Операционные системы
Часть 3. Память, планирование

Клименков С.В.
Версия 1.0.0
30.08.2020
vk.com/serge_klimenkov

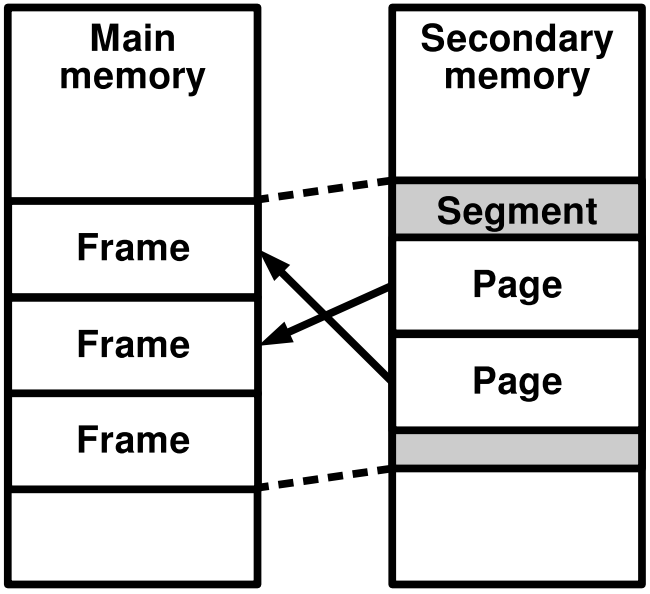
Управление памятью

3.1

- Frame, page, segment
- Требования к организации
- Фиксированные и динамические разделы
- Переместимость программ
- Paging и Segmentation

Определения

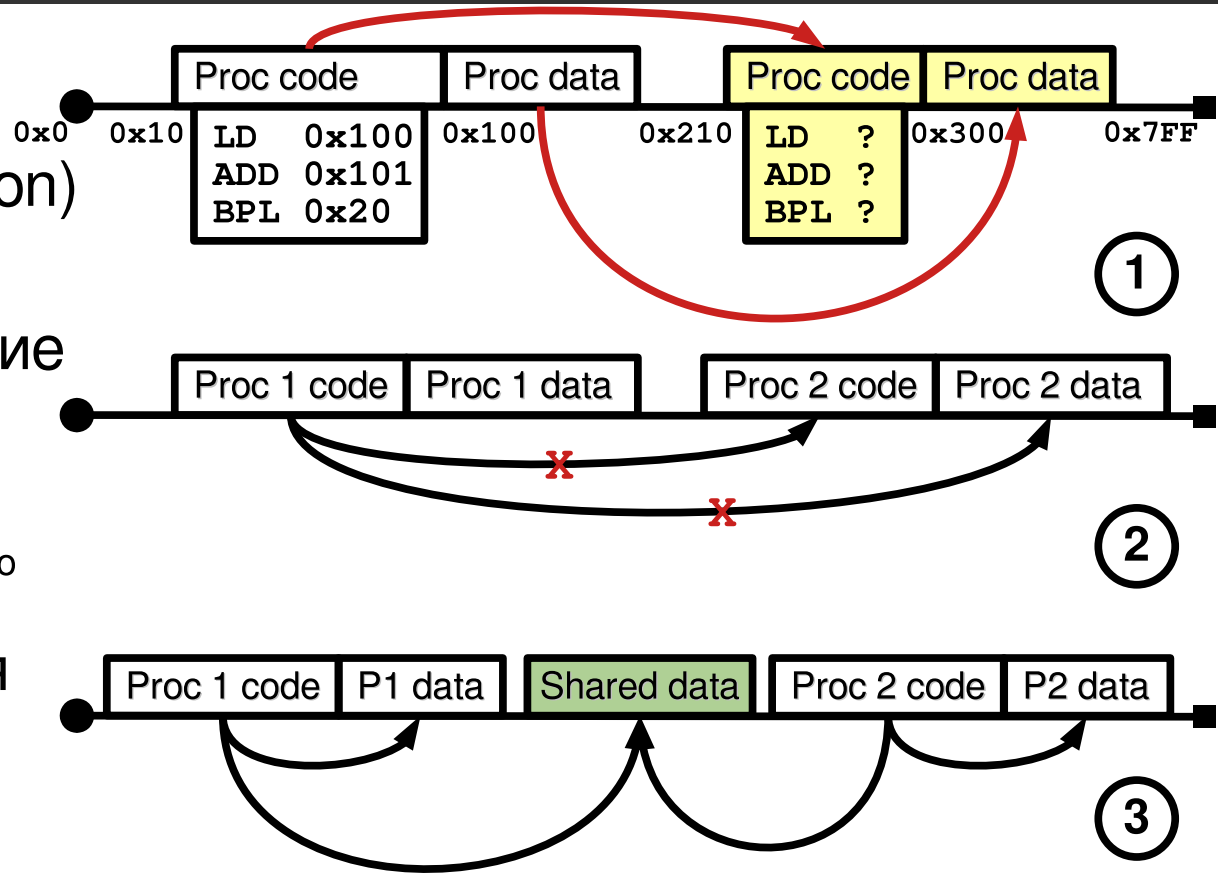
- Основная память (main memory)
  - Область, где процессор может исполнять программы
- Вторичная память (secondary memory)
  - Область, где программы и данные могут храниться, в том числе и во время выполнения
- Кадр (frame) — область фиксированного размера основной памяти
- Страница (page) — область фиксированного размера во вторичной памяти, которая может быть скопирована в кадр
- Сегмент (segment) — область переменного размера в основной или вторичной памяти, может быть разделена на страницы



Требования к организации памяти

Требования:

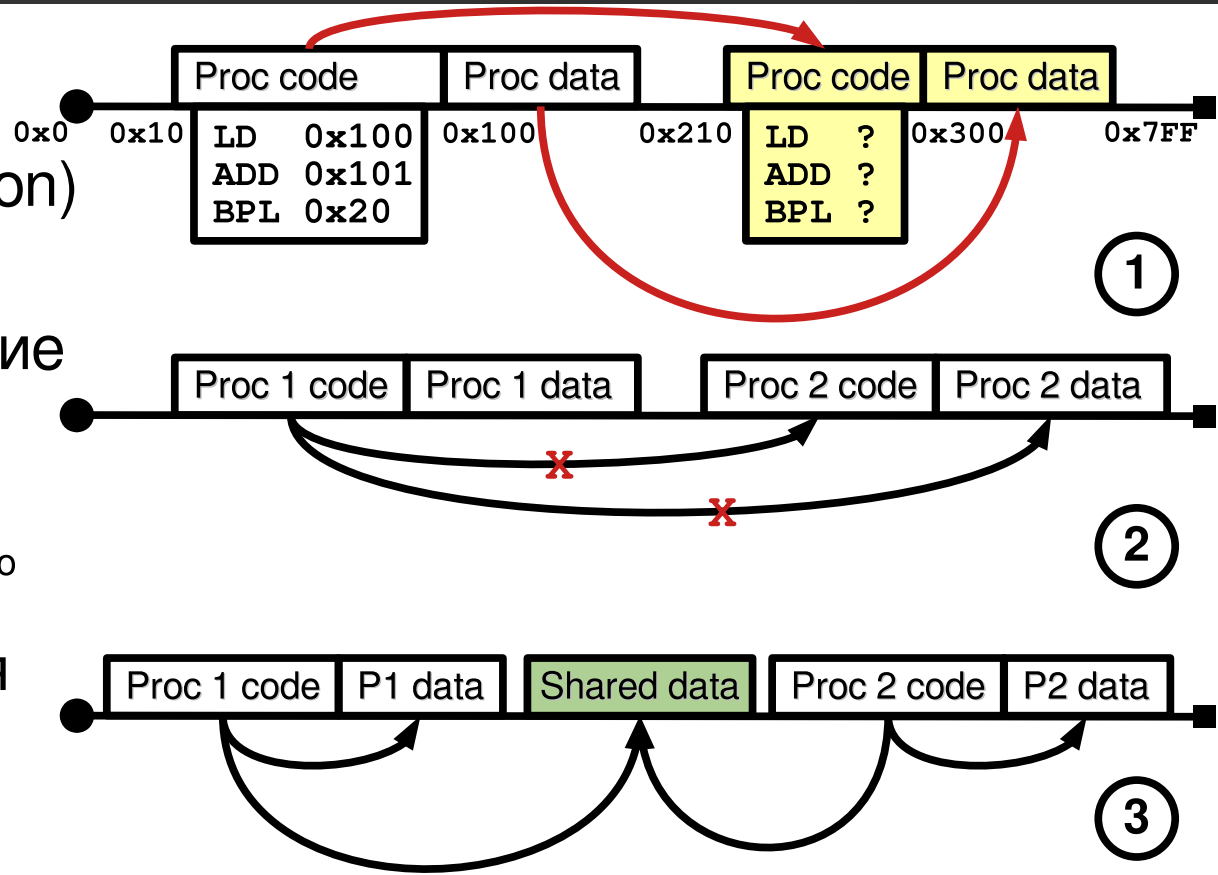
1) Переместимость (relocation)
2) Защита (protection)
3) Совместное использование (sharing)
4) Логическая организация
   - Управление основной и вторичной памятью
   - Модульная организация пограмм
5) Физическая организация
   - Поддержка аппаратуры
   - Перекрытие (overlays) и виртуализация
   - Организация области подкачки



Требования к организации памяти

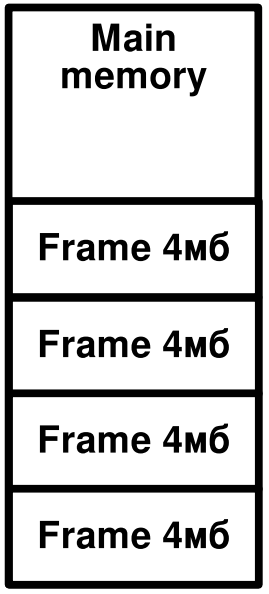
Требования:

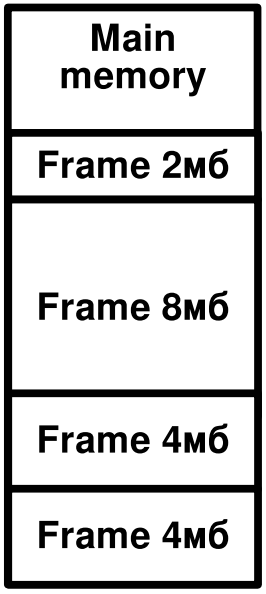
1) Переместимость (relocation)
2) Защита (protection)
3) Совместное использование (sharing)
4) Логическая организация
   - Управление основной и вторичной памятью
   - Модульная организация пограмм
5) Физическая организация
   - Поддержка аппаратуры
   - Перекрытие (overlays) и виртуализация
   - Организация области подкачки



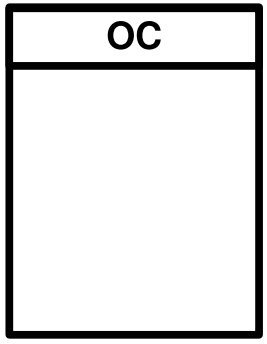
Fixed partitioning

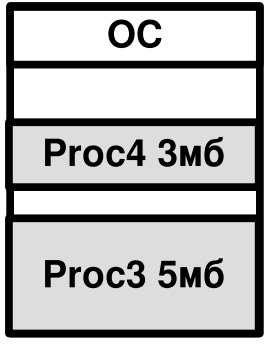
- У программ разная длина, их сложно упаковывать в память
- А что если оперировать кадрами заданного размера?
- Минусы
  - Память расходуется неэффективно для небольших процессов
  - Ограничение на количество одновременных процессов
  - Если процесс не помещается в раздел, он должен использовать перекрытия (overlays), самостоятельно загружая и выгружая свои части





