

Дисциплина:

Операционные системы и среды Принципы построения операционных систем

Преподаватель:

Михайлова Екатерина Евгеньевна

ОПЕРАЦИОННЫЕ СИСТЕМЫ

т. 2. Организация
операционных систем

з. 10. Принципы построения
интерфейсов
операционных систем

Учебные вопросы

1. Интерфейс прикладного программирования
2. Платформенно-независимый интерфейс POSIX
3. Операционные системы реального времени

Назначение интерфейсов

Интерфейс операционной системы – специальный интерфейс системного и прикладного программирования, предназначенные для выполнения следующих задач:

1. Управление процессами
2. Управление памятью
3. Управление вводом/выводом

1. Интерфейс прикладного программирования

API - Application Programming Interface

API – набор функций, предоставляемых системой программирования разработчику прикладной программы и ориентированный на организацию взаимодействия результирующей прикладной программы с целевой вычислительной системой.

Варианты реализации API

1. Реализация на уровне ОС
2. Реализация на уровне системы программирования
3. Реализация на уровне внешней библиотеки процедур и функций

Возможности API оцениваются по следующим параметрам:

- **эффективность выполнения функций API** – включает в себя скорость выполнения функций и объем вычислительных ресурсов, потребных для их выполнения;
- **широта предоставляемых возможностей;**
- **зависимость прикладной программы от архитектуры вычислительной системы.**

Реализация функций API на уровне ОС

Функции входят в состав ОС (или ядра) или поставляются в составе динамически загружаемых библиотек, разработанных для данной ОС.

Недостаток:

- практически полное отсутствие переносимости не только кода результирующей программы, но и кода исходной программы.

Система программирования не сможет выполнить перестроение исходного кода для новой архитектуры вычислительной системы

Реализация функций API на уровне системы программирования

Функции предоставляются пользователю в виде библиотеки функций соответствующего языка программирования (библиотека времени исполнения – RTL (run time library)).

Система программирования обеспечивает подключение к результирующей программе объектного кода, ответственного за выполнение этих функций.

Функции динамического выделения памяти в C - malloc, realloc и free (функции new и delete в C++)

В Pascal – функции new и dispose.

Системы программирования для каждой из этих функций должна подключить к результирующей программе объектный код библиотеки.

Для различных вариантов ОС этот код будет различен даже при использовании одного и того же исходного языка.

Реализация функций API с помощью внешних библиотек

Функции API с помощью внешних библиотек предоставляются пользователю в виде библиотеки процедур и функций, созданной сторонним разработчиком.

Библиотека графического интерфейса
поддерживающая стандарт графической
среды X Window

XLib □ **X Window**

MFC – Microsoft Foundation Classes

VCL – Visual Component Library

ориентированы на архитектуру ОС типа Windows

CLX – Component Library for Cross-Platform

фирмы Borland ориентирована на архитектуру ОС
типа Linux и ОС типа Windows.

2. Платформенно- независимый интерфейс POSIX

POSIX - **(Portable Operating System Interface [based on] uniX)**

Стандарт IEEE (Institute of Electrical and Electronical Engineers), описывающий системные интерфейсы для открытых операционных систем, в том числе оболочки, утилиты и инструментарии. Согласно POSIX, стандартизированными являются задачи обеспечения безопасности, задачи реального времени, процессы администрирования, сетевые функции и обработка транзакций.

IEEE Standard 1003.1-1990 (POSIX.1)

