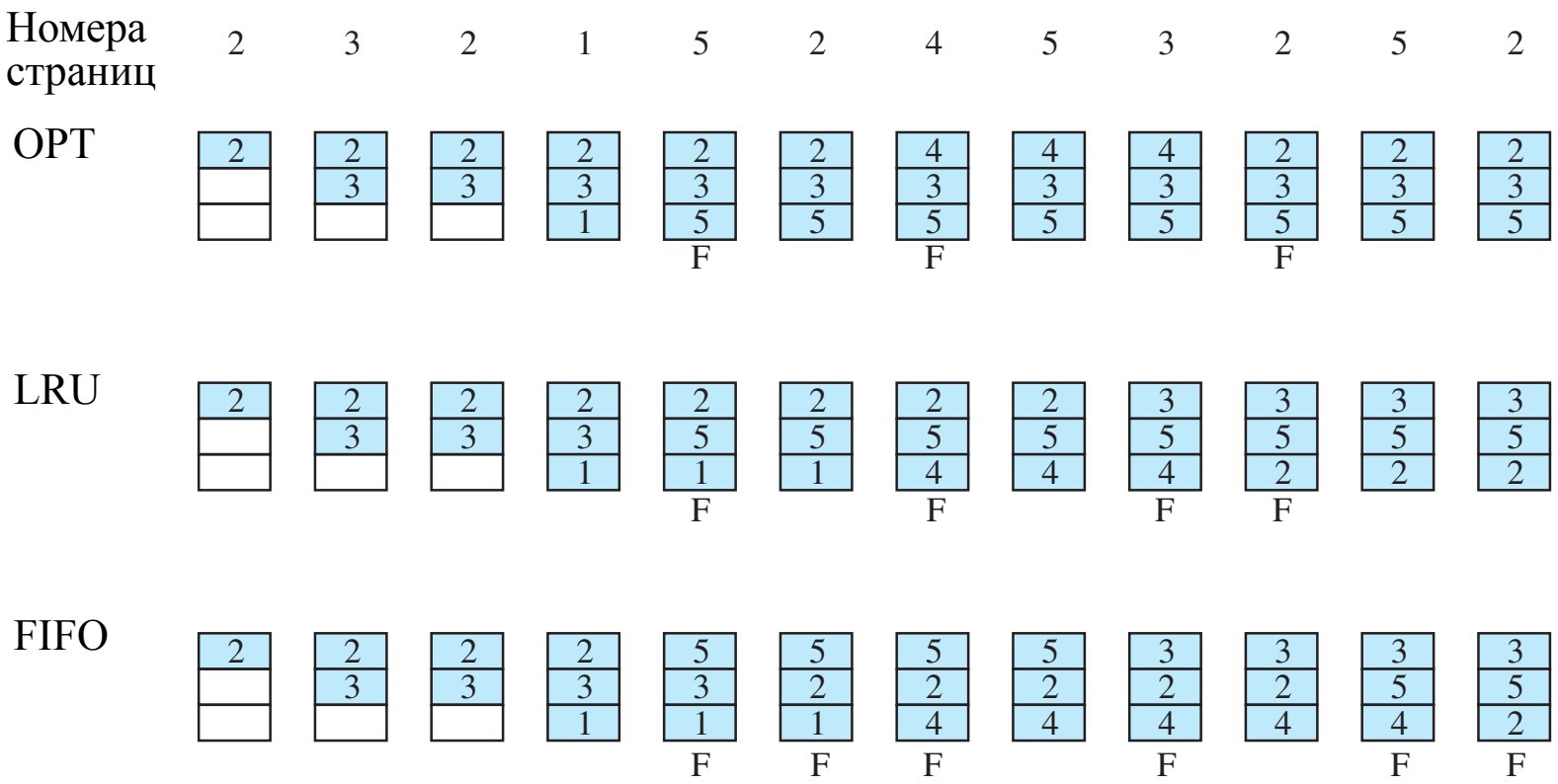
1. Оптимальный (OPT).
2. LRU — *Least Recently Used*.
3. Очередь страниц (FIFO).
4. Часовой (CLOCK).

Оптимальный алгоритм



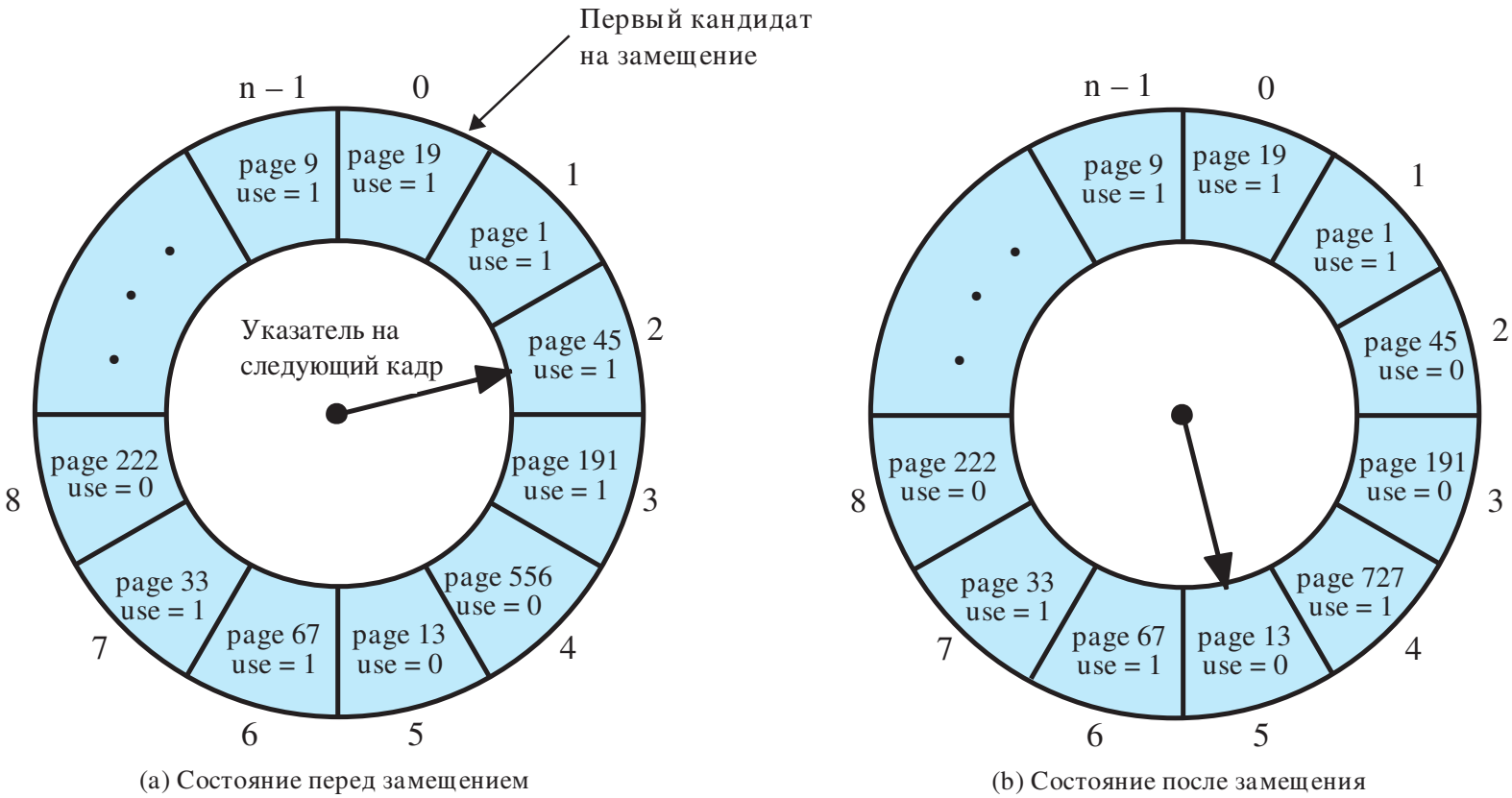
- Замещается страница, которая дольше всех не понадобится.
- Невозможно реализовать.

LRU и FIFO



- LRU: замещается страница, которая дольше всех не использовалась.
- FIFO: замещается страница, которая была загружена раньше остальных.