Dynamic partitioning

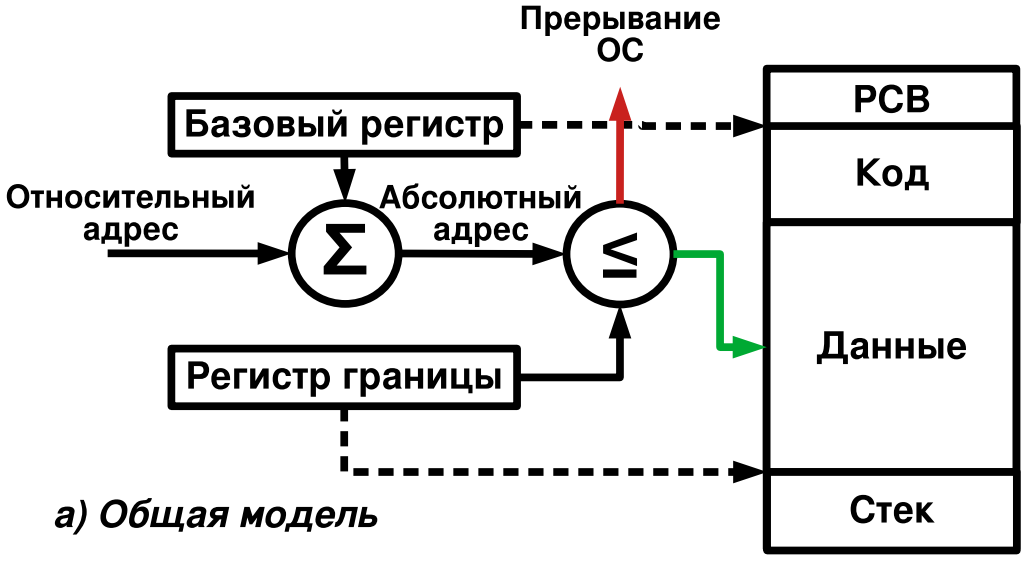




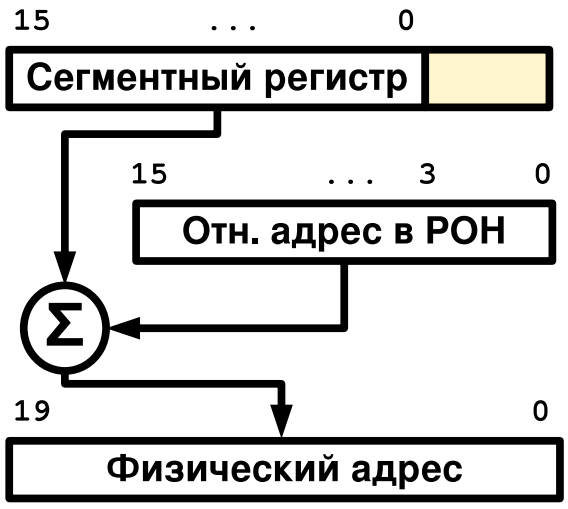
- Блоки процессов разного размера, ОС встраивает их в свободные места
- Минусы
  - Фрагментация памяти
  - Необходимость упаковки памяти
  - Начинает за здравие, заканчивает ну вы понимаете…
  - Сложные и медленные алгоритмы размещения в памяти
- Подходит для загрузки драйверов
  - Поэтому и нужна была перезагрузка при добавлении драйверов
- Разумный компромисс - «Buddy System» - см. Столлингса.

Перемещение программ

- Копировать программы средствами ОС — долго!
  - Нужна аппаратная поддержка, например, как в 8086
- Модели аппаратного перемещения:



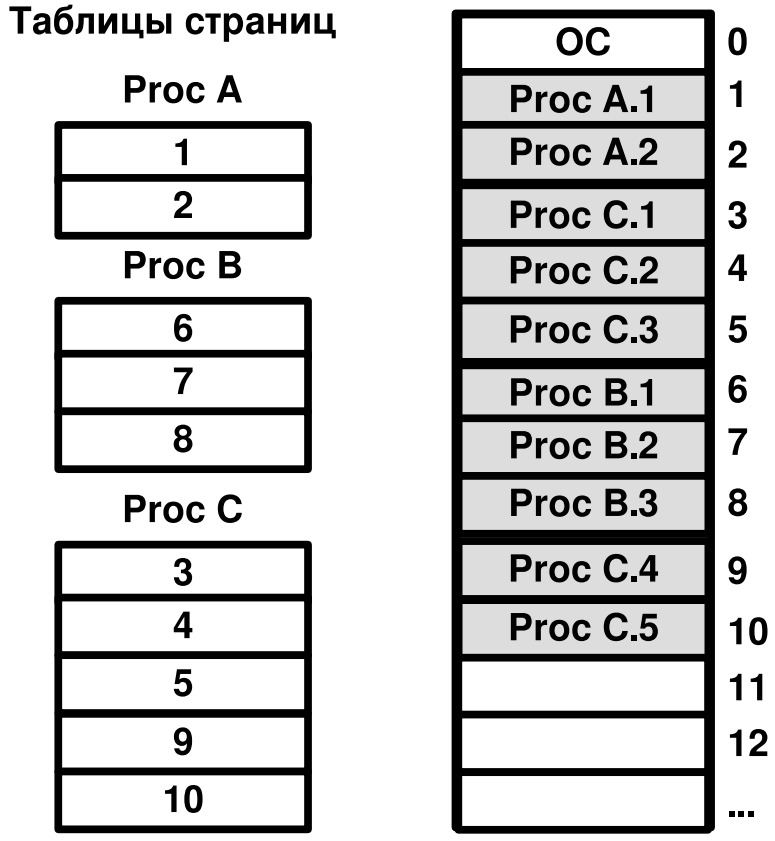
а) Общая модель



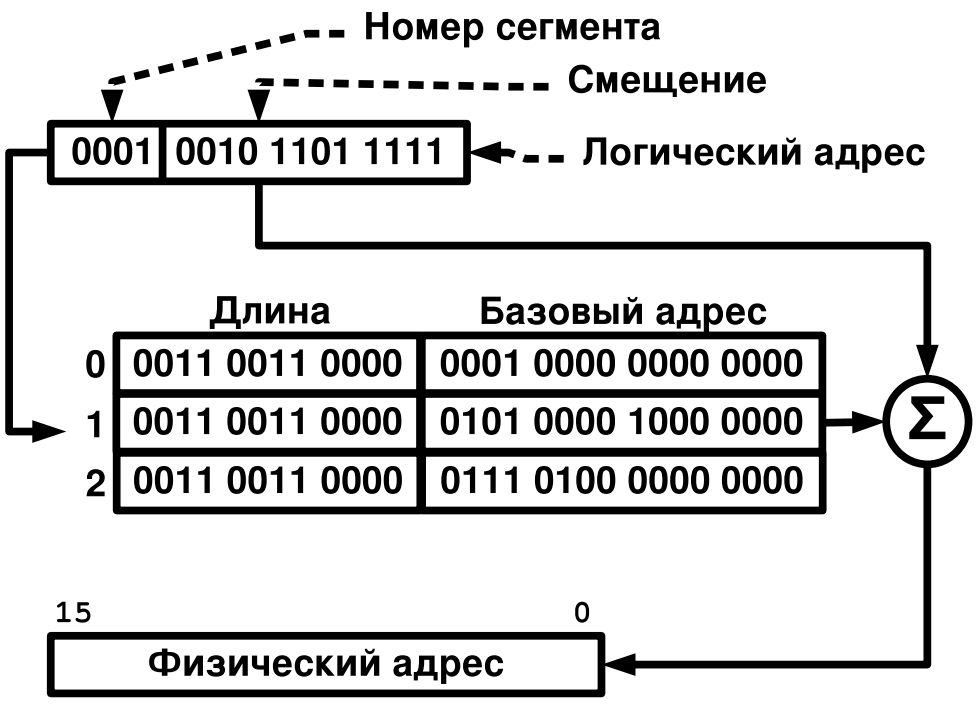
б) Модель 8086

Simple paging

- Давайте разделим память на кадры одинакового (небольшого размера)
  - Пусть страница занимает целиком кадр
  - Для удобства размер кратен степени 2
- Уменьшится внутренняя фрагментация
  - Пустое место будет в последнем блоке каждого процесса
- Внешняя фрагментация исчезнет совсем
- Необходимы таблицы страниц
  - ОС должна знать, где ее герои



Simple Segmentation



- Таблица сегментов. Программист должен устанавливать:
  - Длину сегмента
  - Базовый адрес
- Внутренняя сегментация отсутствует
- Внешняя сегментация снижается
-

Виртуальная память

3.2

- Программная и аппаратная организация
- Memory paging & TLB
- Сегментно-страничная организация
- Стратегии управления

Термины и определения

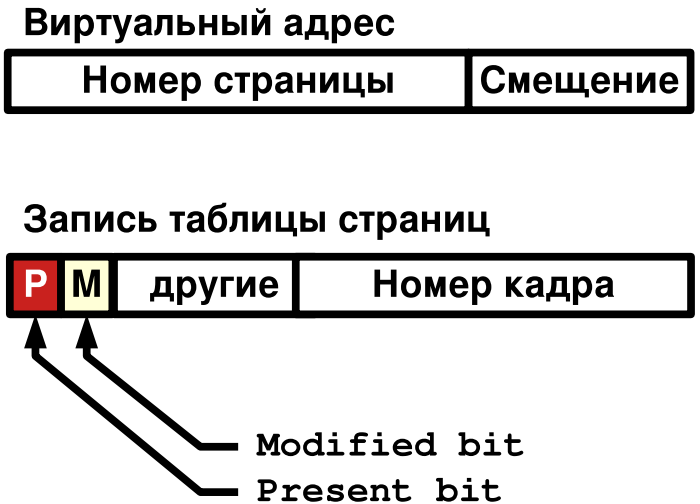
- Реальный (физический) адрес — адрес в основной памяти
- Виртуальный адрес — логический адрес внутри процесса
- Адресное пространство — диапазон адресов процесса
- Виртуальное адресное пространство — область для одного процесса с виртуальными адресами
- Виртуальная память — схема расположения процессов в памяти, в которой:
  - Вторичная память адресуется так-же как и основная
  - Виртуальные адреса транслируются в адреса в основной памяти
  - Основная память может быть расширена на вторичную
  - Размер памяти ограничен схемой адресации, но не фактическим количеством ячеек

Аппаратура и управляющие программы

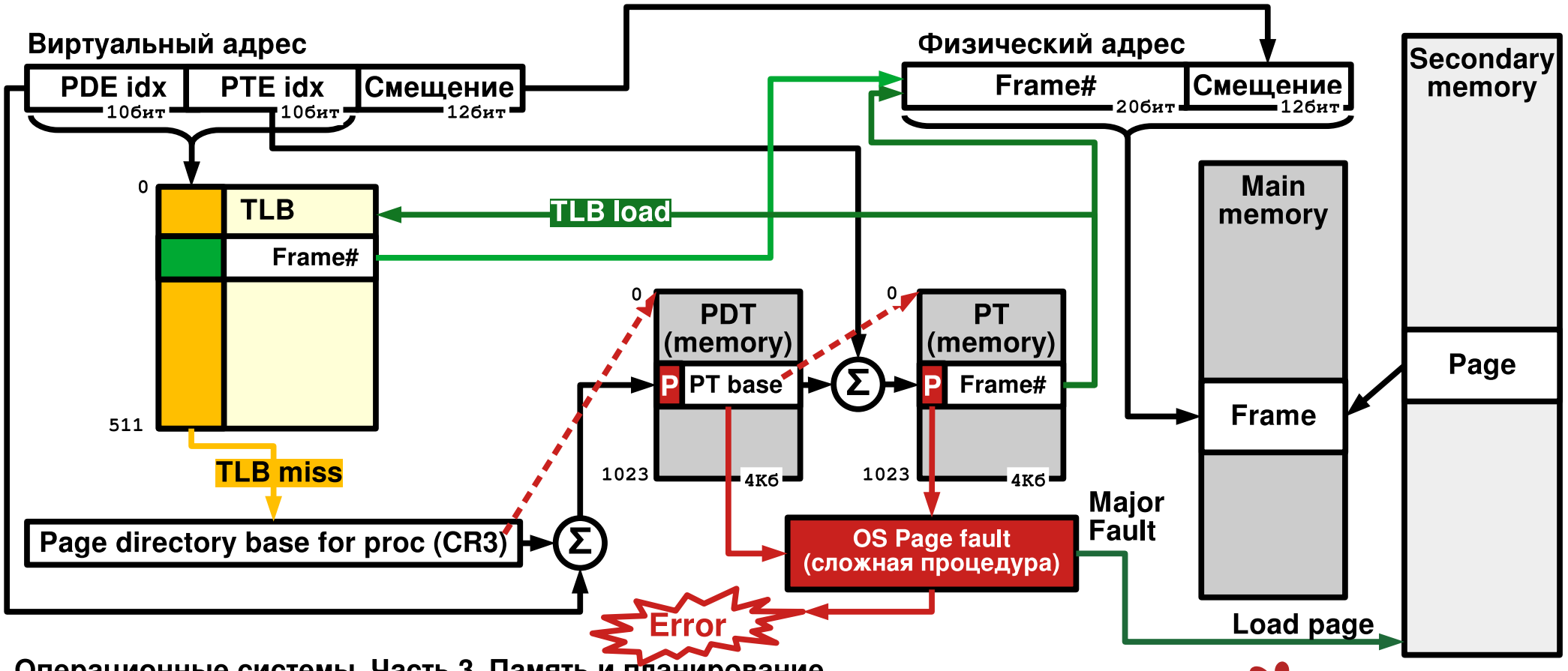
- Организация виртуальной памяти это совокупность аппаратных и программных средств
- Схемы реализации — совокупность страничной и сегментной адресации
  - Логические адреса динамически транслируются в физические
  - Сегменты и страницы не должны располагаться последовательно в основной памяти
  - Любая часть процесса в разные моменты выполнения может находится во вторичной памяти и менять физические адреса в основной
- Resident set (резидентная часть) — часть процесса, находящаяся в основной памяти
- Real memory — часть памяти, где процесс непосредственно выполняется
- Virtual memory — часть памяти, которую процесс может занять
- Следовательно, ОС может одновременно с бОльшим количеством процессов

