Хабаровский Государственный Технический Университет

Институт Информационных Технологий

(Khabarovsk State University of Technology)
(Institute of Information Technologies)

University of Saarland
Department of Computer Science

Документирование среды программирования для операционной системы L4Ka

(Documenting the programmer environment for L4ka)

Yury Chebiryak
urriy@wjpserver.cs.uni-sb.de

Цель работы

Главными задачами диплома являются:

- документирование системных вызовов (system calls) среды программирования операционной системы L4Ka
- создание полноценного руководства программиста
- расширение функциональности менеджера памяти σ_0
- написание процедуры обработки аппаратных прерываний (interrupt service routine)

Работа раскрывает два аспекта: управление памятью (memory management) и взаимодействие между процессами (ВМП) (IPC – inter-process communication)

Управление памятью

- выделение (mapping) регионов физической памяти
- выделение/освобождение (unmapping) и передача (granting) регионов памяти между задачами (tasks)

Взаимодействие между процессами

- обмен сообщениями между задачами (tasks) и нитями (threads)
- написание сервиса для аппаратных прерываний (interrupt service routine)
- расширение функциональности менеджера памяти σ_0

Операционные системы (ОС)

ОС – это программа, предназначенная для управления всеми ресурсами вычислительной машины с целью предоставления возможности пользователям эффективно решать прикладные задачи

Классификация по структуре ядра (kernel):

- операционные системы без ядра (MS-DOS)
- операционные системы с макро-ядром (Unix, Windows)
- операционные системы с микро-ядром (L4, Mac OS X, Mach, Nomad OS)