Часовой алгоритм

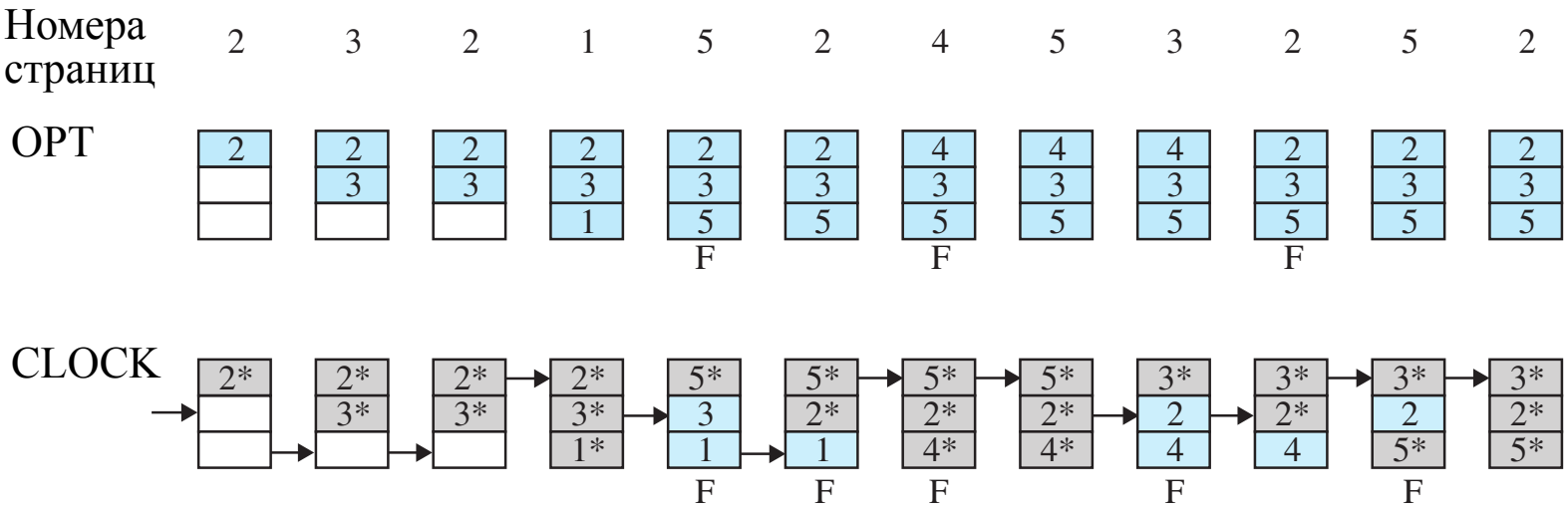


(a) Состояние перед замещением

(b) Состояние после замещения

- Циклический буфер кадров.
- Бит использования страницы (use).
- Поиск первой неиспользуемой страницы.