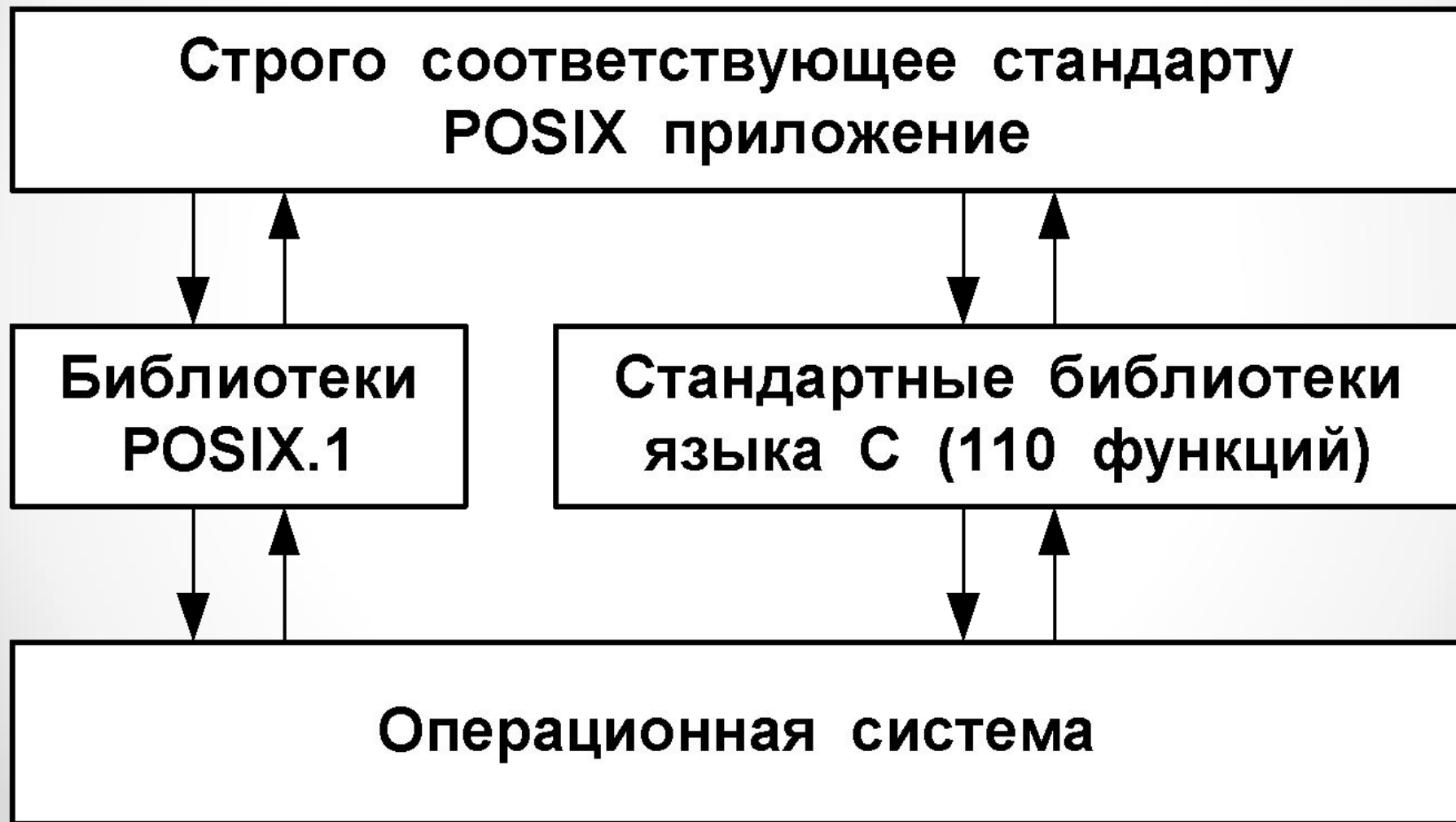
Стандарт **POSIX** подробно описывает **VMS - Virtual Memory System** (систему виртуальной памяти), многозадачность (**MPE - MultiProcess Executing**) и технологию переноса операционных систем (СТОС).