Схема ОС с макро-ядром

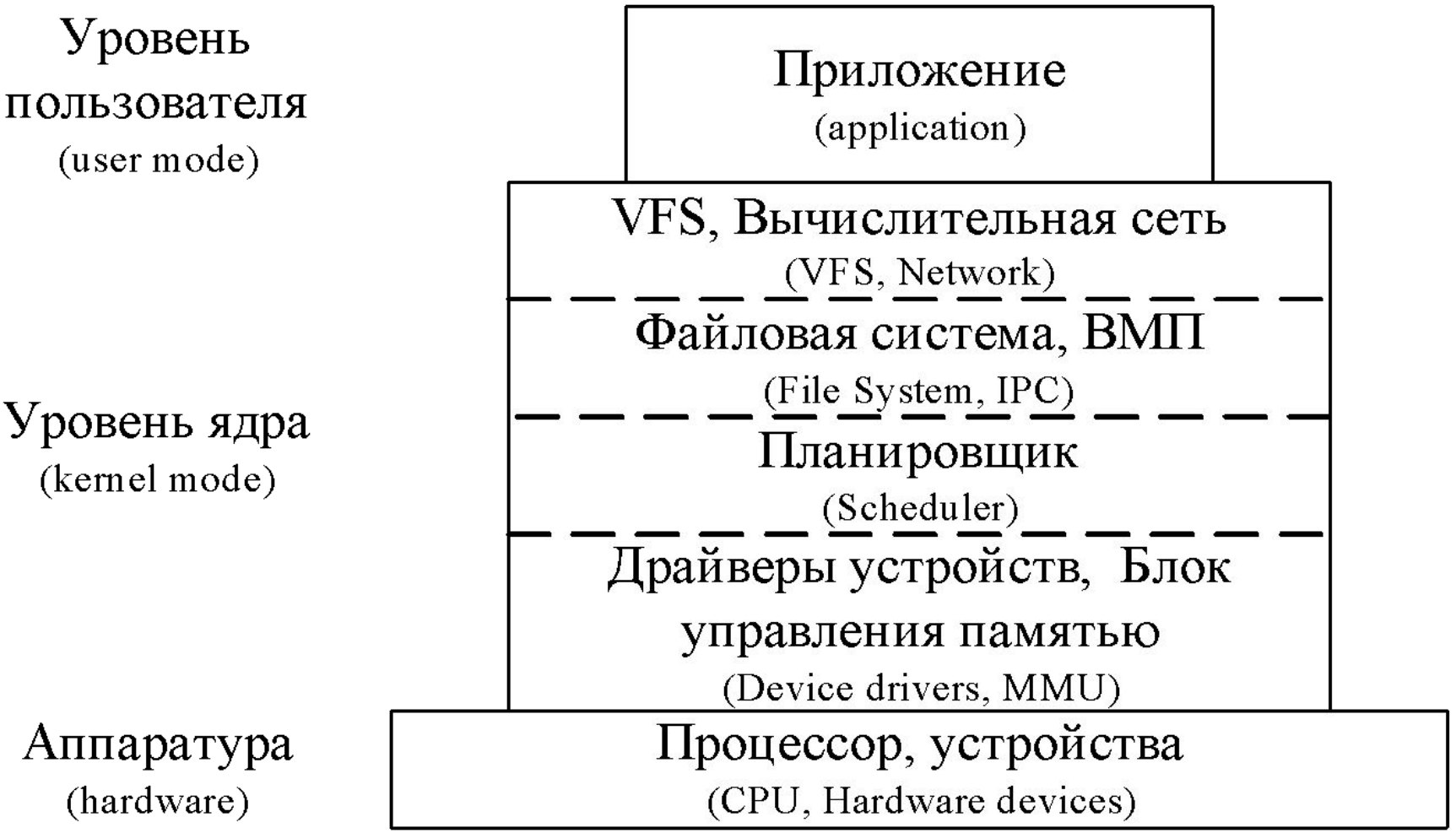
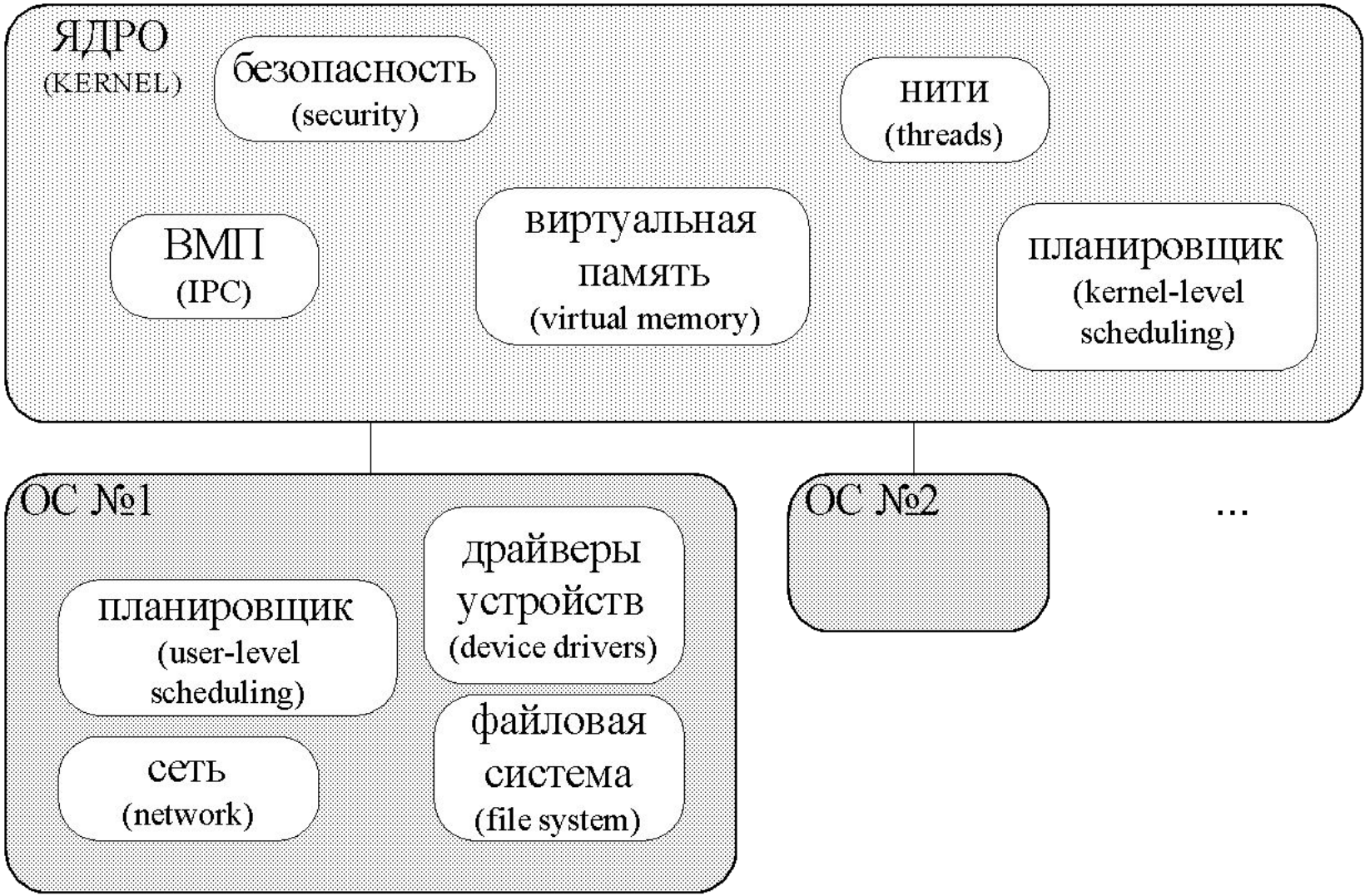


Схема ОС с микро-ядром



L4Ka – ОС с микро-ядром

Основные механизмы:

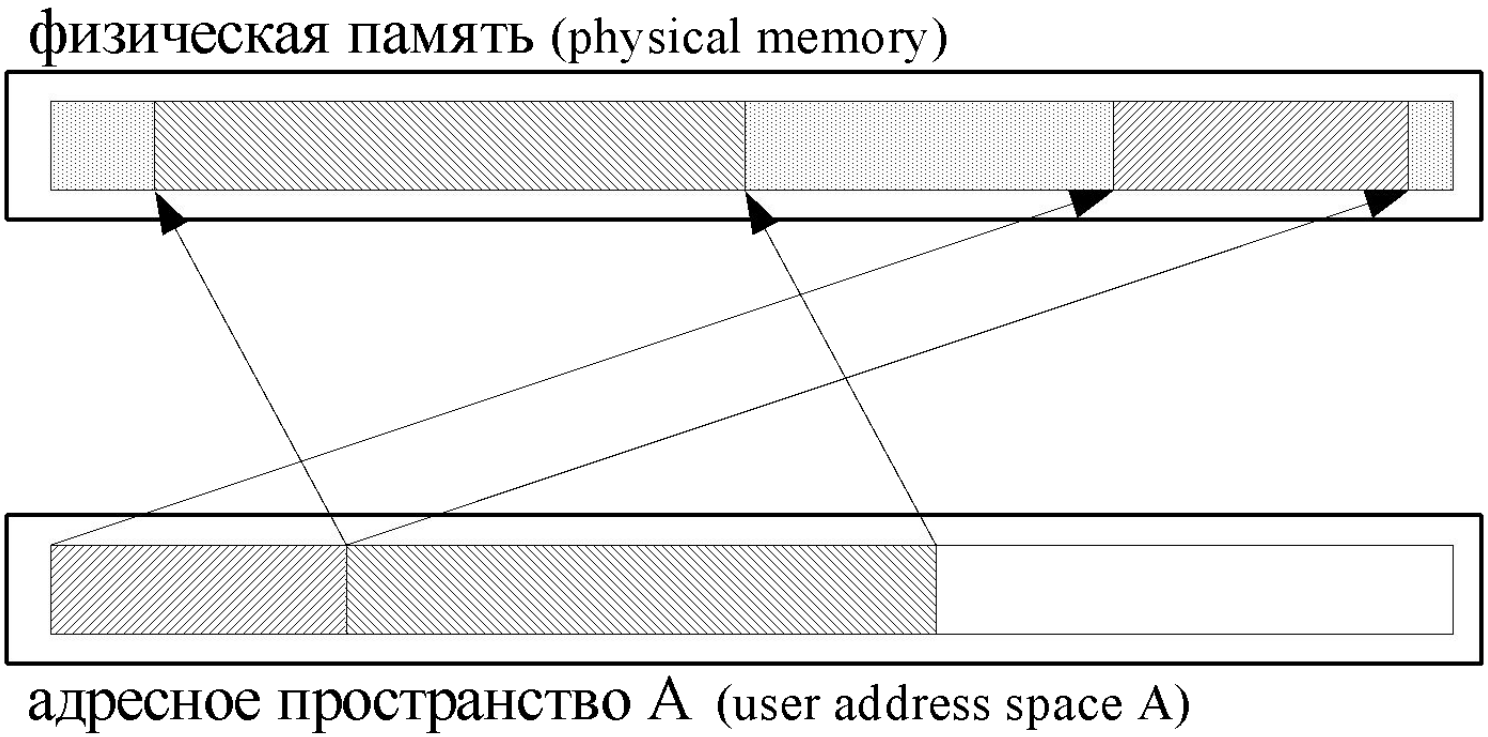
- **нити** (threads) – процессы, имеющие единое адресное пространство и выполняющиеся параллельно
- **ВМП** (IPC) – фундаментальный способ взаимодействия между нитями
- **адресное пространство** (address space) – защищенный набор преобразований виртуальных адресов в физические
- **планирование машинного времени** (scheduling) – обеспечивает контроль над порядком и длительностью выполнения нитей

Системные вызовы:

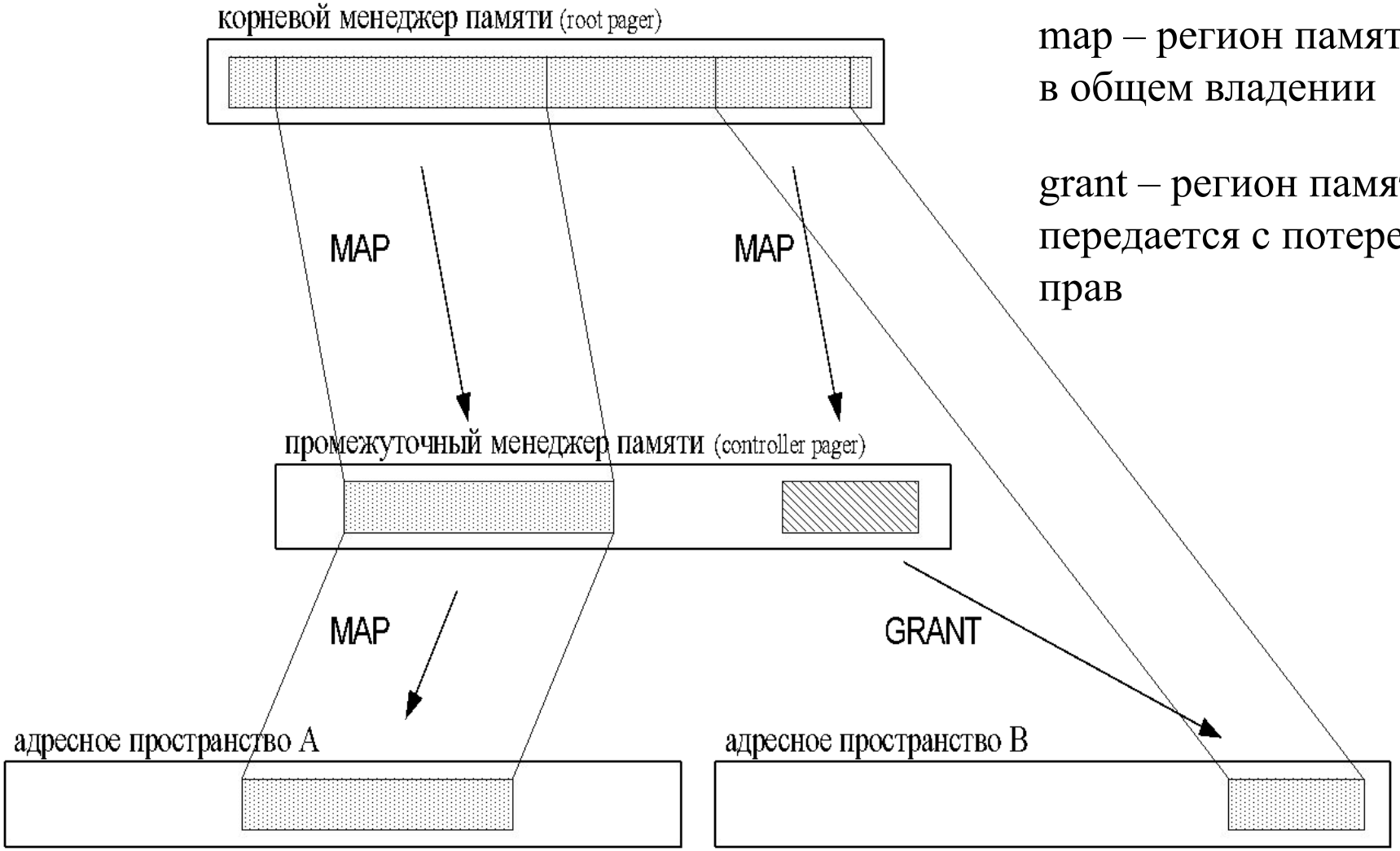
7 системных вызовов для изменения состояния системы

Адресное пространство (АП)

АП – это набор преобразований (translations) из виртуальных адресов в физические; содержит все регионы памяти доступные для нити напрямую.



Адресное пространство (2)



map – регион памяти в общем владении

grant – регион памяти передается с потерей прав

Адресное пространство (3)

Задача – набор нитей, использующих адресное пространство совместно (shared)

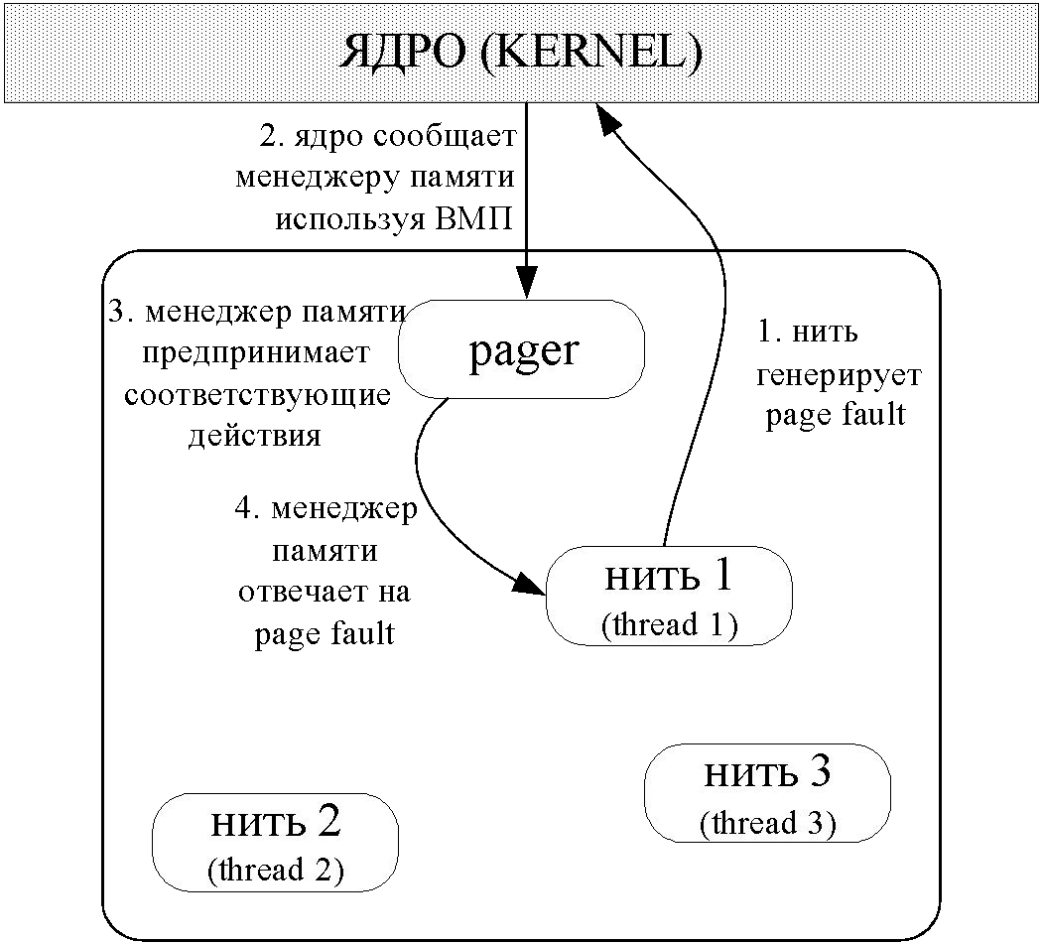
Нить определяется:

- используемым адресным пространством
- уникальным идентификатором (unique ID)
- набор регистров (register set)
- менеджер памяти (pager – page fault handler)

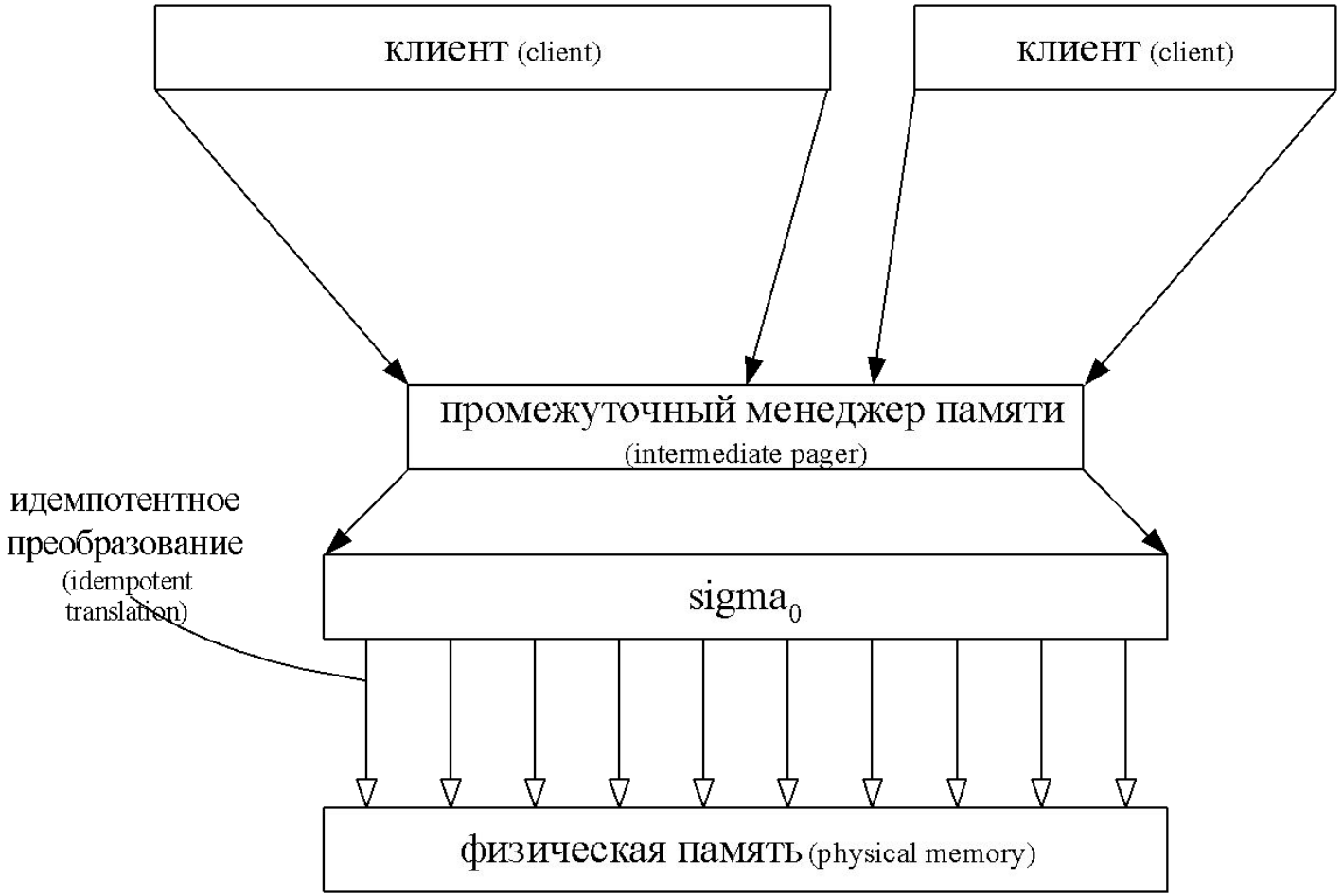
Менеджер памяти – нить, которая обрабатывает ошибки из-за отсутствия страницы памяти (page faults)