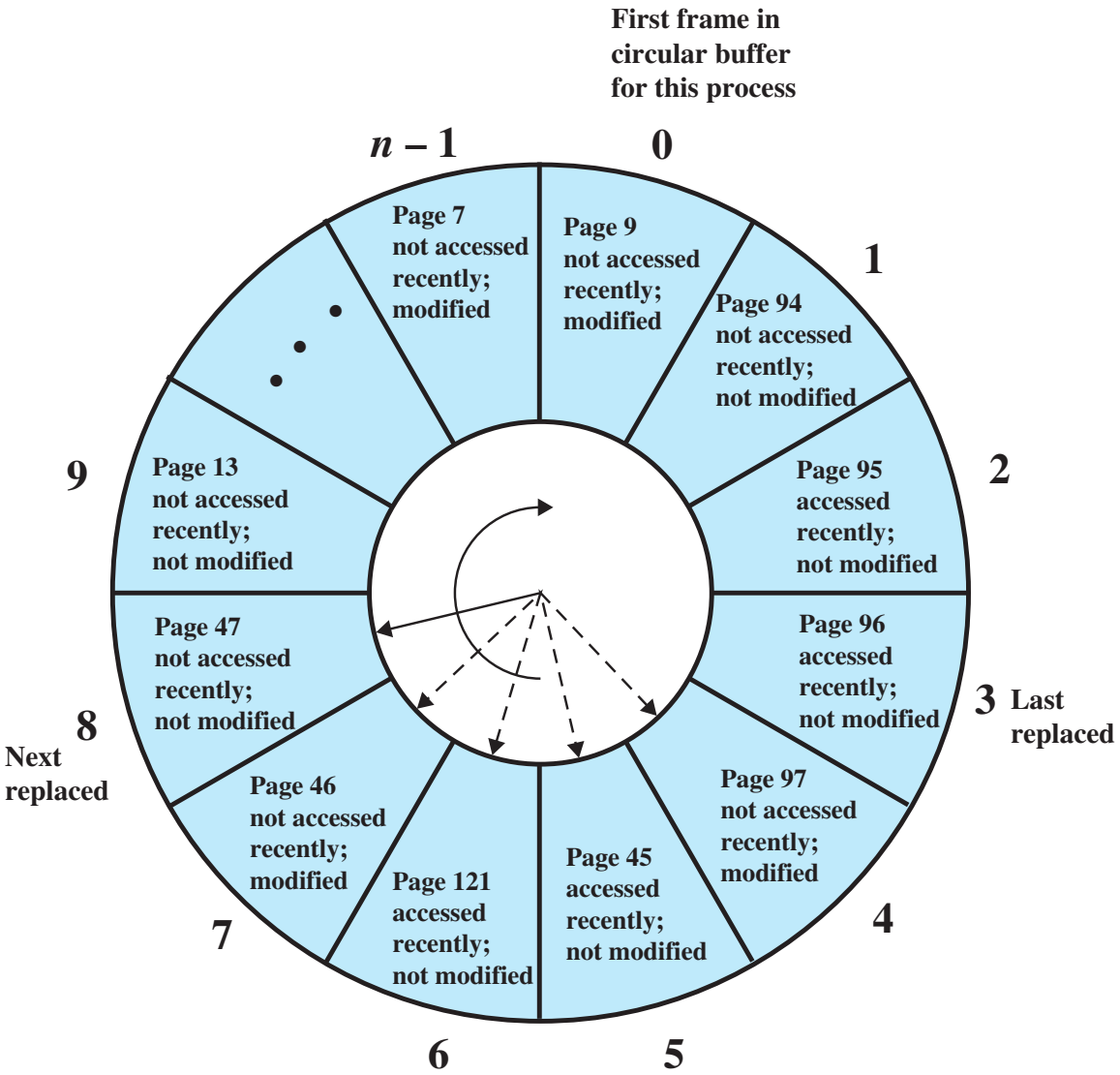
Часовой алгоритм



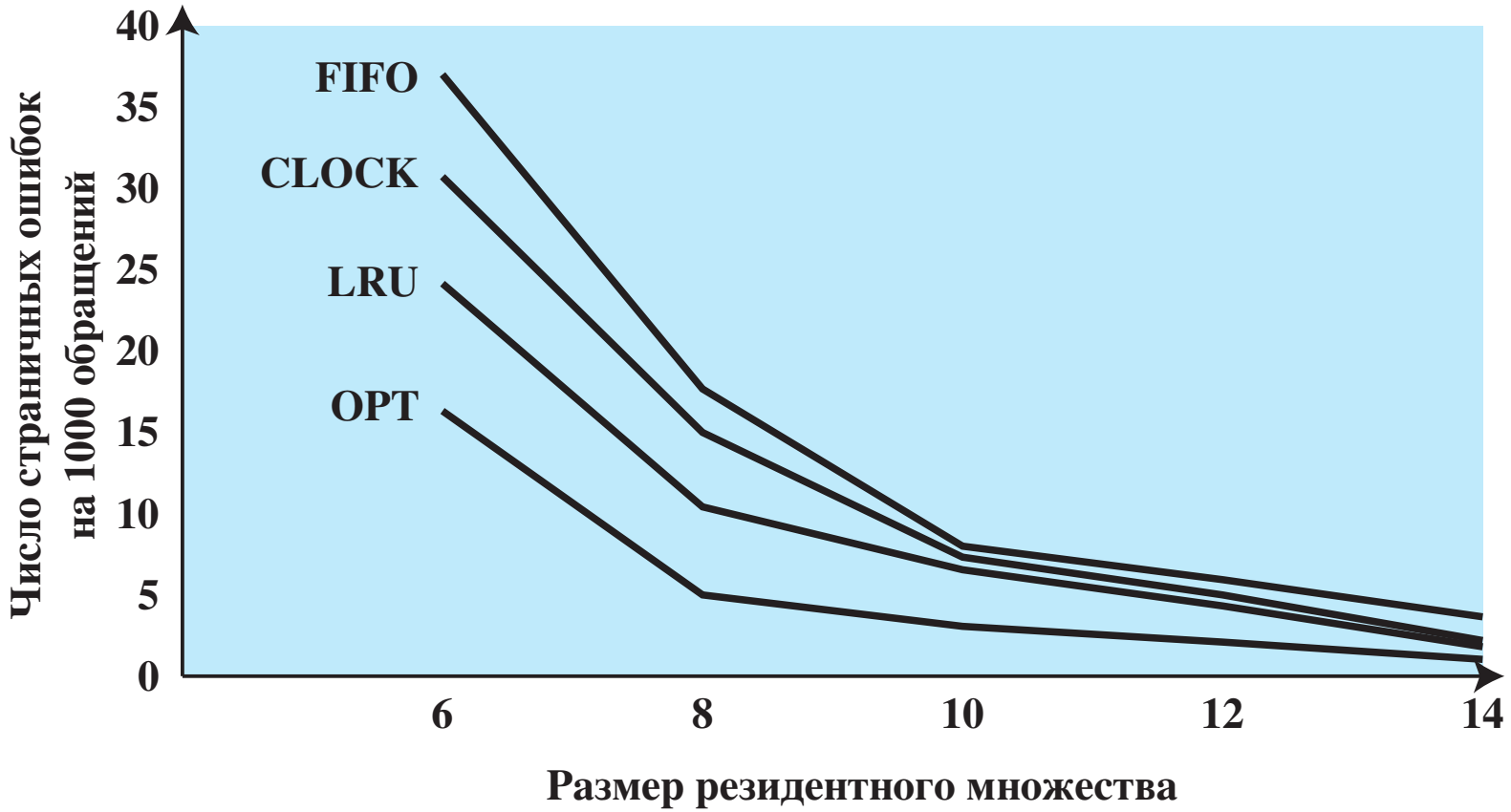
Варианты часового алгоритма

- Бит использования (u) и бит модификации (m)
- Классы кадров:
  - u = 0, m = 0
  - u = 0, m = 1
  - u = 1, m = 0
  - u = 1, m = 1
- Страница (u = 0, m = 0) предпочтительнее для замещения

Варианты часового алгоритма



Сравнение эффективности алгоритмов замещения



Управление резидентным множеством

- Размер и стратегии его определения (фиксированное и переменное распределение)
- Область видимости замещения (локальная и глобальная)

Управление загрузкой

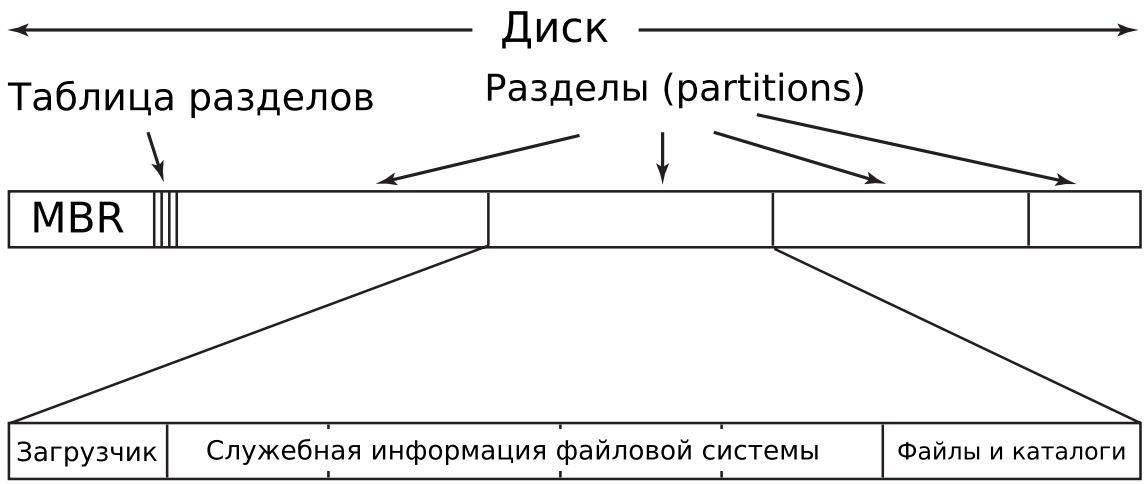
- Управление загрузкой — определение количества процессов, расположенных в оперативной памяти.
- Чем выше загрузка, тем меньше резидентное множество каждого процесса.
- Степень многозадачности.

Содержание

Что такое файловая система?

- Что из себя представляет служебная информация ФС?
- Как размещаются файлы и каталоги?
- Как учитывается свободное пространства диска?
- Какие дополнительные услуги может предоставлять ФС?

Зона ответственности файловой системы



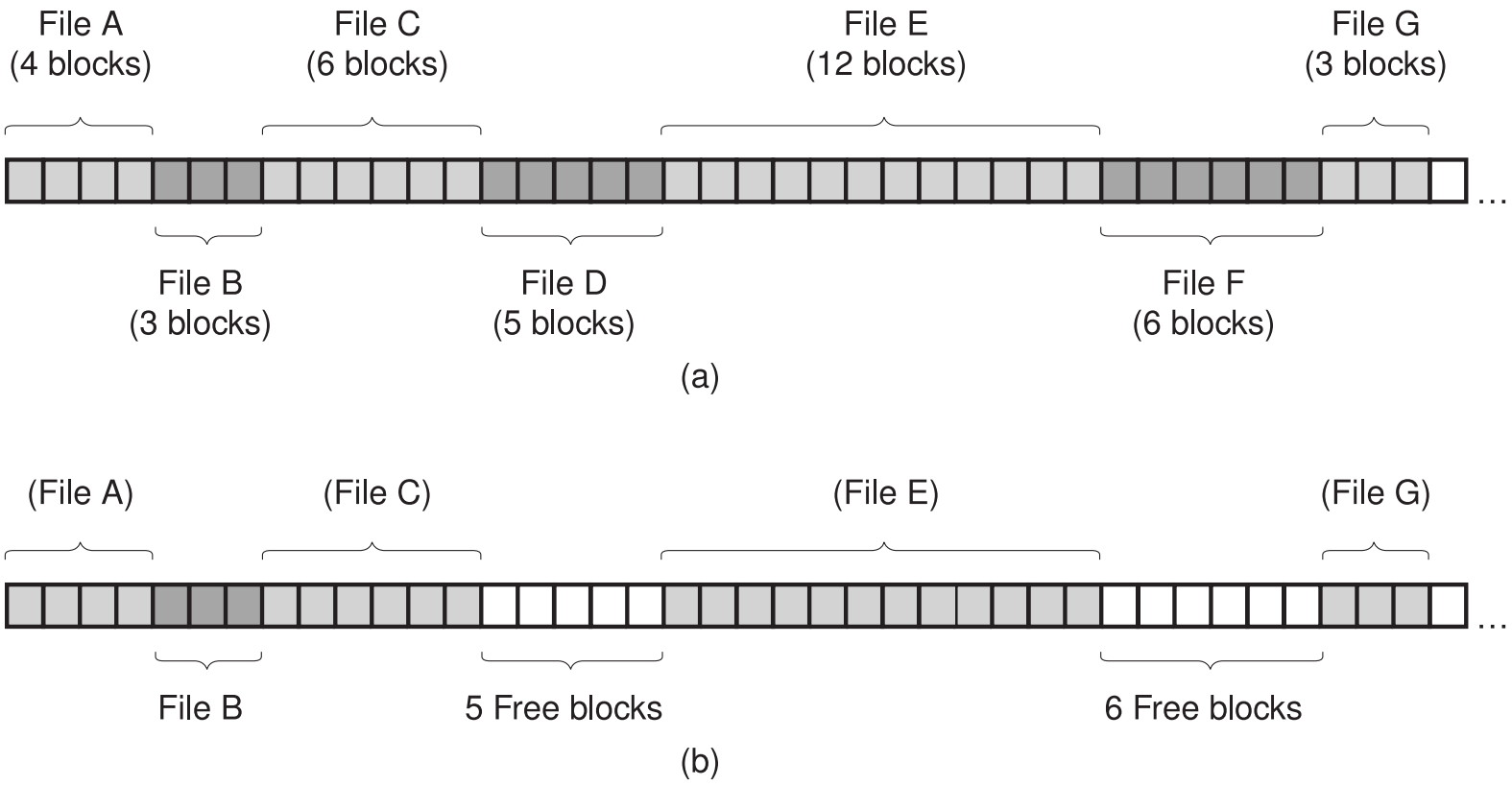
- Зона ответственности: либо первичный раздел, либо подраздел.
- Блок (кластер) — несколько секторов — единица размещения данных.