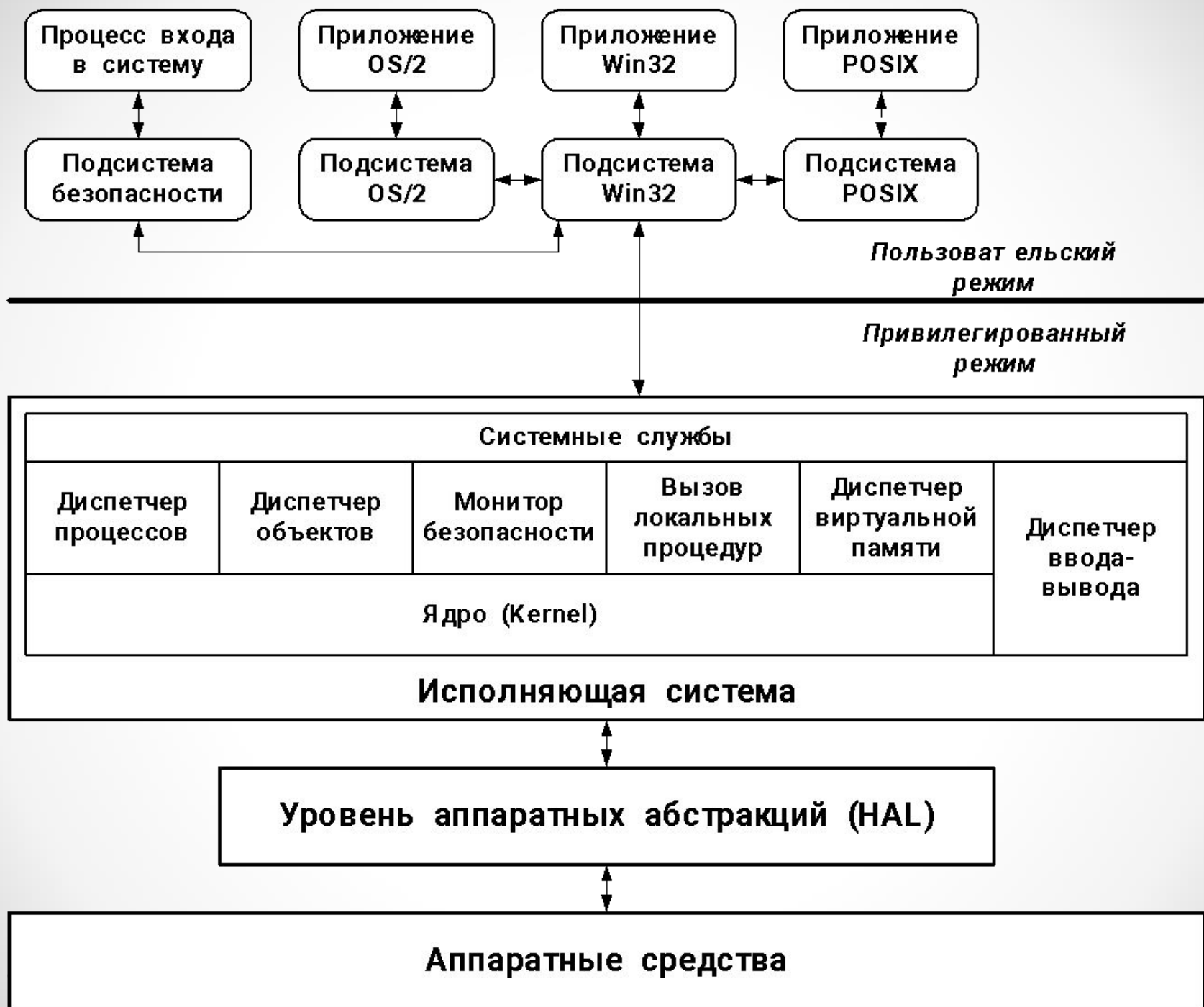
POSIX - множество стандартов, именуемых POSIX.1 – POSIX.12.

Стандарт	Краткое описание
POSIX.0	Введение в стандарт открытых систем
POSIX.1	Системный API (язык C)
POSIX.2	Оболочки и утилиты (одобренные IEEE)
POSIX.3	Тестирование и верификация
POSIX.4	Задачи реального времени и потоки
POSIX.5	Использование языка ADA применительно к стандарту POSIX.1
POSIX.6	Системная безопасность
POSIX.7	Администрирование системы

Стандарт	Краткое описание
POSIX.8	Сети «Прозрачный» доступ к файлам Абстрактные сетевые интерфейсы, не зависящие от физических протоколов RPC (Remote Procedure Call, вызовы удаленных процедур) Связь системы с протоколо-зависимыми приложениями
POSIX.9	Использование языка Fortran применительно к стандарту POSIX.1
POSIX.10	Super-computing Application Environment Profile (AEP) Профиль прикладной среды организации вычислений на супер-ЭВМ
POSIX.11	Обработка транзакций AEP
POSIX.12	Графический интерфейс пользователя (GUI)

Приложения, строго соответствующие стандарту POSIX





Модуль поддержки **POSIX API**, работающий на уровне привилегий пользовательских процессов обеспечивает конвертацию и передачу вызовов из пользовательской программы к ядру системы и обратно, работая с ядром через **WinAPI**.

Другие приложения, написанные с использованием **WinAPI**, могут передавать информацию POSIX-приложениям через стандартные механизмы потоков ввода-вывода

3. Операционные системы реального времени

Характеристики операционных систем, определяющие возможности в решении различных задач

- аппаратная платформа и скорость работы ОС;
- поддерживаемое периферийное оборудование;
- возможности для организации сетей;
- обеспечение совместимости с другими операционными системами;
- переносимость ОС на другие аппаратные платформы;
- наличие в составе ОС инструментальных средств для разработки прикладных систем и др.