Virtual memory paging

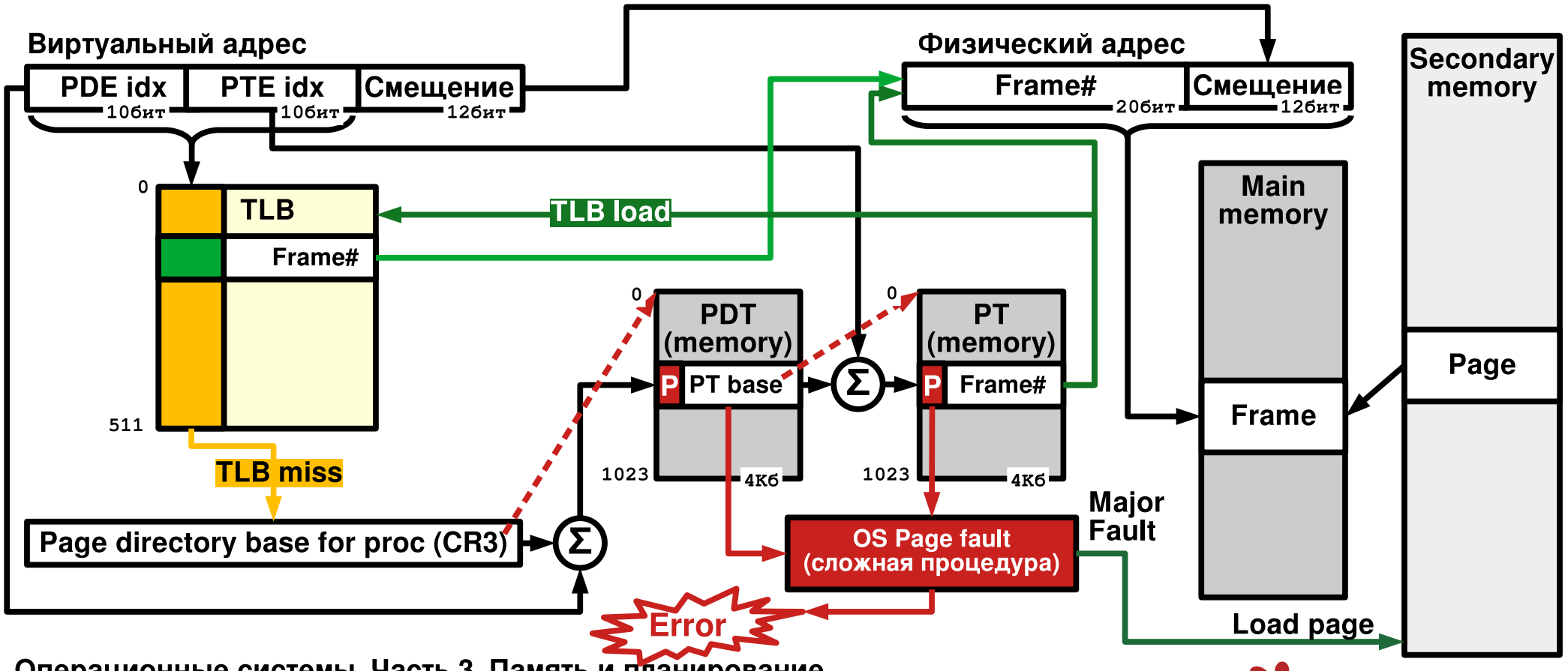
- Разные структуры организации
  - Одноуровневая
  - Двух- и многоуровневая
  - Инвертированная
- Таблицы страниц хранятся в основной памяти
  - Для обращения к памяти нужно сделать несколько служебных обращений к памяти
  - Используется TLB для ускорения



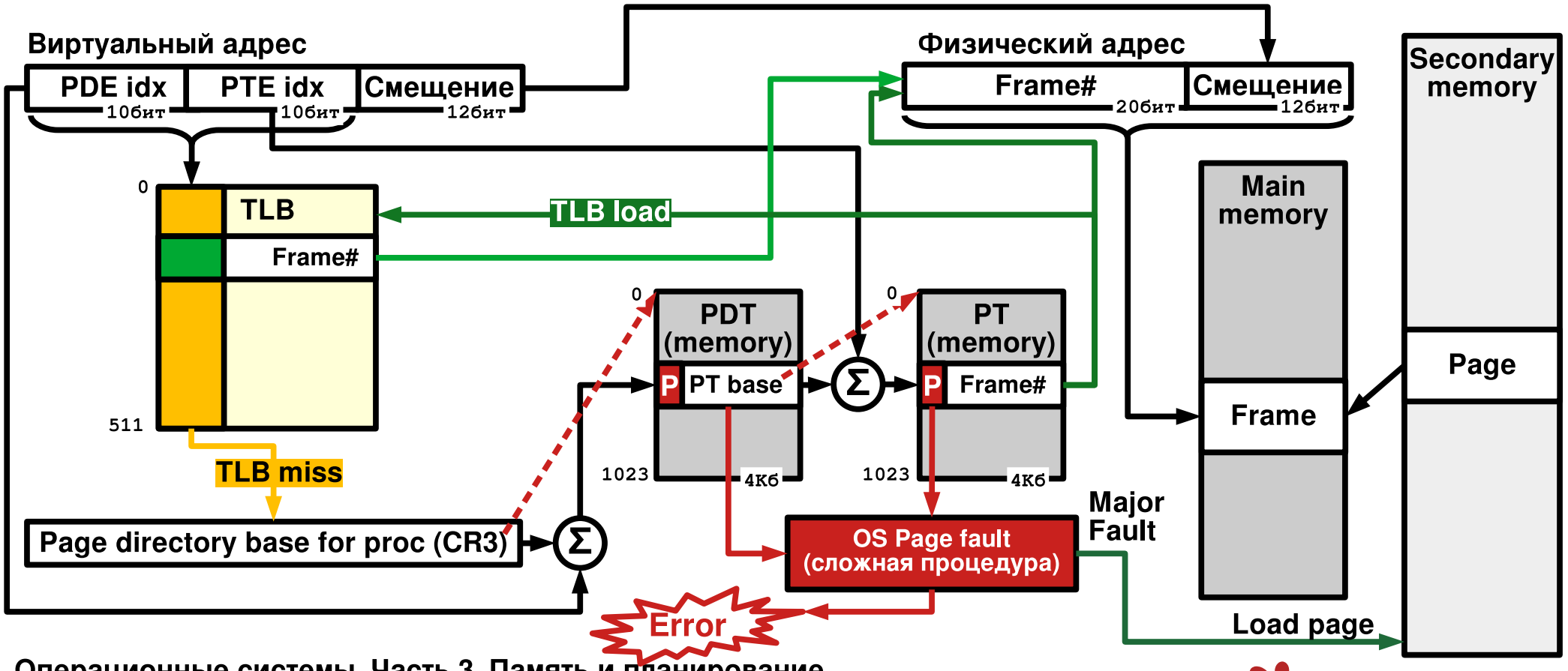
2 level - virtual memory paging схема с TLB



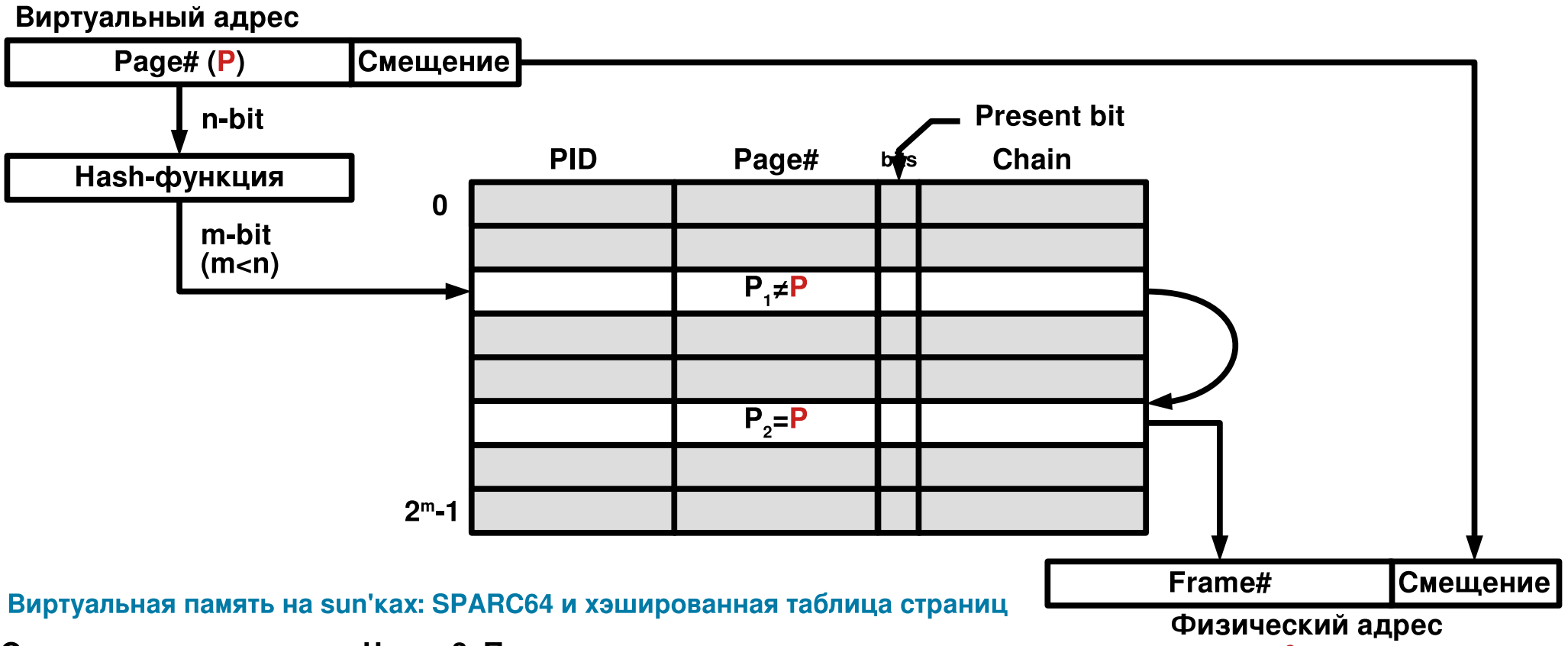
2 level - virtual memory paging схема с TLB