Содержание

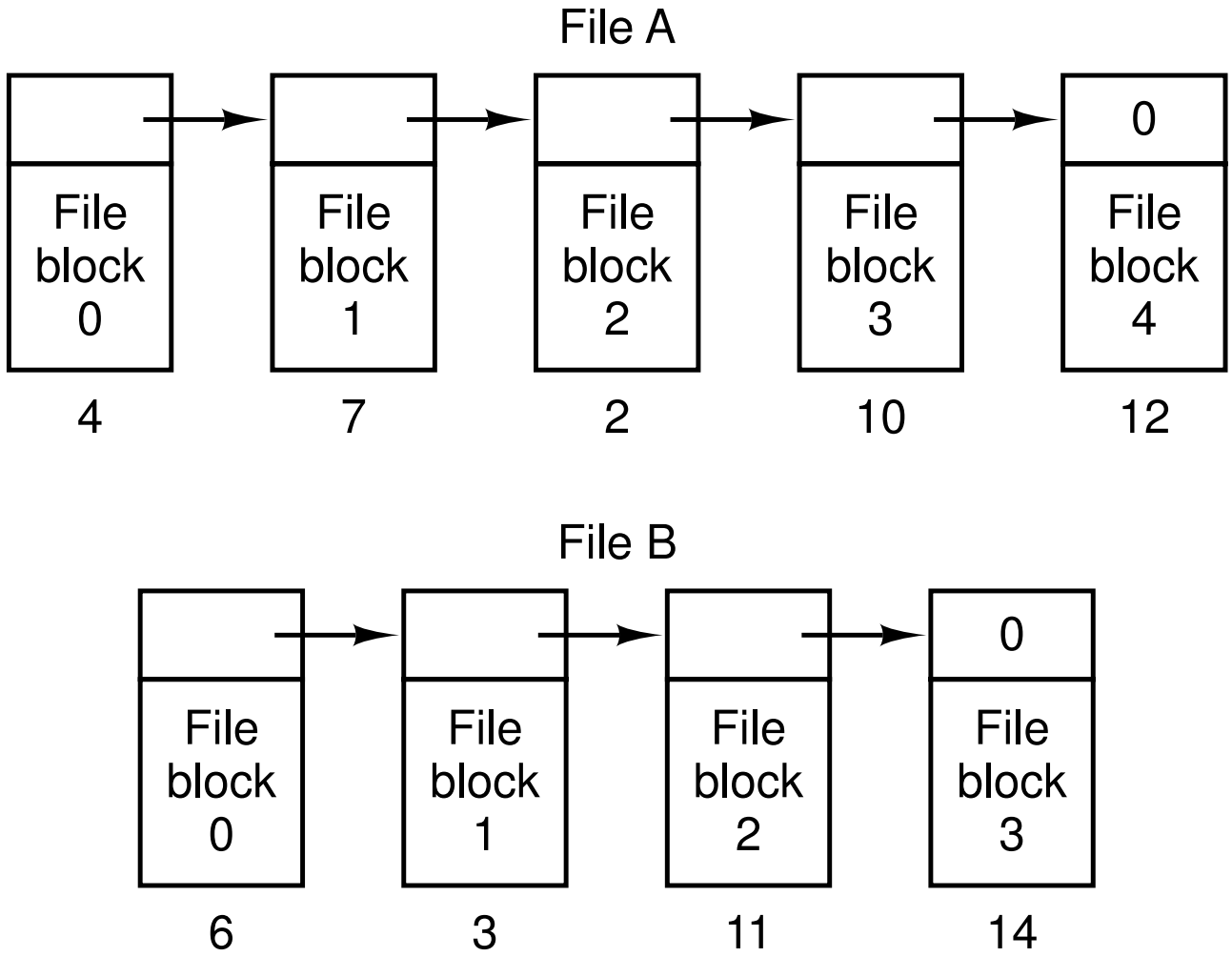
Файлы и атрибуты файлов

- Имя.
- Идентификатор.
- Тип (приложение-создатель).
- Местоположение в рамках файловой системы.
- Размер и максимальный размер.
- Информация о владельцах.
- Права доступа.
- Времена создания, модификации, последнего обращения.
- Блокировка, архивация, сокрытие, ограничения записи.