Операционная система реального времени
предназначена для разработки прикладного программного обеспечения пользователя (трансляцию исходного текста программ, компоновку программ и получение загрузочного модуля, поддержку режима отладки программ пользователя в целевой (инструментальной) ЭВМ), а также для обеспечения функционирования программ пользователя в режиме реального времени и в режиме отладки на целевой ЭВМ.

Классы ОС:

1. ОС «мягкого» реального времени, в которых проектирование прикладного программного обеспечения и его исполнение производится на одной и той же ЭВМ (ОС OS9/9000 и QNX).

Используются, когда конечное применение не требует:

- высокой производительности ОС
- максимальной скорости реакции на внешние события
- строгой детерминированности ее поведения.

2. ОС, в которых применяется кросс-технология, т.е. проектирование прикладного программного обеспечения ведется на одной машине (host), называемой «инструментальной», а разработанное программное обеспечение выполняется на другой, «целевой» машине (target)



Применяются в системах “жесткого” реального времени, в которых недостаточная скорость реакции или непредсказуемость поведения влечет за собой “аварию” на объекте управления, и/или невыполнение задачи, и/или создает опасность для жизни людей

ОС	ЭВМ	Время на переключение (микросекунды)
VxWorks	MC680040-25	6
VxWorks	R3000-25	24
OS9/9000	PowerPC601	26
QNX 4.2	ALR Pentium-60	28
HP-RT 1.1	HP-747x-100	34
QNX 4.2	IBM80486DX2-60	44
DEC OSF/1 V1.3	DEC 21064-150	93
SunOS 4.1.3	SuperSPARCv8-50	95
HP-UX 9.x	Snake-60	106
SunOS 4.1.3	Viking-40	128
Ultrix 4.3	Digital MIPS R3000-40	132
Linux 0.99.13p	Gateway 8048DX2-66	171
SunOS 4.1.1	SunSPARC-33	198
386BSD 0.1	IBM486DX2-33	210
AIX 3.2	RIOS-50	212
SunOS 4.1.3	SPARCv7-50	230
Unicons	Cray Y/MP	373
QNX 4.2	IBM386SX-16	525
Solaris 2.3	MicroSPARCv8-50	595

Недостатки:

1. Каждая такая система снабжена своим собственным уникальным интерфейсом.
2. Отсутствие совместимости разрабатываемых ОС ограничивает расширение возможностей любой системы реального времени по решению новых задач, использующей в своей работе определенную ОС.

Требования, предъявляемые к ОС реального времени:

1. Предсказуемость - поведение ОС должно быть известно и достаточно точно прогнозируемо.

Заданные характеристики:

- латентная задержка;
- максимальное время выполнения каждого системного вызова;
- максимальное время маскирования прерываний драйверами И ОС.

2. Мультипрограммность и многозадачность

ОС должна быть многопоточной по принципу абсолютного приоритета (прерываемой).

3. Приоритеты задач (потоков)

Необходимо определять какой задаче ресурс требуется более всего. ОСРВ отдает ресурс потоку или драйверу с ближайшим крайним сроком (это называется управлением временным ограничением).

4. Наследование приоритетов

Инверсии приоритетов - комбинация приоритетов потоков и разделения ресурсов между ними

Время, необходимое для завершения потока высшего приоритета, зависит от нижних приоритетных уровней.

5. Синхронизация процессов и задач

Необходимы механизмы, гарантированно предоставляющие возможность параллельно выполняющимся задачам и процессам оперативно обмениваться сообщениями и синхросигналами.