2 level - virtual memory paging схема с TLB

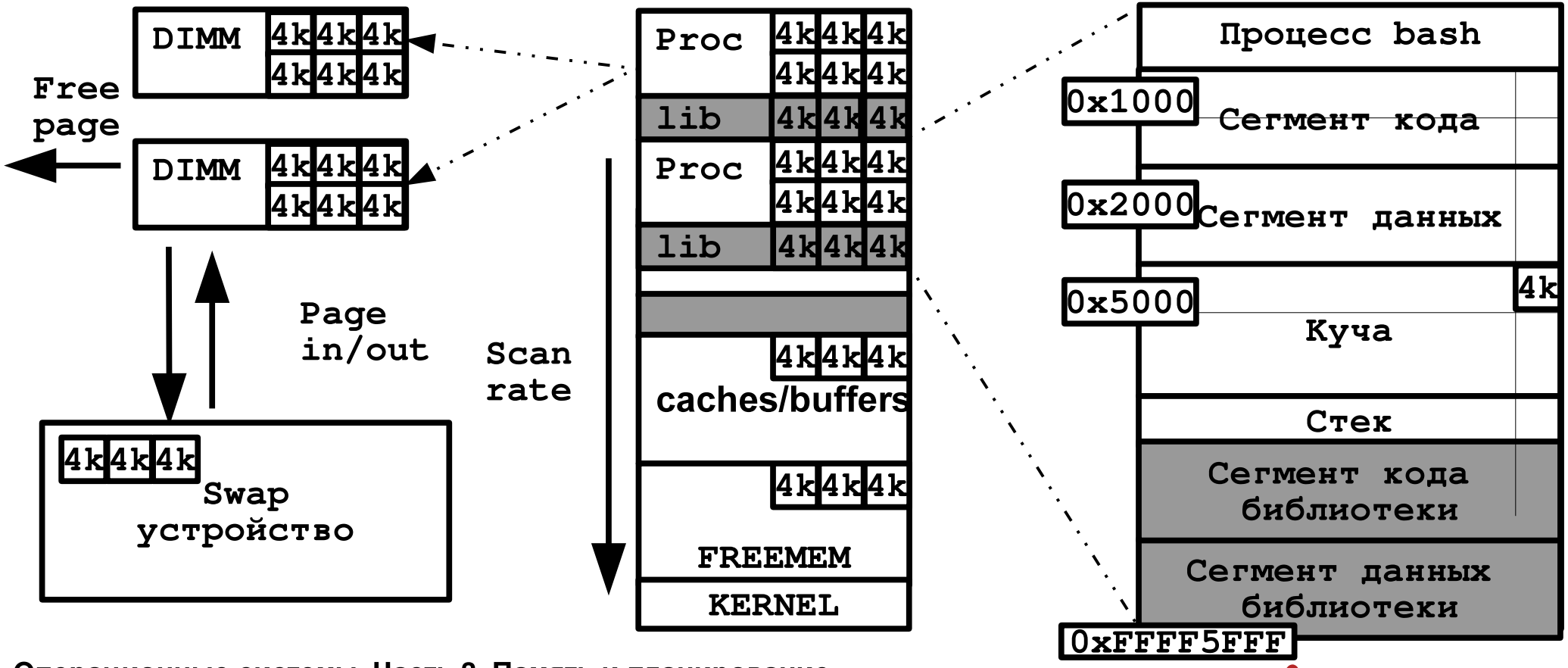


Инвертированная таблица страниц



Виртуальная память на sun'ках: SPARC64 и хэшированная таблица страниц

Сегментно-страничная виртуальная память



Размер страницы

- При разработке ОС необходимо определится с размером страницы. Важно учитывать:
  - Размеры таблиц страниц
  - Внутреннюю фрагментация
  - Количество page-fault при трансляции адреса
  - Скорость взаимодействия со вторичной памятью (и размер блока)
  - Локальность данных (в многопоточных приложениях ниже)
  - Количество промахов TLB, размер TLB, размер кэша L1, L2, L3, и др.
- Современные процессоры могут работать с разными размерами страниц (large, huge pages)
  - Intel 4Кб, 2Мб, 1Гб
  - Sparc64 VI 8Кб, 64Кб, 512Кб, 4Мб, 32Мб, 256Мб

Стратегии ОС по работе с виртуальной памятью

- Стратегия выборки страниц из вторичной памяти:
  - По требованию, предварительная выборка
- Стратегия размещения
  - В современных NUMA системах скорость выше рядом с процессором процесса/потока
- Стратегия очистки (выгрузки во вторичную память)
  - По требованию и предварительная очистка
- Управление многозадачностью
  - Выгрузка процессов целиком
- Стратегии замещения (см. далее)
- Управление резидентной частью процессов (см. далее)

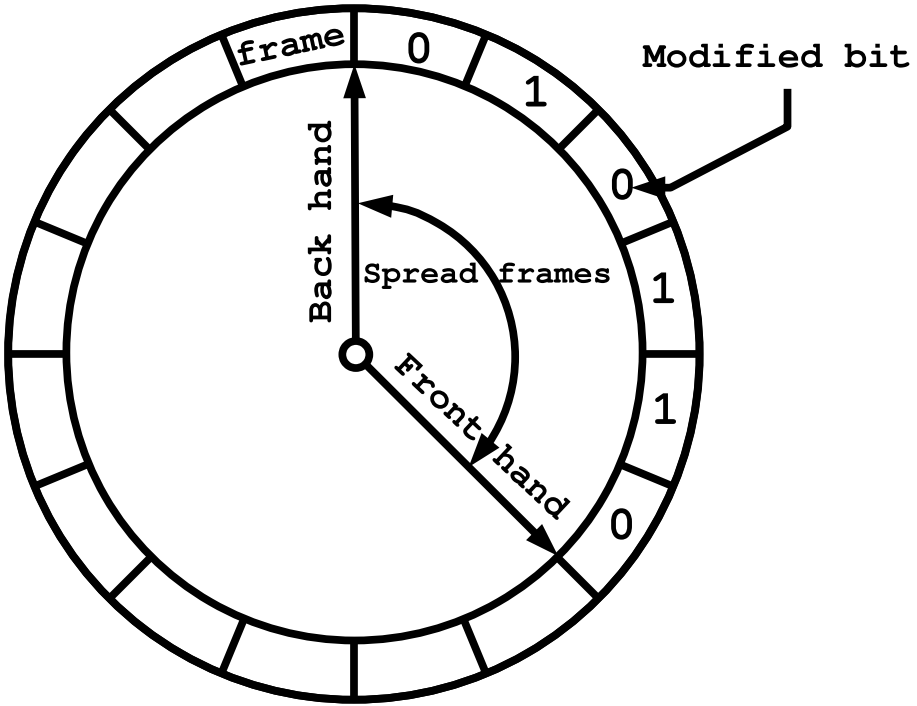
Стратегии замещения ОС

- Какие именно кадры работающего процесса нужно выбрать для замещения и использования другими процессами?
  - Выгрузим кадр, обращения к которому не последует в ближайшее время
  - Используется статистика прошлого поведения
  - Необходимо учитывать locked frames — не все страницы можно выгрузить (часть ядра, буферы, ...)
  - Если кадр совместно используется много раз (н-р код разделяемой библиотеки) то его выгрузка может повлиять на много процессов

Стратегии замещения ОС (2)

- «Оптимальная стратегия» - страница к которой обращение будет через наибольший промежуток времени
  - Мы не Пифии, не Оракулы мы.
  - Обычно используется для сравнения с остальными
- LRU (least recently used) — заменять наименее используемые страницы
  - Сложность определения наименее используемых
- FIFO — просто реализовать, но неэффективно
- Clock algorithm

Модифицированный Clock Algorith (Solaris 2.0 - 11)



- Алгоритм:
  - FH — сбрасывает бит модификации
  - Модификация фрейма устанавливает
  - BH — Проверяет
- Расстояние между стрелками меняется динамически
- На современных размерах памяти занимает много времени
  - 16Гб/4кб = 4194304 фреймов

Управление резидентной частью процессов

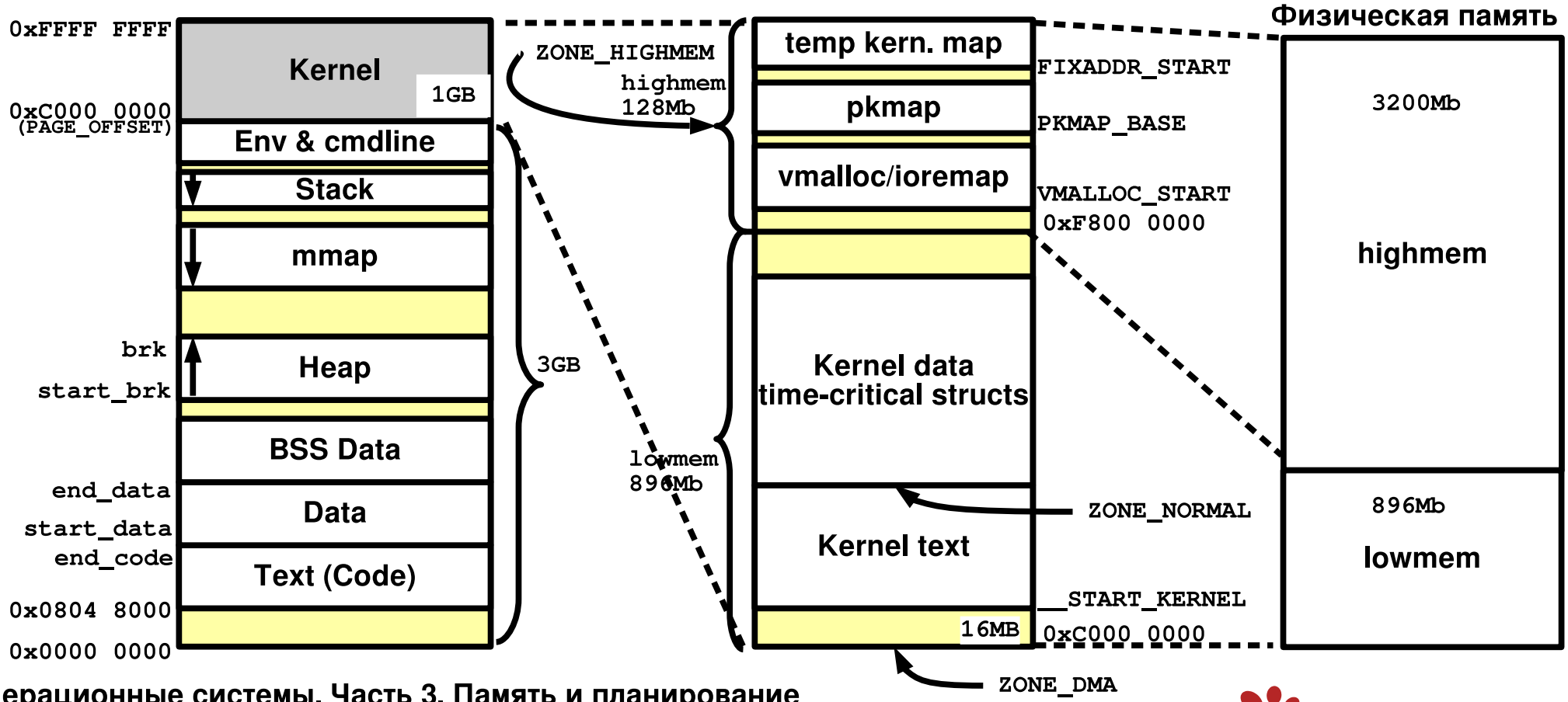
- Размер резидентной части:
  - При загрузке и работе процесса нет смысла держать в памяти все его страницы
  - Меньше памяти процессу → больше процессов в памяти
  - Меньше страниц процесса в памяти → больше pagefaults
  - После N-страниц дополнительное выделение памяти не приводит к кардинальному снижению pagefaults
  - Стратегии управления: Фиксированный и динамический размер
- Область видимости (scope)
  - Локальная — страница замещается у процесса вызвавшего pagefault
  - Глобальная — сканируются страницы всех процессов, кроме locked и сильно shared