Непрерывное размещение

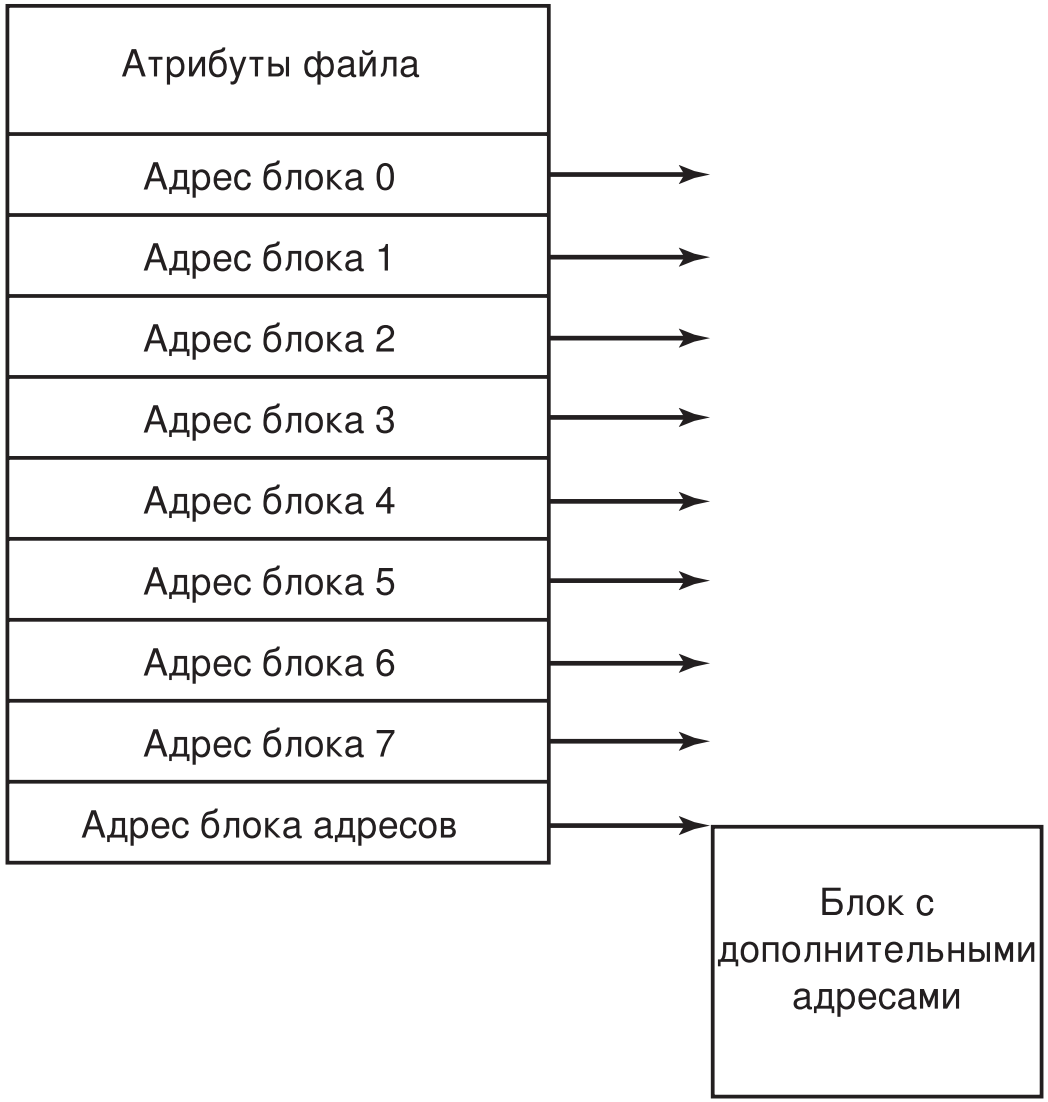


Связные списки

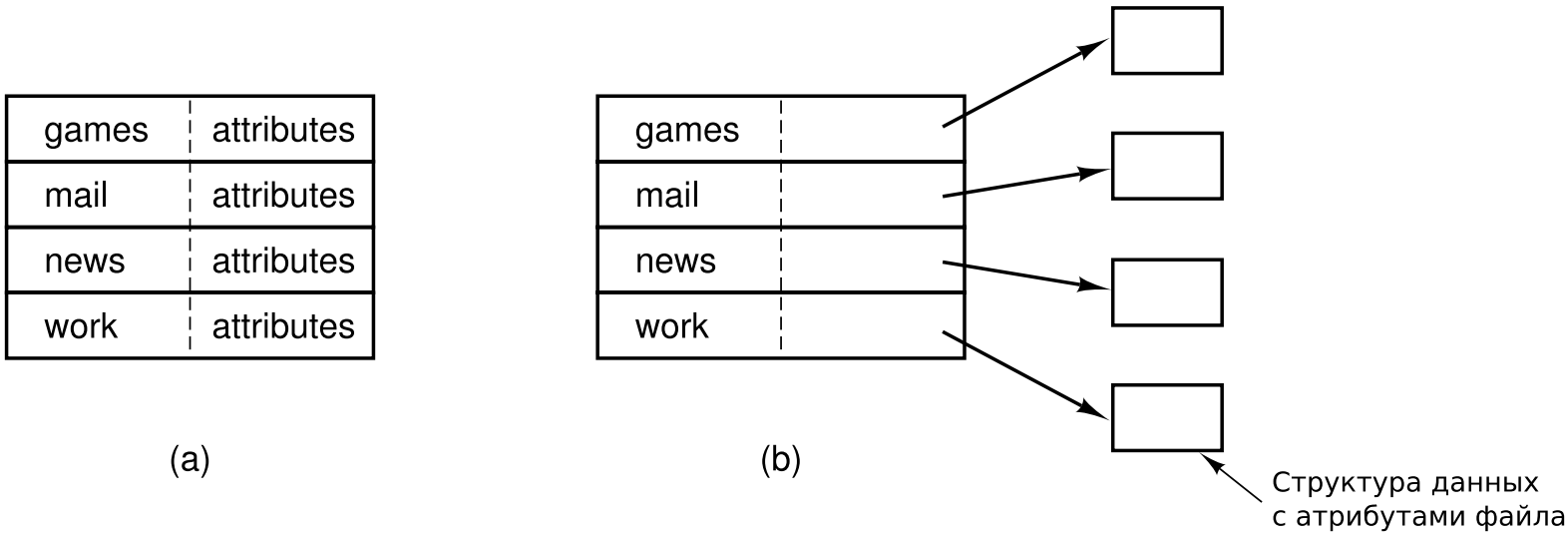


Таблицы размещения файлов (FAT)

I-nodes (индексные узлы)



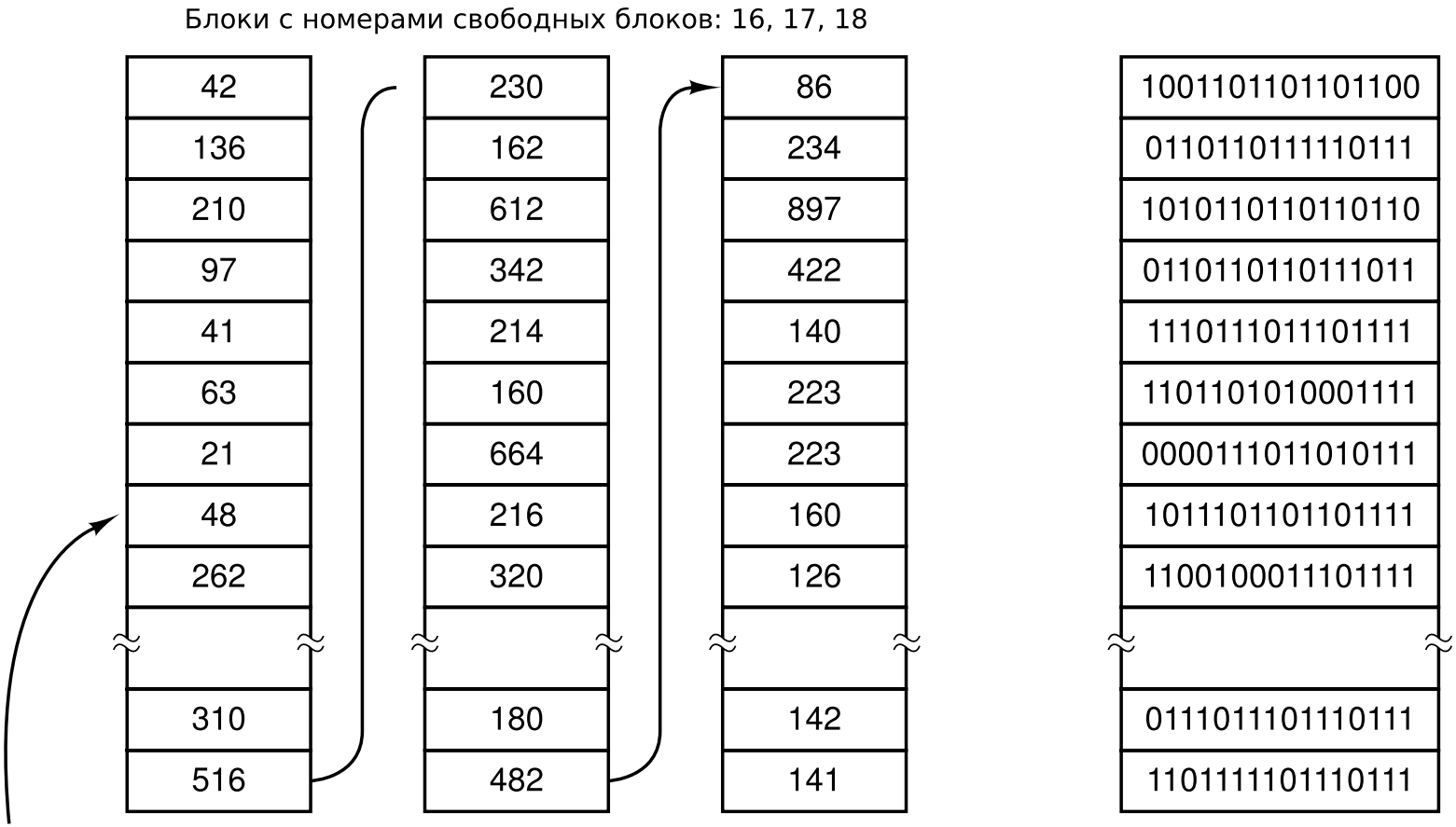
Реализация каталогов



Поиск файлов в каталоге

- Последовательный просмотр.
- Хеш-таблица или другая структура данных в каждом каталоге.

Хранение информации о свободных блоках



- Списки номеров свободных блоков.