σ_0 – корневой менеджер памяти, владеющий всей памятью после запуска L4Ka

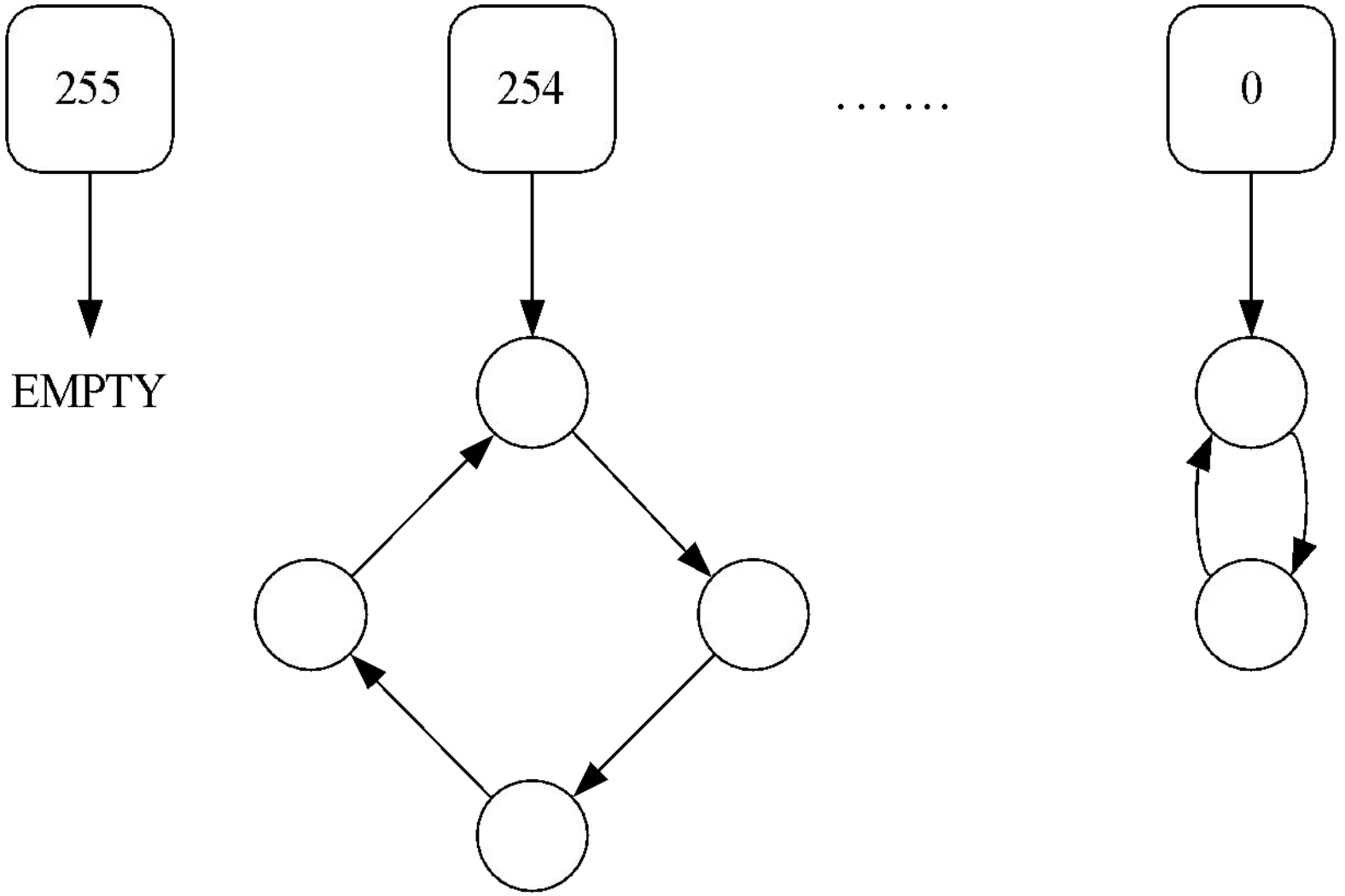
Менеджер памяти



Корневой менеджер памяти σ_0



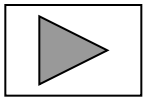
Планирование машинного времени



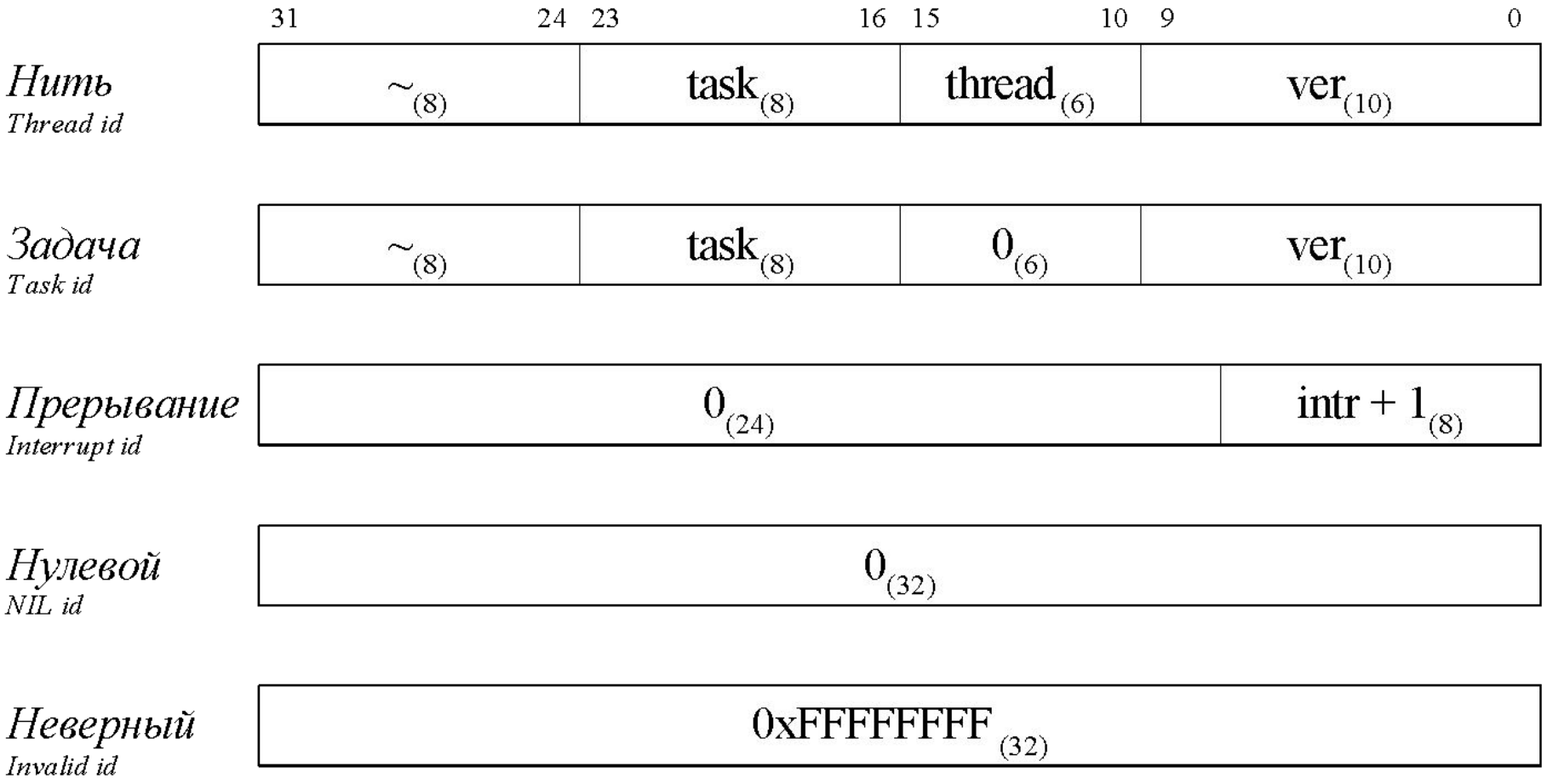
Типы данных ОС L4Ka

Основные типы данных:

- уникальный идентификатор (UID – unique ID)
- flex-pages
- тайм-аут (timeout)
- результат ВМП (IPC result status)
- параметр планирования (schedule parameter word)