Виртуальная память Linux

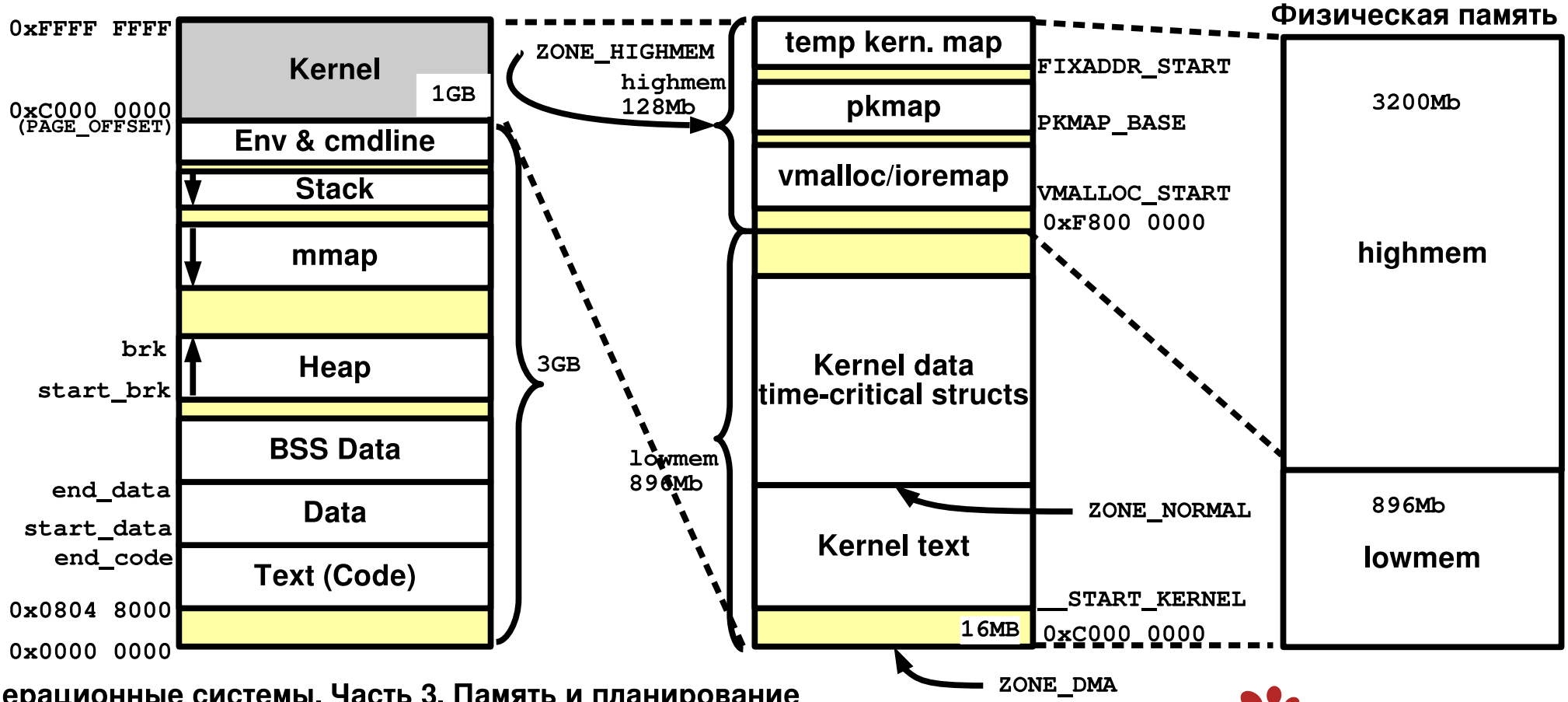
3.3

- Модели памяти
- Структуры памяти
- Выделения памяти на уровнях пользователя и ядра
- SLUB — allocator
- Copy On Write, pagefault
- Замещение страниц