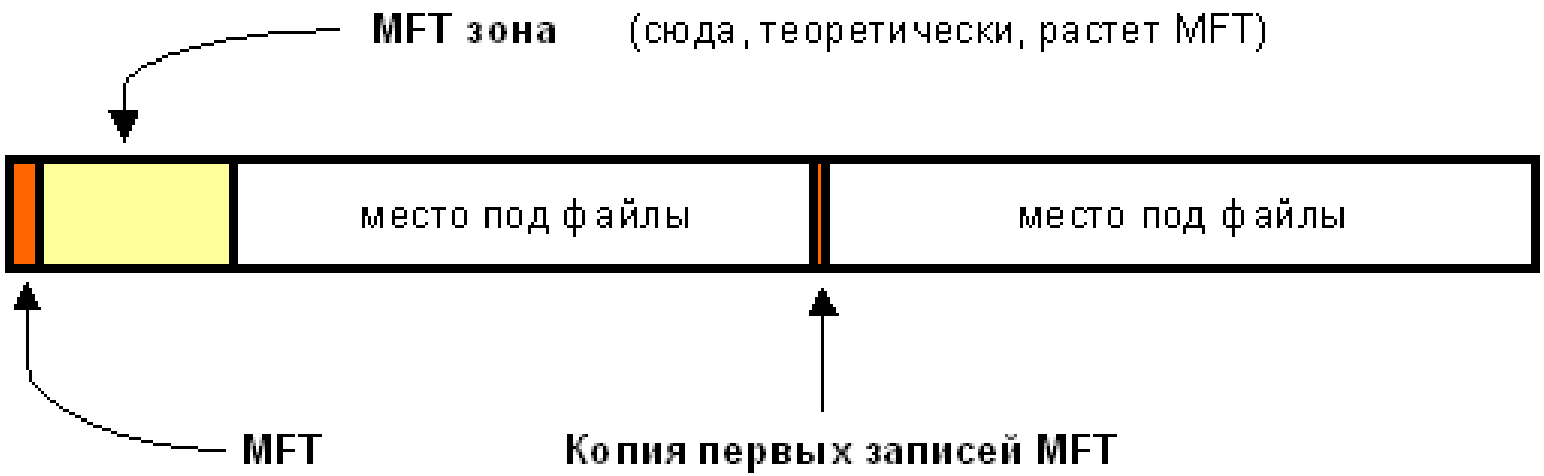
Содержание

Основные возможности NTFS

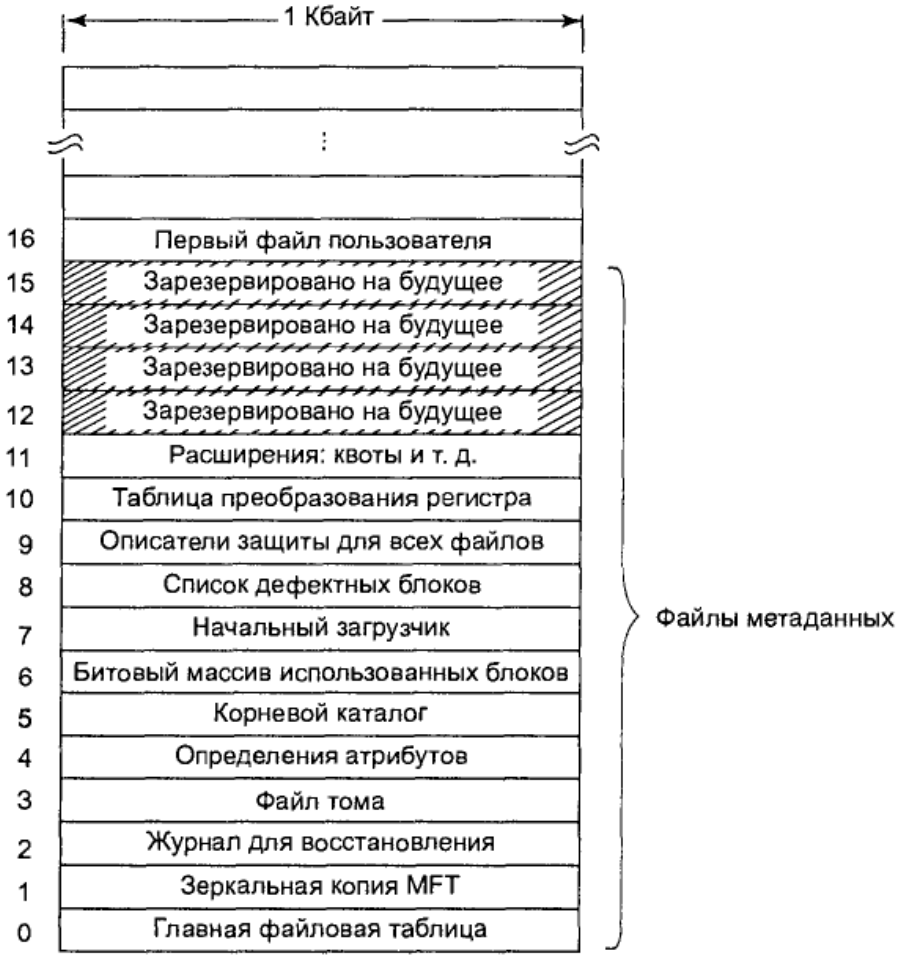
- Размеры блоков: от 512 байт до 64K.
- Максимальный размер диска: 2^{64} блока.
- Максимальный размер файла: 2^{64} байт.
- Альтернативные потоки данных.
- Квоты.
- Разреженные файлы.
- Монтирование.
- Жесткие ссылки.
- Сжатие файлов.
- Шифрование данных.
- Журналирование.

Распределение пространства раздела

- MFT — Master File Table — набор записей с информацией о файлах (1 Кб).
- MFT-зона — 12% раздела.



Начальные записи MFT

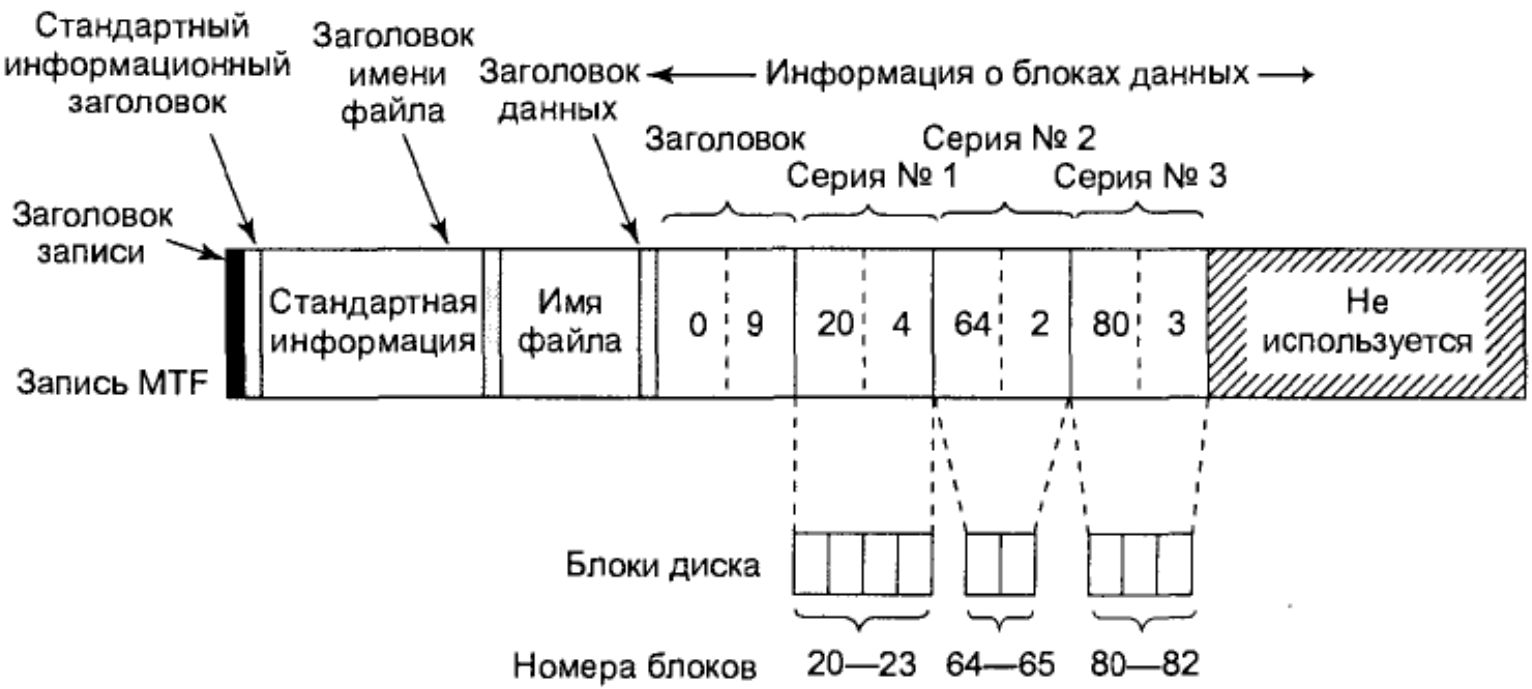


0. $MFT
1. $MFTMirr
2. $LogFile
3. $Volume
4. $AttrDef
5. $.
6. $Bitmap
7. $Boot
8. $BadClus

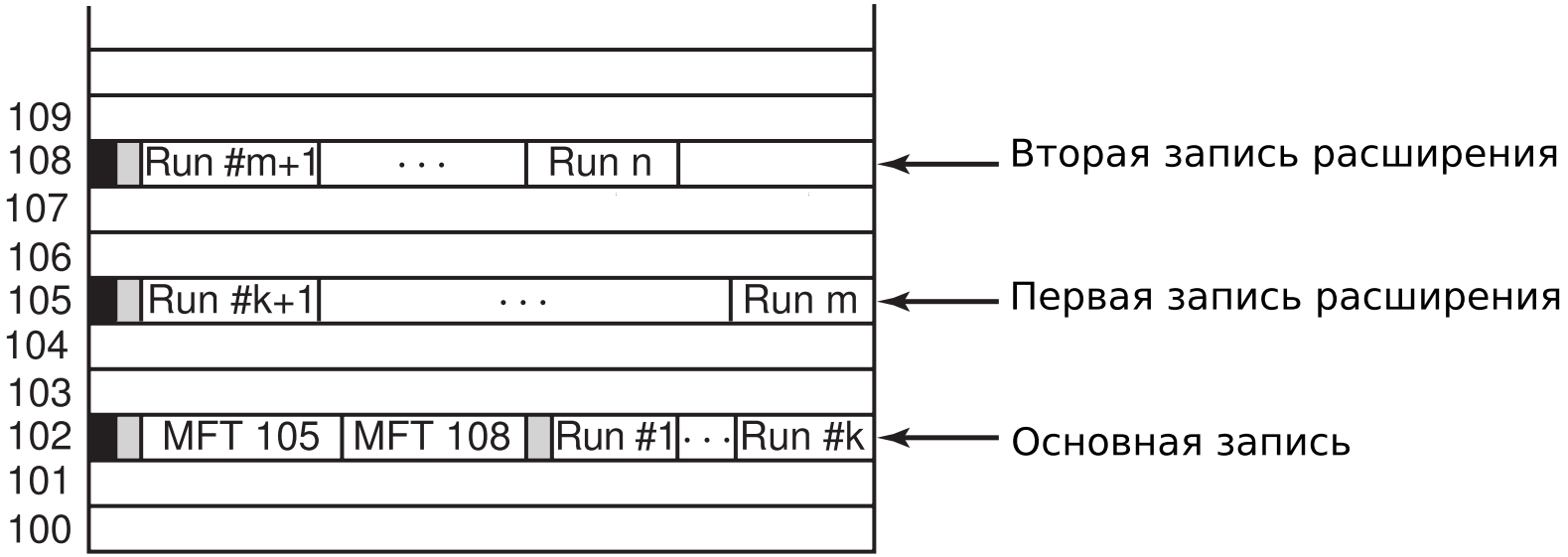
Атрибуты файлов в NTFS

- Резидентные и нерезидентные.
- Виды атрибутов:
  - стандартная информация (флаги, отметки времени);
  - имя файла;
  - числовой идентификатор;
  - информация о блоках данных;
  - номера записей с дополнительной информацией.