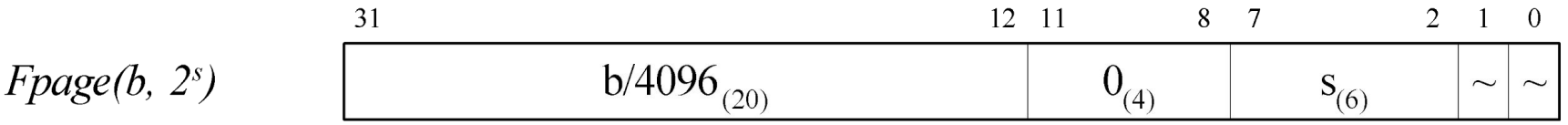
Уникальный идентификатор

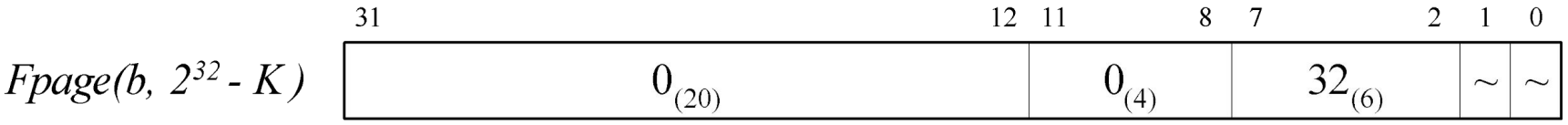


Flex-pages

Flex-page – это непрерывные регионы адресного пространства

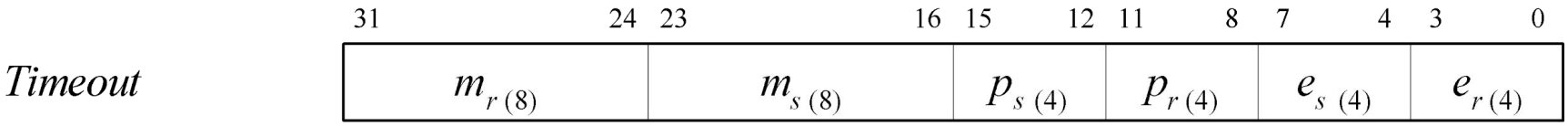
- необходимы для операций map и grant
- передаются как часть ВМП сообщений (IPC messages)
- имеют размер 2^s (для x86 архитектуры $s \geq 12$)
- базовый адрес b выровнен по значению 2^s





Тайм-аут (timeout)

Тайм-аут используется для контроля ВМП.
В 32-х битном поле определены четыре тайм-аута.