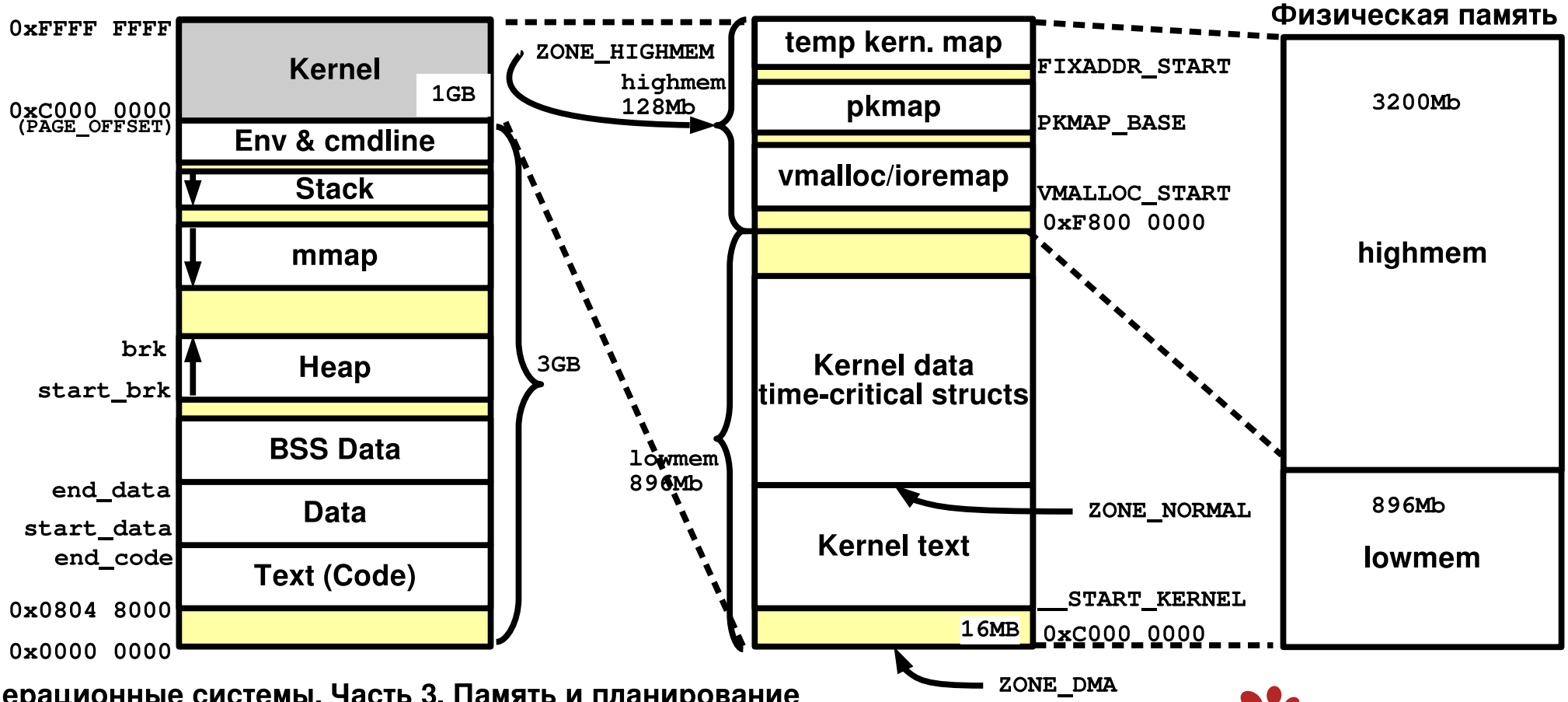
Memory layout 32bit



Memory layout 32bit

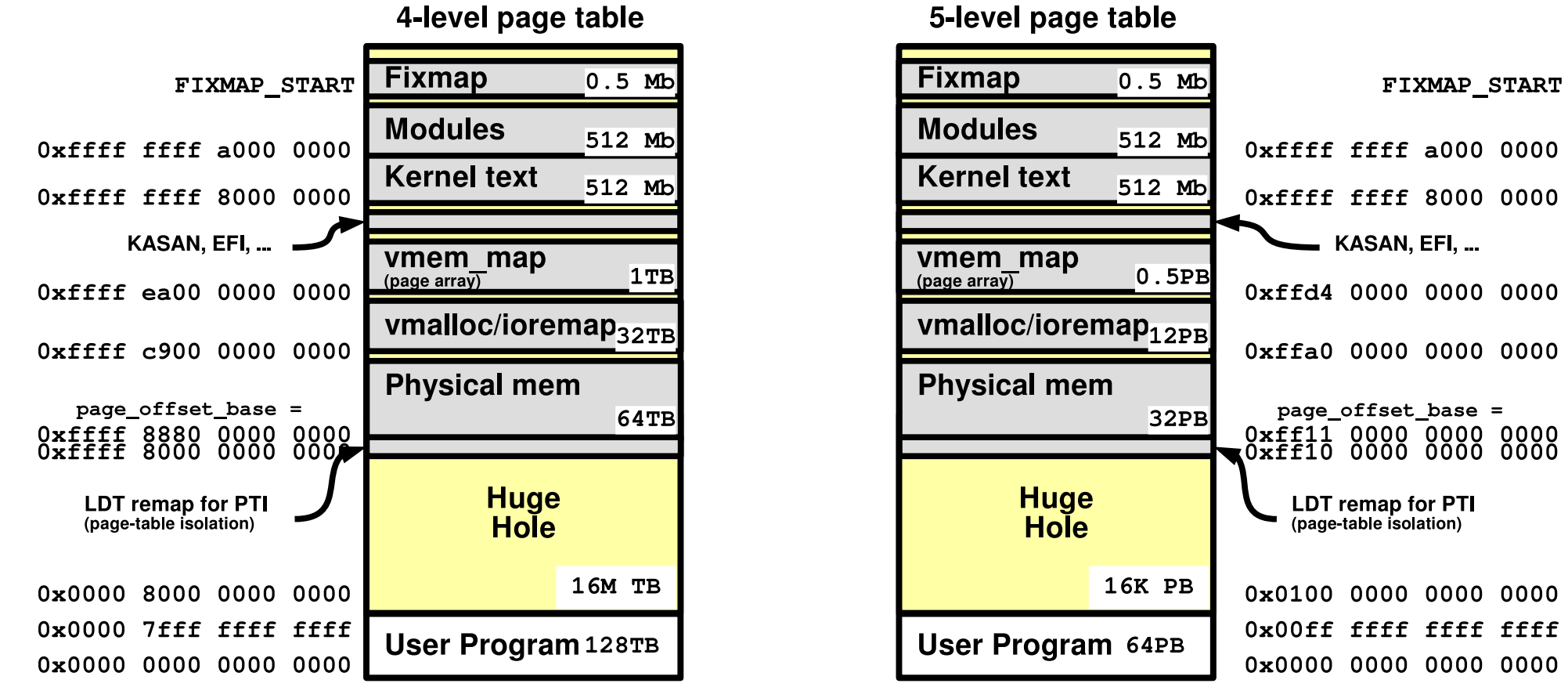


Memory layout 32bit



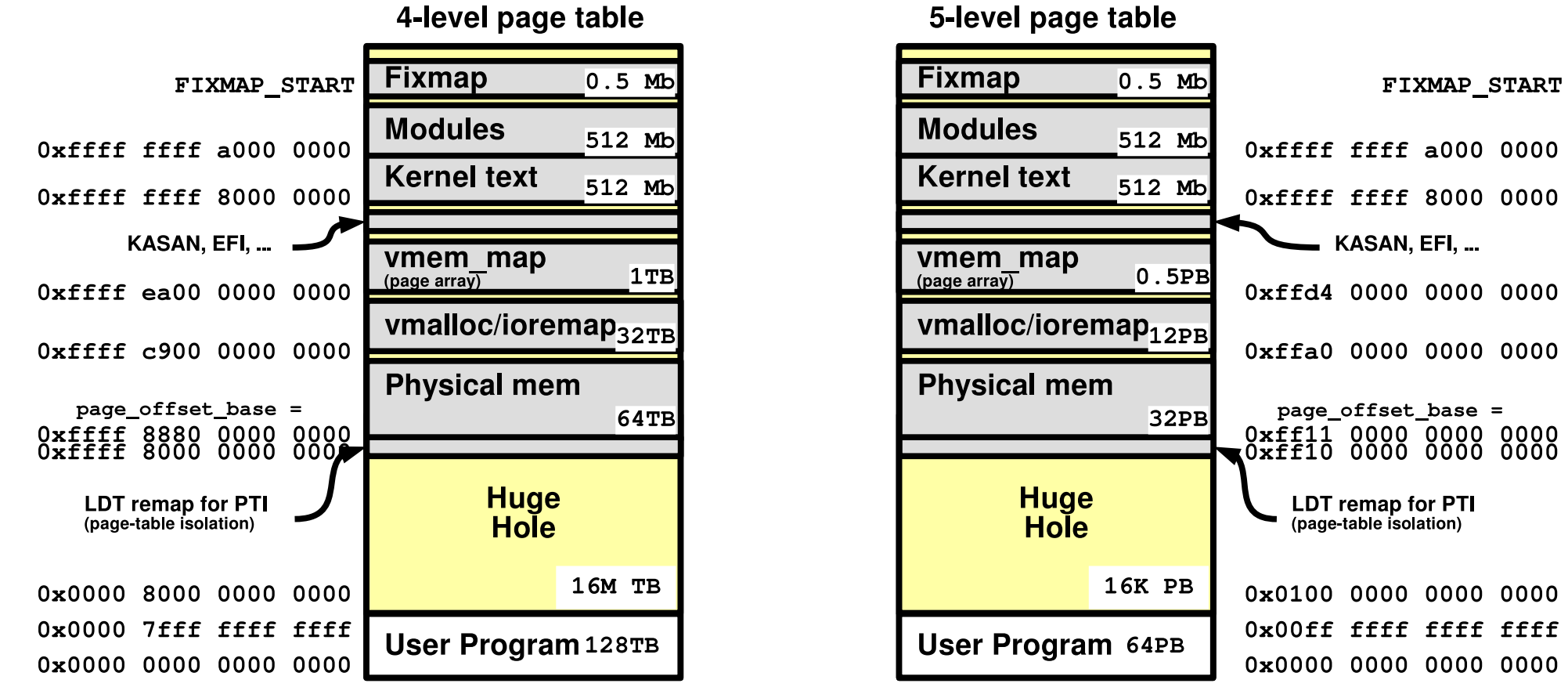
Memory Layout 64bit

https://elixir.bootlin.com/linux/v5.4.74/source/Documentation/x86/x86_64/mm.rst

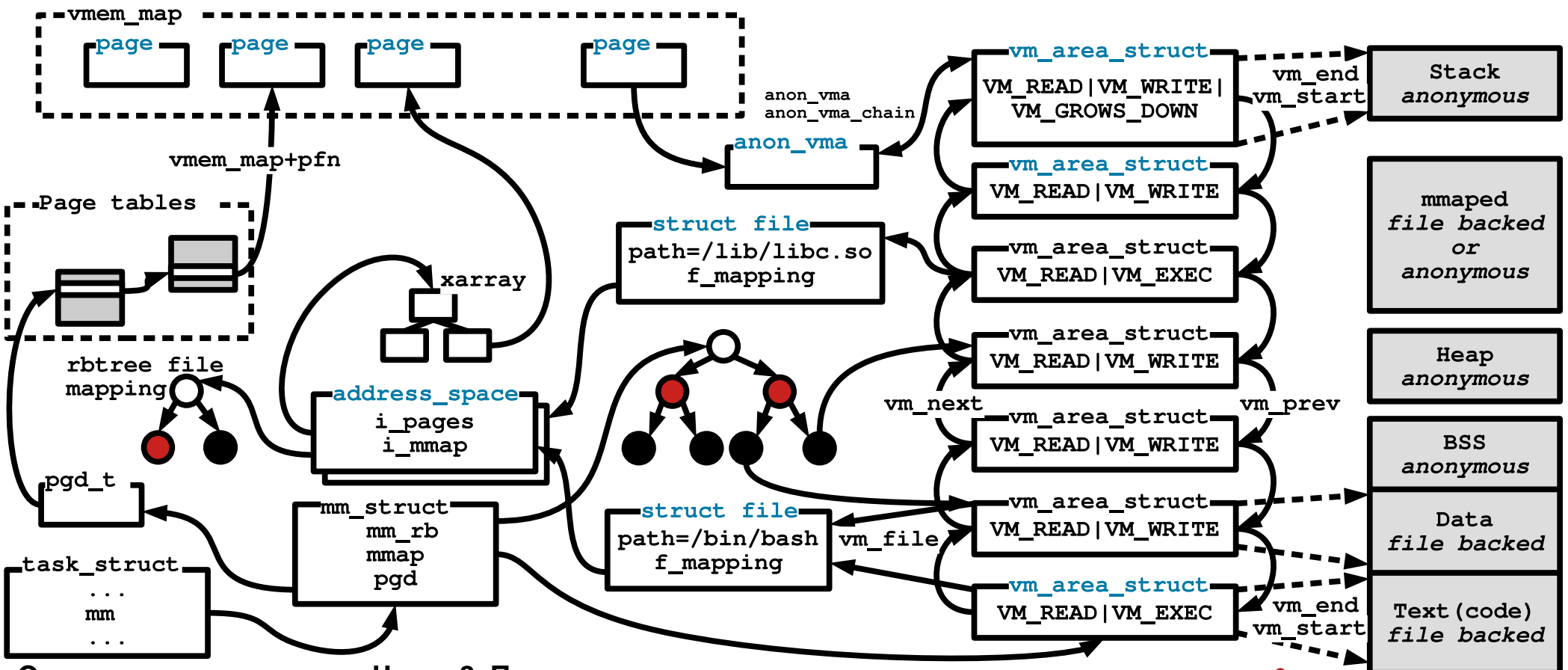


Memory Layout 64bit

https://elixir.bootlin.com/linux/v5.4.74/source/Documentation/x86/x86_64/mm.rst



Структуры памяти



Выделение памяти user-space

- Malloc
  - Несколько реализаций (buddy, glibc, ...)
  - При нехватке памяти двигает program break — sbrk()
- Mmap
  - Гибкий вызов создания областей памяти
  - Создает «file backed» и анонимную память

```
void *malloc(size_t size);
void free(void *ptr);
void *calloc(size_t nmemb, size_t size);
void *realloc(void *ptr, size_t size);
void *reallocarray(void *ptr,
      size_t nmemb, size_t size);
```

```
void *mmap(void *addr, size_t length,
           int prot, int flags,
           int fd, off_t offset);
int munmap(void *addr, size_t length);
```

```
PROT_NONE
PROT_EXEC
PROT_READ
PROT_WRITE
```

```
MAP_SHARED
MAP_PRIVATE
MAP_32BIT
MAP_ANONYMOUS
MAP_FIXED
MAP_GROWSDOWN
MAP_HUGETLB
MAP_HUGE_2MB
MAP_HUGE_1GB
```

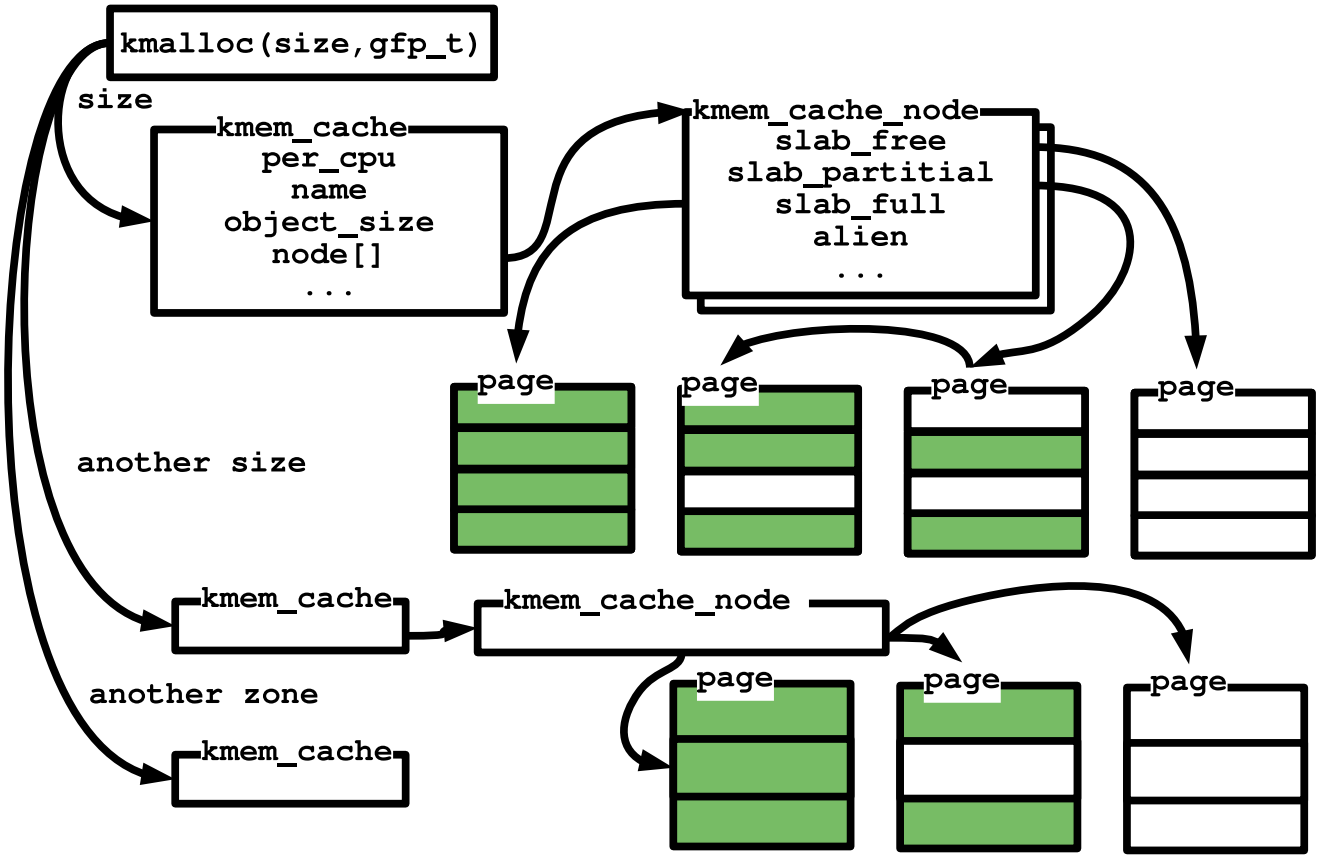
```
MAP_LOCKED
MAP_NONBLOCK
MAP_NORESERVE
MAP_POPULATE
MAP_STACK
MAP_UNINIALIZED
```

Выделение памяти в пространстве ядра

- Выделение страниц
  - page *alloc_pages(gfp_t flags,order) — выделяет 2^{order} последовательных станиц в памяти, возвращает struct page; free_pages() - освобождение;
    __get_free_pages(gfp_t flags ,order) - возвращает адрес
- Динамическая аллокация
  - vmalloc(unsigned long size),vzalloc() - выделение непрерывной области из нескольких страниц, достаточных для size; vmalloc_user() - выделение обнуленный страниц в пространстве пользователя
  - ioremap(), iounmap() - мапинг области ввода-вывода в физической памяти в область виртуальной памяти
  - page_frag_alloc(), page_frag_free() - выделение фрагментов страниц, оптимизация для сетевых драйверов
  - kmalloc(size, gfp_t flags), kfree() — выделеление памяти меньшей чем page в различных зонах ядра (использует SLAB-аллокатор)
- Отображение
  - kmap(page), kunmap(page) — отобразить страницу в highmem,
    kmap_atomic(page), kunmap_atomic(kvaddr) — отобразить в FIXMAP, не блокируется!