Структура записи MFT

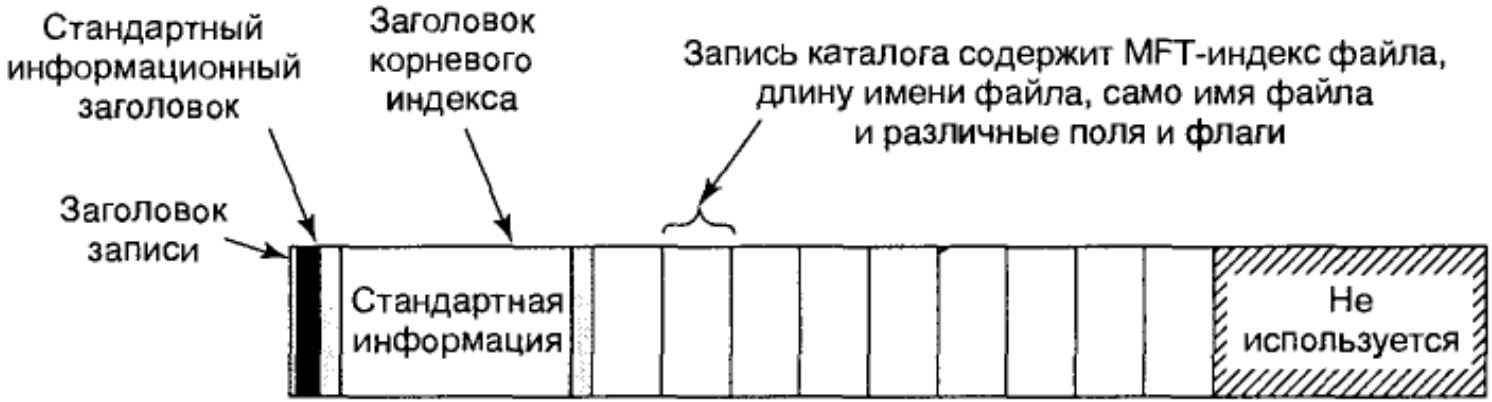


Запись MFT с расширениями



Запись MFT для каталога

Маленький каталог



Большой каталог

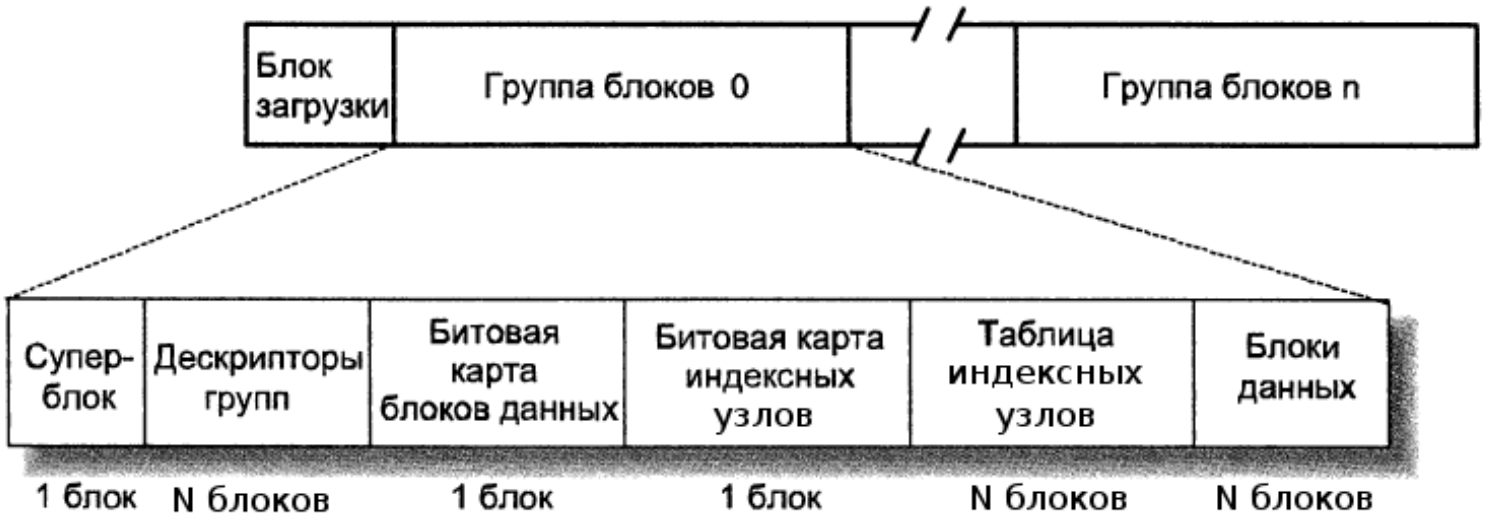
- Содержимое каталога — B^+-дерево.
- Ключи — имена файлов.
- Сопутствующая информация — номер MFT-записи файла.

Содержание

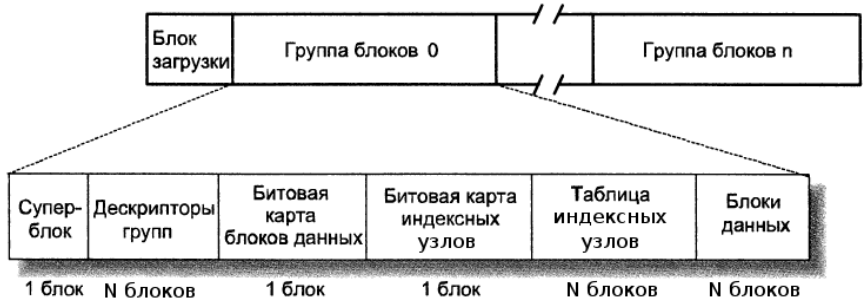
Файловая система ext2: общие характеристики

1. В основе — minix (ext=extended), 1994.
2. Выбор размера блока (от 1K до 4K).
3. Выбор количества индексных дескрипторов.
4. Разбиение раздела на группы.
5. Предварительное выделение смежных блоков.
6. Быстрые символьные ссылки.
7. Устойчивость и гибкость.

Структура раздела и группы блоков ext2

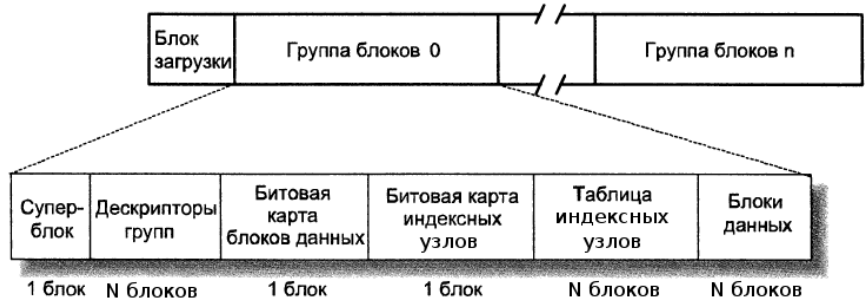


Содержимое суперблока



1. Размер файловой системы.
2. Общее и текущее количества свободных блоков и индексных дескрипторов.
3. Время последнего монтирования.
4. Размер групп.
5. Имя тома.
6. Количество предварительно выделяемых блоков.
7. Размер структуры индексного узла.
8. ...