$$\text{rcv timeout} = \begin{cases} \infty & \text{if} \quad e_r = 0; \\ 4^{15-e_r} m_r \ \mu s & \text{if} \quad e_r > 0; \\ 0 & \text{if} \quad m_r = 0, \quad e_r \neq 0; \end{cases}$$

Тайм-аут (timeout) (2)

$$\text{snd timeout} = \begin{cases} \infty & \text{if} \quad e_s = 0; \\ 4^{15-e_s} m_s \ \mu s & \text{if} \quad e_s > 0; \\ 0 & \text{if} \quad m_s = 0, \quad e_s \neq 0. \end{cases}$$

$$\text{rcv pagefault timeout} = \begin{cases} \infty & \text{if} \quad p_r = 0; \\ 4^{16-p_r} \ \mu s & \text{if} \quad 0 < p_r < 15; \\ 0 & \text{if} \quad p_r = 15. \end{cases}$$

$$\text{snd pagefault timeout} = \begin{cases} \infty & \text{if} \quad p_s = 0; \\ 4^{15-p_s} \ \mu s & \text{if} \quad 0 < p_s < 15; \\ 0 & \text{if} \quad p_s = 15. \end{cases}$$

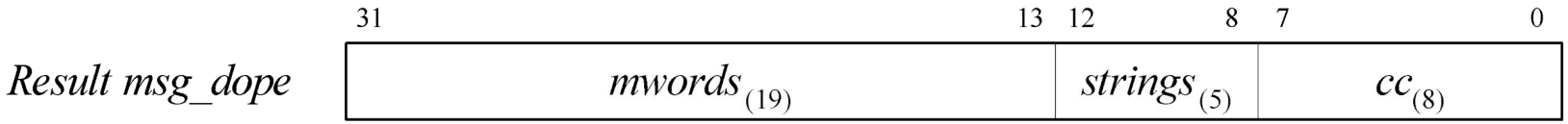
Результат ВМП (IPC result status)

Результат одной коммуникации между нитями возвращается в 32х-битном поле.

Поле «mwords» говорит о числе принятых 32х-битных слов.

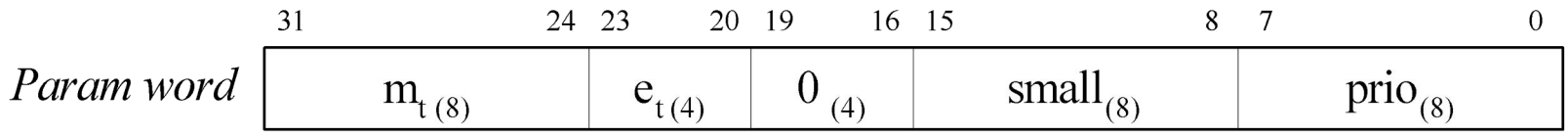
Поле «strings» говорит о числе принятых строк.

Младшие восемь бит представляют «код условия» (condition code), в котором закодирован код ошибки и тип сообщения.



Параметр планирования (schedule parameter word)

В L4Ka имеется только один параметр для планирования. Он содержит поля в которых закодирован приоритет и длина кванта времени (в соответствии со спецификацией тайм-аута).



Системный вызов l4_myself

Системный вызов возвращает уникальный идентификатор текущей нити.

Системный вызов l4_unmap

Регион памяти, описанный входным параметром fpage, будет освобожден (удален из адресного пространства нити).

Системный вызов l4_thread_ex_regs

С помощью этого системного вызова можно получить или изменить регистры EIP и ESP набора регистров нити.

Таким образом, можно создать новую нить, указав в качестве входных параметров значения для EIP, ESP, уникальный идентификатор менеджера памяти.

Системный вызов l4_thread_switch

Вызывающая нить добровольно освобождает ресурсы процессора. Планировщик выбирает следующую нить.

Системный вызов l4_thread_schedule

Изменение значений приоритета и длины кванта времени нити производится посредством передаче в качестве входного параметра структуры schedule param word.

Системный вызов также возвращает время процессора в микросекундах, израсходованное нитью.

Системный вызов l4_task_new

Этот системный вызов удаляет и/или создает задачу.

Удаление задачи означает удаление адресного пространства задачи и всех ее нитей.

Задачи могут быть созданы *активными* или *пассивными*.

При создании активной задачи создается новое адресное пространство вместе с полным набором нитей (64).

Пассивная задача пуста – не поглощает ресурсов процессора, не имеет ни адресного пространства, ни нитей, коммуникация с ней невозможна.

Пассивная задача по сути не существует и представляет собой лишь возможность, право создать активную задачу с данным уникальным идентификатор.

Системный вызов l4_ipc_call

Основной системный вызов для ВМП и синхронизации.

Вызов предоставляет несколько прототипов для коммуникации:

- l4_ipc_send – послать сообщение другой нити
- l4_ipc_receive – получить сообщение от указанной нити
- l4_ipc_reply_and_wait – ответить на сообщение указанной нити и ждать сообщения от произвольной нити
- l4_ipc_wait – ждать сообщения от произвольной нити

Вышеприведенные примитивы позволяют реализовать различные схемы взаимодействия (например, клиент-сервер).