SLAB/SLUB/SLOB allocators

- Несколько реализаций, в Linux осталось 2: SLOB и SLUB
- Быстрое управление небольшими объектами
- В качестве бекэнда используется alloc_page



COW — Copy On Write

- Механизм использования данных в случае их записи
  - В реальной нагрузке процесс может умереть быстрее, чем изменит страницу данных
  - Давайте при fork создавать *мапинги,* а не сами страницы
  - В случае обращения на чтение — просто прочитаем
  - В случае записи — создадим копию страницы для порожденного процесса (и anonymous mapping), и потом запишем
- COW порождает множество совместно используемых для чтения страниц, и копирует их при записи

do_page_fault

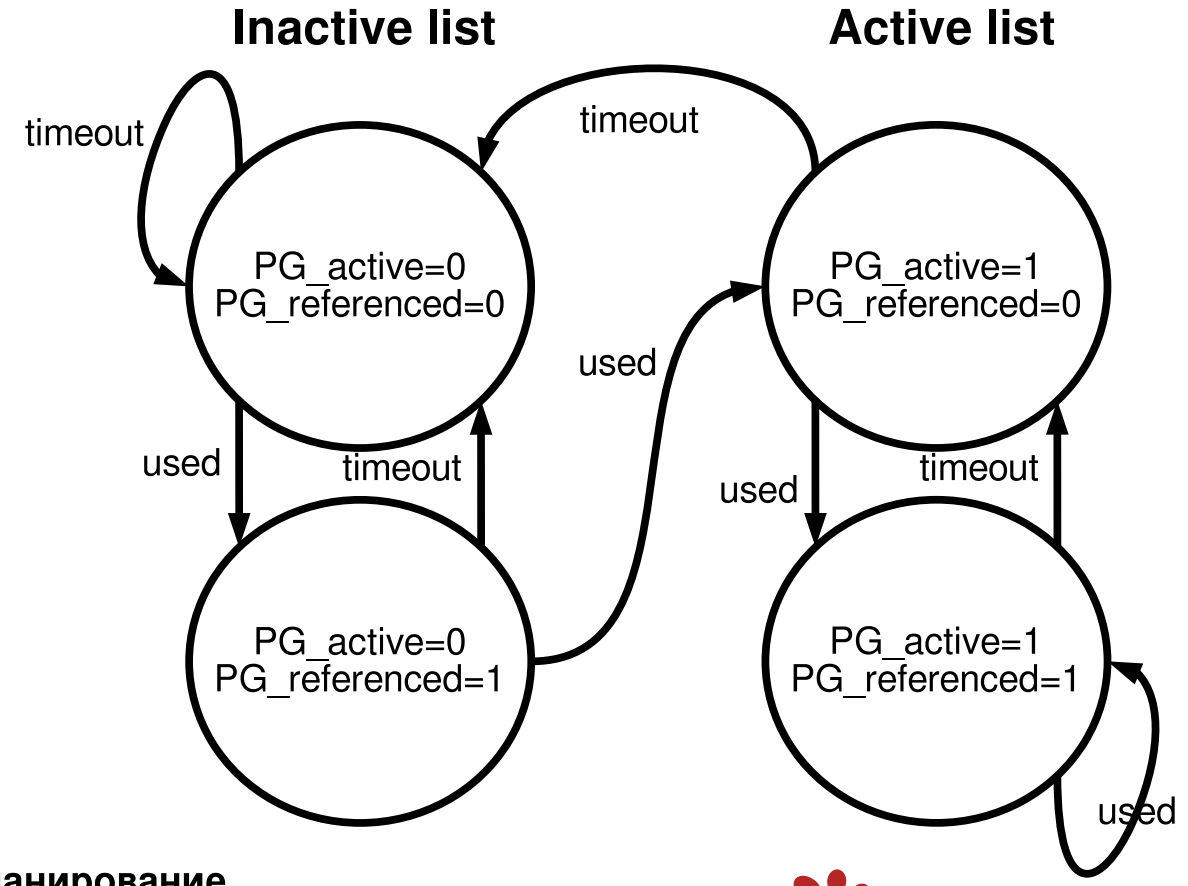
- Обращение к странице, которой нет в требуемом сегменте памяти
  - Minor fault — на самом деле нужные данные в памяти есть, но по разным причинам недоступны для текущего процесса
  - Major fault — frame выгружен из памяти
    - Очищен (выгружен) для страниц кода, статических данных, …
    - Находится в области подкачки для анонимных страниц, …
  - Segmentation fault — если мы обращаемся в запрещенную область
  - Kernel panic — если мы обращаемся в запрещенную область из ядра
- Может быть вызвано дефектом кода, сбоем аппаратуры, ….
- Основная функция do_page_fault

do_page_fault

- Обращение к странице, которой нет в требуемом сегменте памяти
  - Minor fault — на самом деле нужные данные в памяти есть, но по разным причинам недоступны для текущего процесса
  - Major fault — frame выгружен из памяти
    - Очищен (выгружен) для страниц кода, статических данных, …
    - Находится в области подкачки для анонимных страниц, …
  - Segmentation fault — если мы обращаемся в запрещенную область
  - Kernel panic — если мы обращаемся в запрещенную область из ядра
- Может быть вызвано дефектом кода, сбоем аппаратуры, ….
- Основная функция do_page_fault

Замещение страниц Linux. kswapd

- Для каждой зоны свой набор из двух списков page
- Кандидат на замещение — page у которой
  - PG_active=PG_referenced=0
- kswapd запускается при нехватке памяти в заданной зоне
- Из списков исключаются страницы: SHM_LOCK, VM_LOCKED, ramfs



Виртуальная память Windows

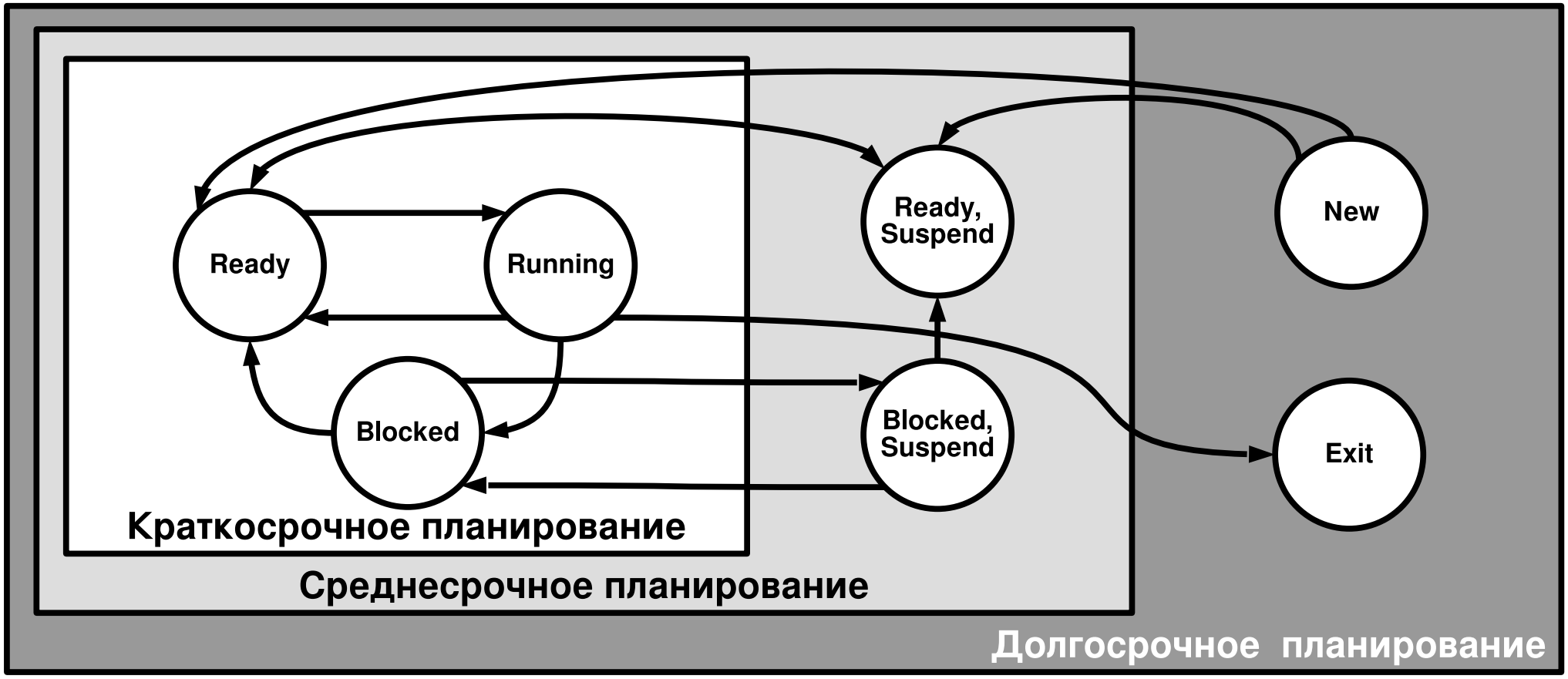
3.4

Планирование

3.5

- Сроки планирования
- Критерии планирования
- Приоритеты
- Стратегии планирования
- Классы планирования

Виды планирования



Критерии краткосрочного планирования

- Пользовательские, связанные с производительностью
  - Turnaround time — время оборота
  - Responce time — время отклика
  - Deadline — предельное время
- Пользовательские, иные
  - Predictability — предсказуемость (независимость от остальной загрузки системы)
- Системные, связанные с производительностью
  - Throughput — пропускная способность
  - Processor Utilization — загуженность процесора
- Системные, иные
  - Fairness — справедливость
  - Enforcing priorities — принудительная приоритезация
  - Balancing resources - сбаласированная загрузка

Использование приоритетов

- Процессам присваивается цифровой приоритет
  - В разных ОС - разные схемы назначения приоритетов
- Из очередей выбирается процесс с наивысшим приоритетом
- Если приоритеты процессов совпадают, то используется дополнительная стратегия
- Процессы с низким приоритетом могут голодать
- Приоритеты могут динамически изменяться

Стратегии планирования

- First Come First Served (FCFS) — аналог FIFO
- Round Robin — карусельное планирование
- Shortest Process Next — Короткие процессы вперед
- Shortest Remaining Time — Наименьшее время до завершения
- Highest Response Ratio Next — Наивысшее отношение отклика
- Feedback — снижения приоритета в зависимости от длительности времени исполнения
  - в случае blocked или preempted — feed it back

Сравнение стратегий

	FCFS	RR	SPN	SRT	HRRN	Feedback
Функция выбора	max(w)	Const TQ	min(s)	min(s-e)	max($\frac{w+s}{s}$)	$w\uparrow \rightarrow prio\uparrow$ $e\uparrow \rightarrow prio\downarrow$
Preemption	No	At Time Quantum	No	At arrival to ready queue	No	At Time quantum
Throughtput	-	Can be low if TQ low	High	High	High	-
Responce time	Can be high	Good for short	Good for short	Good	Good	-
Overhead	Minimum	Minimum	Can be high	Can be high	Can be high	Can be high
Effect	Bad for short and high I/O	fair	Bad for long	Bad for long	Balanced	Can preffer high IO proc.
Starvation	No	No	Possible	Possible	No	Possible

w — общее время ожидания процесса в очередях, e — общее время выполнения,
s — общее время, предполагаемое или заданное, для обслуживания процесса, включая e

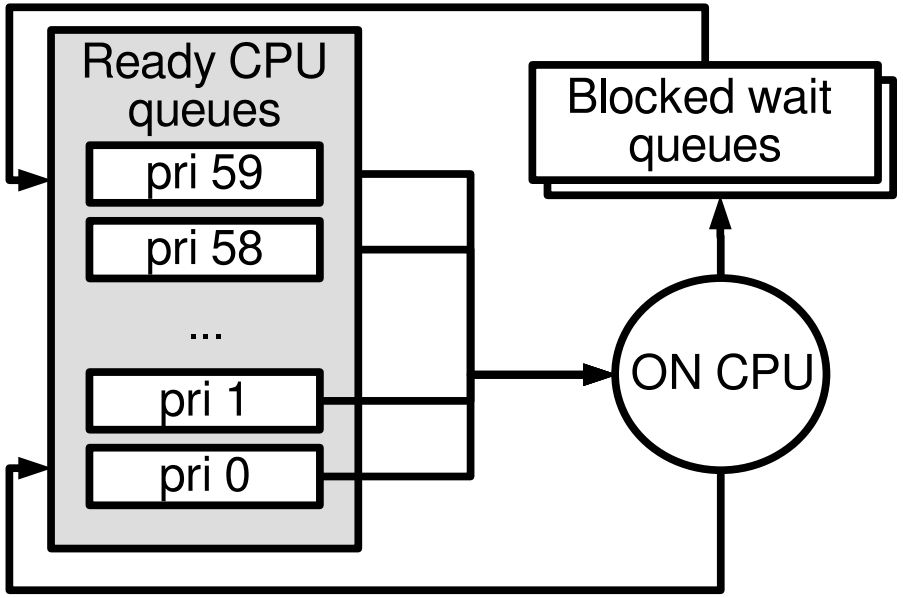
Сравнение стратегий

	FCFS	RR	SPN	SRT	HRRN	Feedback
Функция выбора	max(w)	Const TQ	min(s)	min(s-e)	max($\frac{w+s}{s}$)	$w\uparrow \rightarrow prio\uparrow$ $e\uparrow \rightarrow prio\downarrow$
Preemption	No	At Time Quantum	No	At arrival to ready queue	No	At Time quantum
Throughtput	-	Can be low if TQ low	High	High	High	-
Responce time	Can be high	Good for short	Good for short	Good	Good	-
Overhead	Minimum	Minimum	Can be high	Can be high	Can be high	Can be high
Effect	Bad for short and high I/O	fair	Bad for long	Bad for long	Balanced	Can preffer high IO proc.
Starvation	No	No	Possible	Possible	No	Possible

w — общее время ожидания процесса в очередях, e — общее время выполнения,
s — общее время, предполагаемое или заданное, для обслуживания процесса, включая e