Дескриптор группы

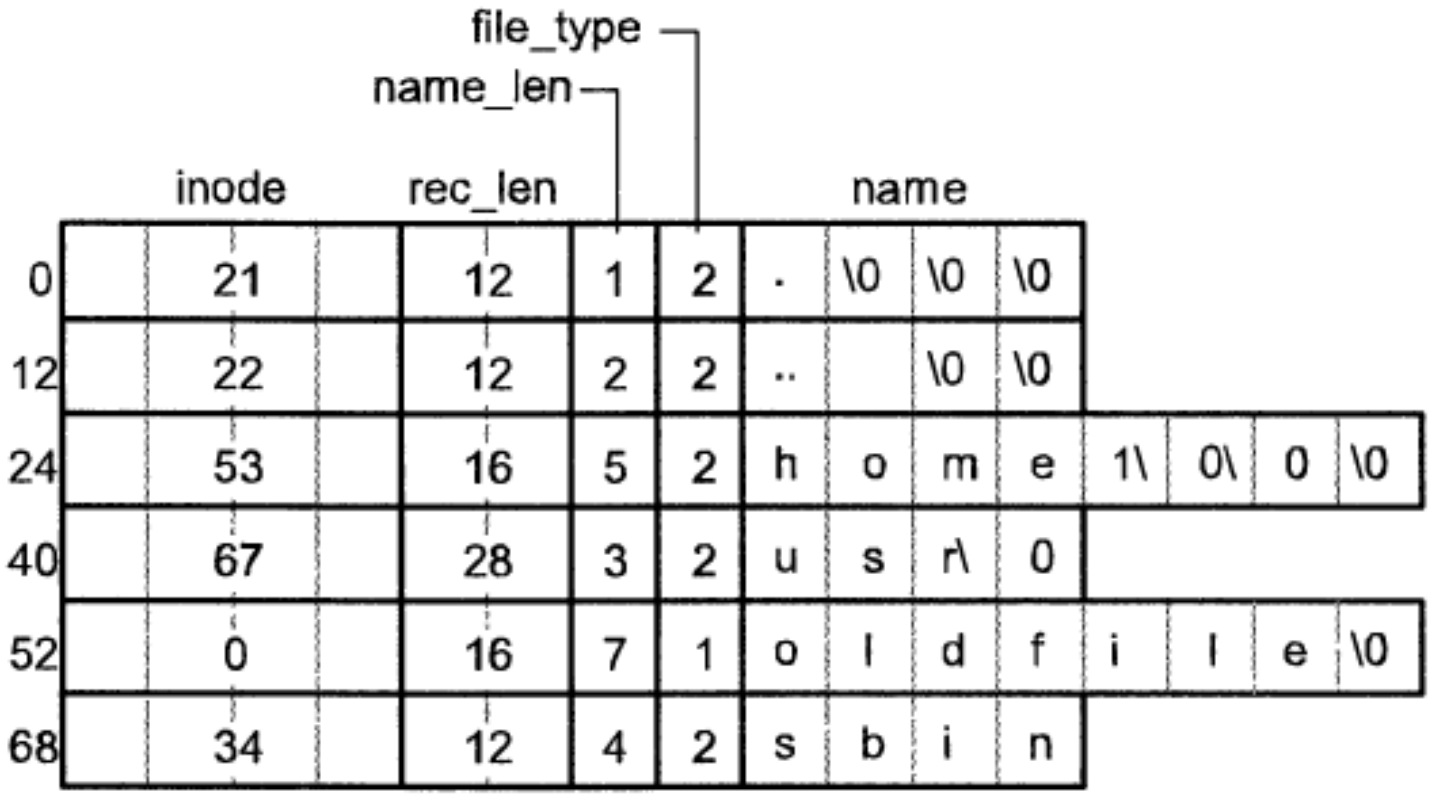


1. Номер блоков с битовыми картами блоков и индексных узлов.
2. Номер первого блока таблицы индексных узлов.
3. Количество свободных блоков и индексных узлов.
4. Количество каталогов.
5. ...

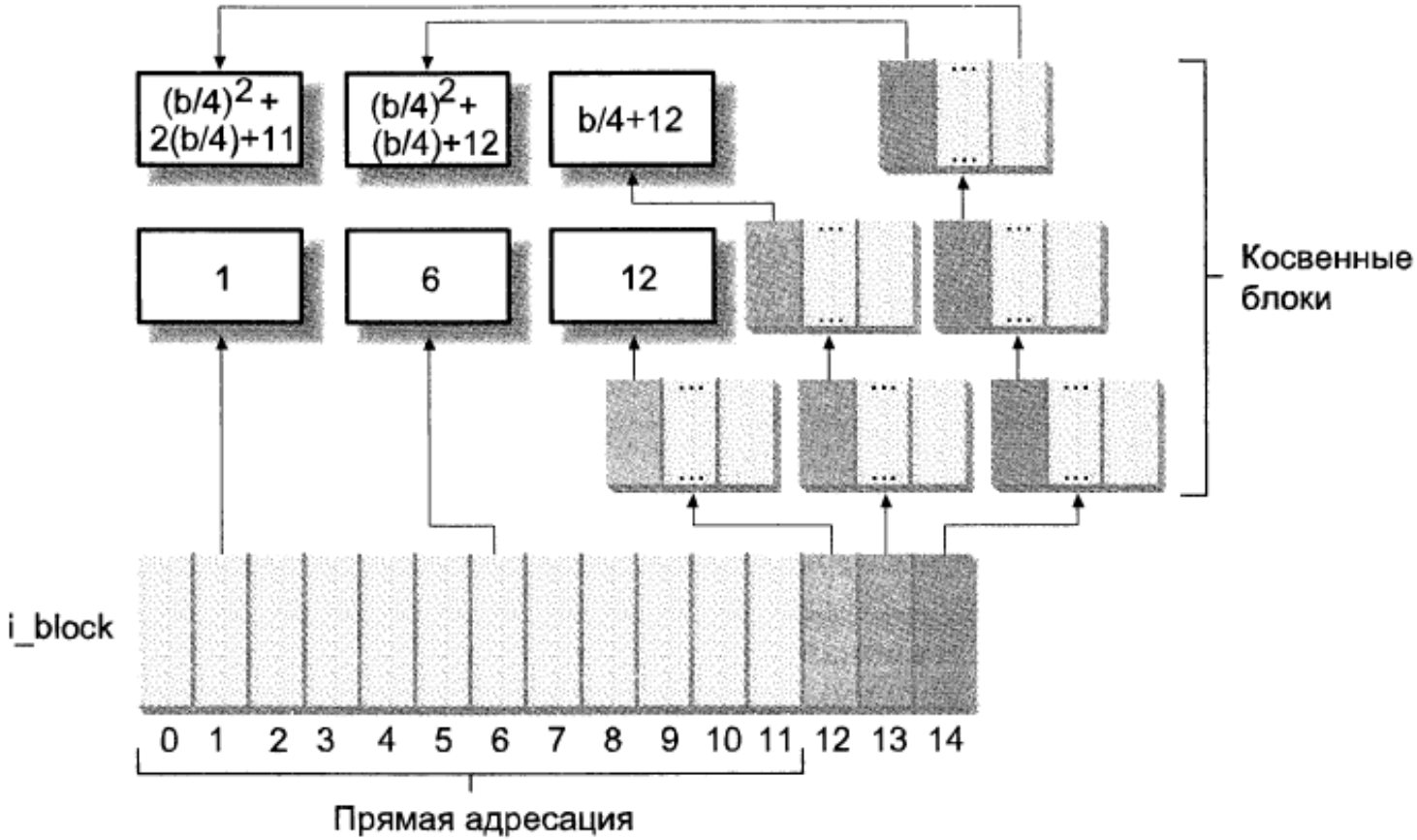
Индексный узел

1. 128 байтов.
2. Тип файла и права доступа.
3. Идентификатор владельца и группы-владельца.
4. Длина в байтах.
5. Времена последнего обращения и изменения.
6. Счетчик жёстких ссылок.
7. Номер блока с расширенными атрибутами.
8. Количество блоков данных.
9. Номера блоков данных.
10. ...

Пример каталога ext2



Адресация блоков данных



Номера блоков хранятся в индексном узле (прямая адресация) и его расширениях (косвенная адресация).

Верхние пределы для размера файла при разной адресации

Размер блока	Прямая	Косвенная, 1 уровень	Косвенная, 2 уровень	Косвенная, 3 уровень
1024	12K	268K	64,26M	16,06Г
2048	24K	1,02M	513,02M	256,5Г
4096	48K	4,04M	4Г	$\sim$ 4Т

Файловые системы ext3 и ext4

ext3

- Расширение ext2, 2001.
- Структура каталога: Htree.
- Журнализация: запись изменений в журнал, запись на диск, очистка журнала.

ext4

- Расширение ext3, 2008.
- Поддержка больших файловых систем.
- Экстенты вместо хранения номеров блоков.
- Дефрагментация во время работы.
- Ускоренная проверка диска.
- Атрибут времени создания файла.

Литература

1. Хопкрофт Дж., Мотвани Р., Ульман Дж. Введение в теорию автоматов, языков и вычислений. — 2-е изд. — М.: Вильямс, 2002. — 528 с.
2. Частичный конспект лекций по теории алгоритмов: `http://bit.ly/1cfJcPq`
3. Википедия: статья «Функциональное программирование».
4. Таненбаум Э. Современные операционные системы. — 2-е изд. — СПб.: Питер, 2007. — 1038 с.
5. Столлингс В. Операционные системы. — 4-е изд. — М.: Вильямс, 2004. — 848 с.