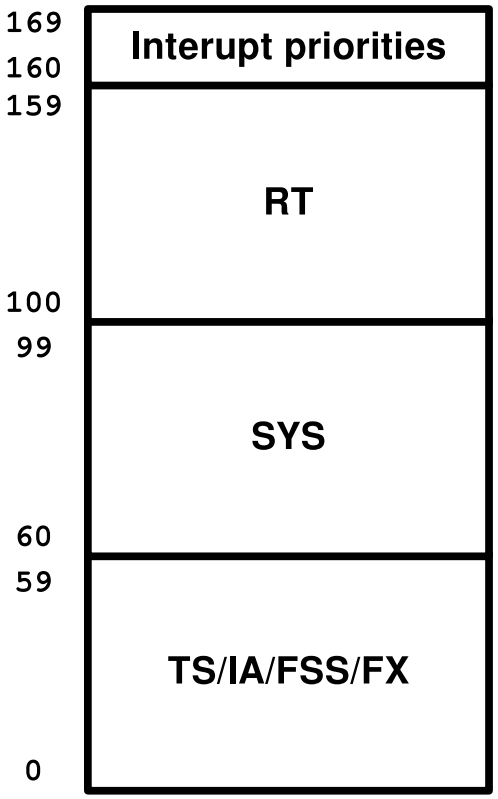
Feedback on SVR4 (Solaris) TimeSharing class

globpri	quantum	tqexp	slpret	maxwait	lwait
59	2	49	59	32000000	59
58	4	48	58	0	59
57	4	47	58	0	59
...					
40	4	30	55	0	55
...					
30	8	20	53	0	53
29	12	19	52	0	52
...					
21	12	11	52	0	52
20	12	10	52	0	52
19	16	9	51	0	51
...					
12	16	2	51	0	51
11	16	1	51	0	51
10	16	0	51	0	51
9	20	0	50	0	50
...					
1	20	0	50	0	50



globpri - глобальный приоритет,
quantum - выделяемый квант времени (мс),
tqexp - приоритет, если квант полностью выбран CPU,
slpret - приоритет, если ожидаем ресурса
maxwait — граничное время длинного ожидания (мс)
lwait - прирритет, в случае длинного ожидания ресурса

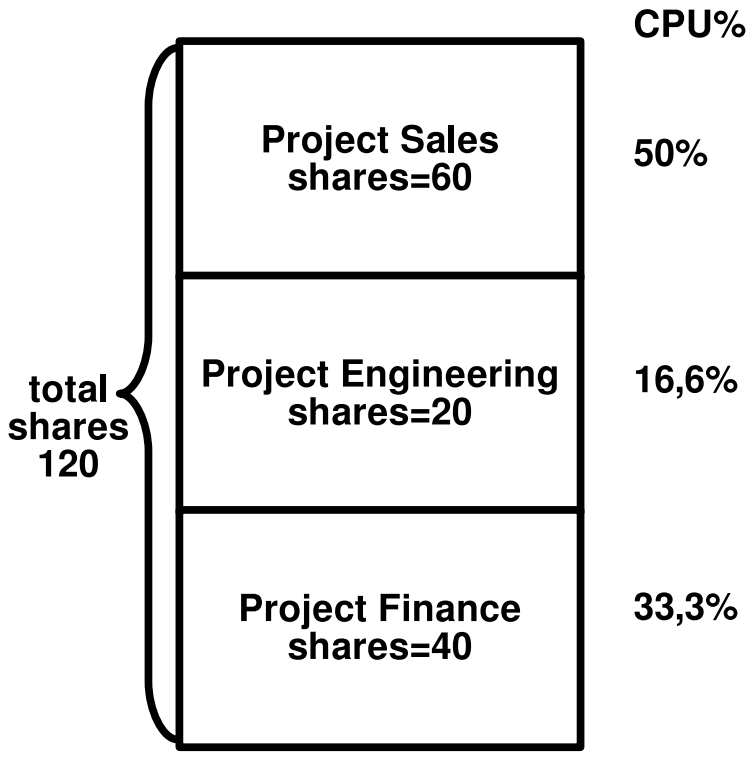
Классы планирования SVR4



- TimeSharing — разделение времени (Feedback)
- InterActive — интерактивный (boost к активному приложению)
- Fixed, System, RealTime — классы с фиксированными приоритетами
- Fair Share Scheduler — справедливый планировщик
- Отдельные приоритеты для Interrupt threads
- Учет афинити для NUMA

Справедливое планирование SVR4

- Выделяет время процессора на основании заданных user shares
- Используется как замена TS/IA классов
- Учитывает все процессоры в системе



Многопроцессорное Планирование

3.6

- Типы многопроцессорных систем
- Гранулярность
- Вопросы планирования
- Планирование реального времени
- Инверсия приоритетов

Типы многопроцессорных систем

- Слабосвязанные (распределенные, кластеры)
  - Набор систем со своей основной памятью и подсистемой ввода-вывода
  - Распределение заданий между системами
  - Распределение общих данных и результатов
- Функционально-специализированные
  - Ведущий процессор выполняет координацию
  - Ведомые процессоры выполняют вычисления
- Сильносвязанные процессоры
  - Имеют общую память и систему кэшей
  - Управляются одной ОС

Synchronization Granularity

- Гранулярность синхнхронизации — частота синхронизации между процессами в системе
- Fine (тонкая) — параллельные вычисления на уровне отдельных машинных команд
  - Менее 20 команд
- Medium (средняя) — на уровне одного приложения
  - 20-200 команд
- Coarse (губая) — на уровне взаимодействующих процессов
  - 200-2000 команд
- Very Coarse (очень грубая) — на уровне распределенных систем
  - 2000-1 000 000
- Independent (независимая) — нет синхронизации, независимые процессы

Вопросы проектирования планировщиков в случае (сильно) многопроцессорных систем

- Назначение процессов процессорам
  - Статическое — выделяем процессор для процесса
  - Динамическое — общая очередь для всех процессов
  - Динамическая балансировка нагрузки
- Использование многозадачности на отдельных процессорах
  - Нужна ли для статического назначения многозадачность?
- Диспетчеризация процесса
  - Так ли плох будет FCFS? Влияние выбора алгоритма диспетчеризации снижается.

Подходы к планированию потоков

- Load Sharing — глобальная очередь потоков
  - Равномерное распределение нагрузки
  - Нет централизованного планировщика
  - Минусы — блокировки центральной очереди, промахи кэшей, неэффективность при тонкой средней гранулярности
- Gang scheduling — связанные потоки распределяются на связанные процессы по одному на процессор
  - Снижение накладных расходов на планирование при тонкой и средней гранулярности
- Dedicated processor assignment — назначается пул процессоров равный количеству потоков
  - В экстремальном случае увеличивает простои процессора
  - Полное устранение переключений повышает скорость работы
- Динамическое планирование
  - В отсутствии свободных CPU ресурсы изымаются из процесса, использующего несколько CPU

ОС реального времени

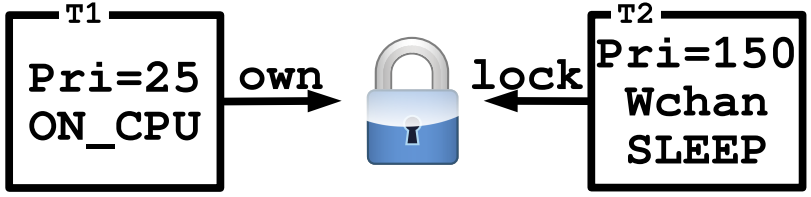
- Hard & Soft realtime, Периодичность работы
- Основные Требования
  - Determinism — выполнение операций в предопределенный интервал времени. Мера: время от прерывания до начала его обработки.
  - Responsiveness — сколько времени требуется для ответа?
  - User control — управление планированием со стороны пользователя
  - Reliability — поышенные требования по сравнению с «обычными» ОС
  - Fail-soft operation — мягкая реакция на ошибки (не kernel dump)
- Планировщик: строгое использование приоритетов с вытеснением и ограниченные, минимальные задержки

Deadline планирование

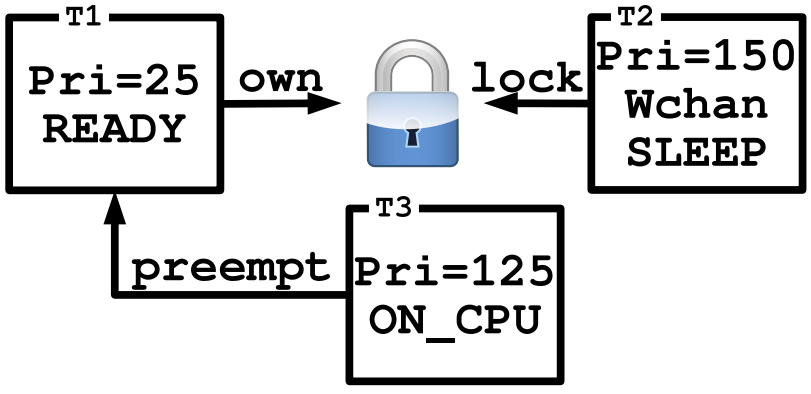
- Важены своевременные завершение или начало выполнения задания
  - Конфликты, сбои и временные недостатки ресурсов не должны влиять
- Задания могут включать дополнительную информацию:
  - Ready time — время готовности задания к выполнению
  - Starting deadline — предельное время начала выполнения
  - Compleation deadline — предельное время полного завершения задания
  - Processing time — время, необходимое для полного выполнения задания
  - Resource requirements — список ресурсов (не процессор) для задания
  - Priority — мера важности задания
  - Subtask structure — обязательные и необязательные задачи
- Rate Monotonic Sceduling — см. Столлингс гл. 10.2

Инверсия приоритетов

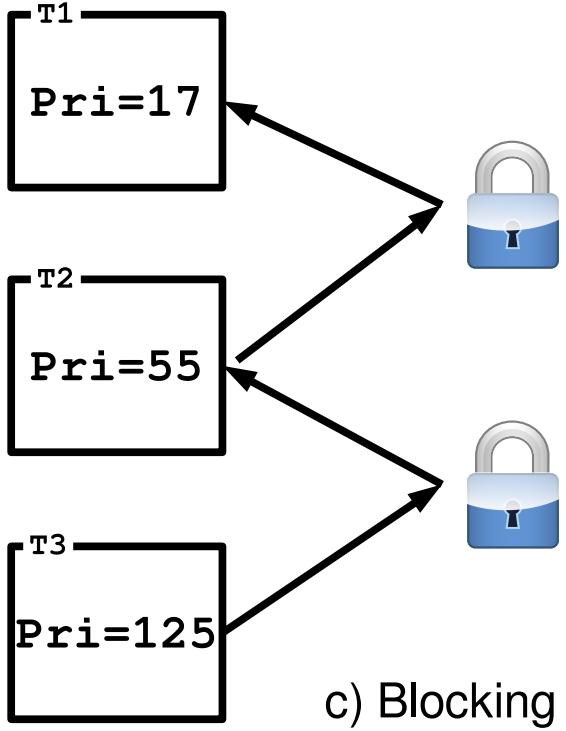
- Решается при помощи наследования приоритетов



a) Bounded priority inversion



b) Unbounded priority inversion



c) Blocking chain

Планирование Linux

Планирование Windows

3.8