Операционная система Linux

Linux является UNIX- подобной системой

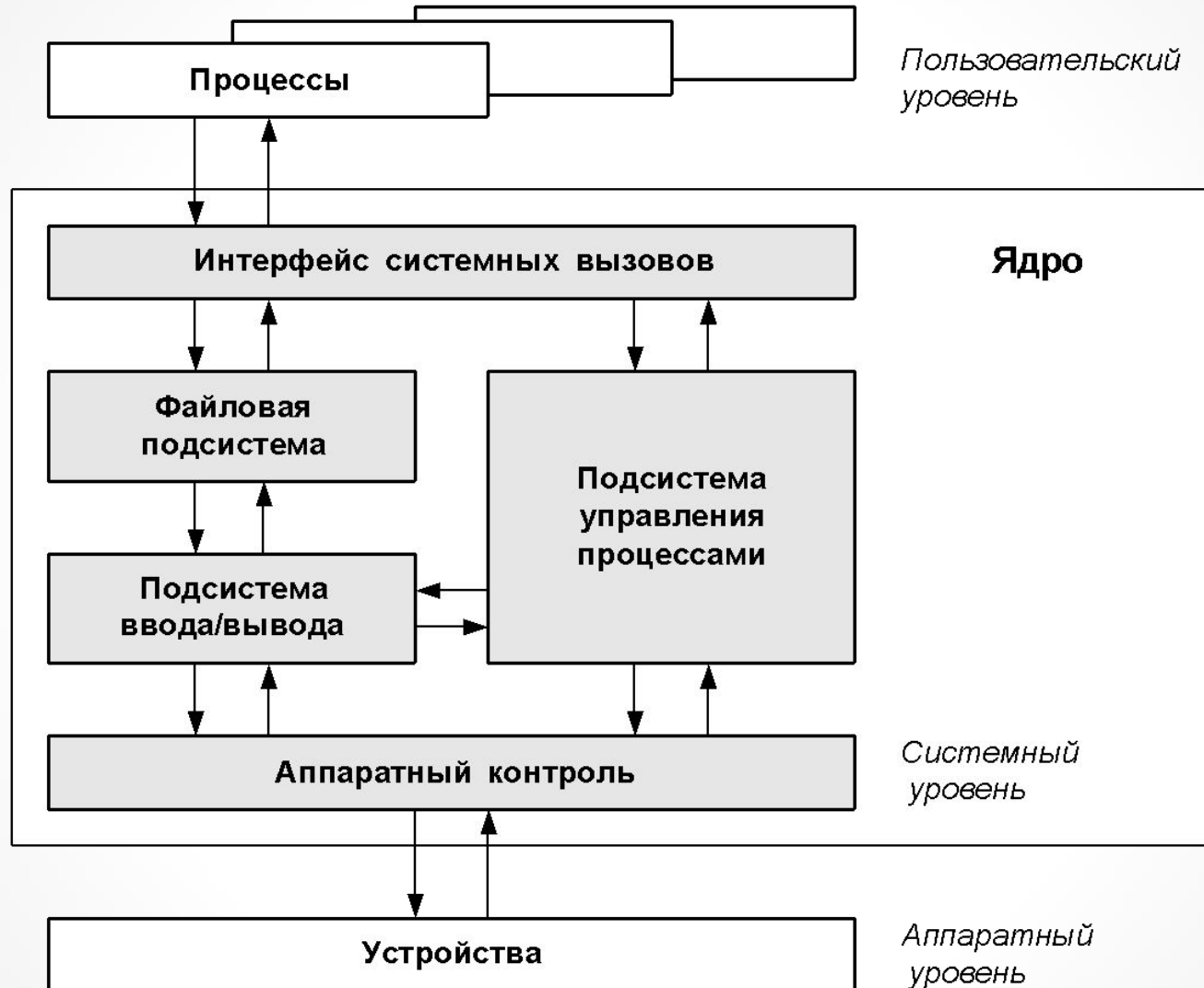
Модель ОС Linux



Ядро системы

Ядро обеспечивает базовую функциональность ОС:

- создает процессы и управляет ими
- распределяет память
- обеспечивает доступ к файлам и периферийным устройствам.



- **Файловая подсистема** обеспечивает унифицированный интерфейс доступа к данным, расположенным на дисковых накопителях, и к периферийным устройствам

Задачи **подсистемы управления процессами**:

- создание и удаление процессов;
- распределение системных ресурсов (памяти, вычислительных ресурсов) между процессами;
- синхронизация процессов;
- межпроцессное взаимодействие.

- **Планировщик процессов(scheduler)** – специальный модуль ядра, который разрешает конфликты между процессами и конкуренцию за системные ресурсы
- **Модуль управления памятью** обеспечивает размещение оперативной памяти для прикладных задач
- **Модуль межпроцессного взаимодействия** отвечает за уведомление процессов о событиях с помощью сигналов и обеспечивает возможность обмена данными между различными процессами.

- **Подсистема ввода/вывода** выполняет запросы файловой подсистемы и подсистемы управления процессами для доступа к периферийным устройствам (дискам, терминалам и т.п.).

Операционная система QNX

Операционная система QNX - это ОС стандарта POSIX, позволяющая обеспечить на ЭВМ:

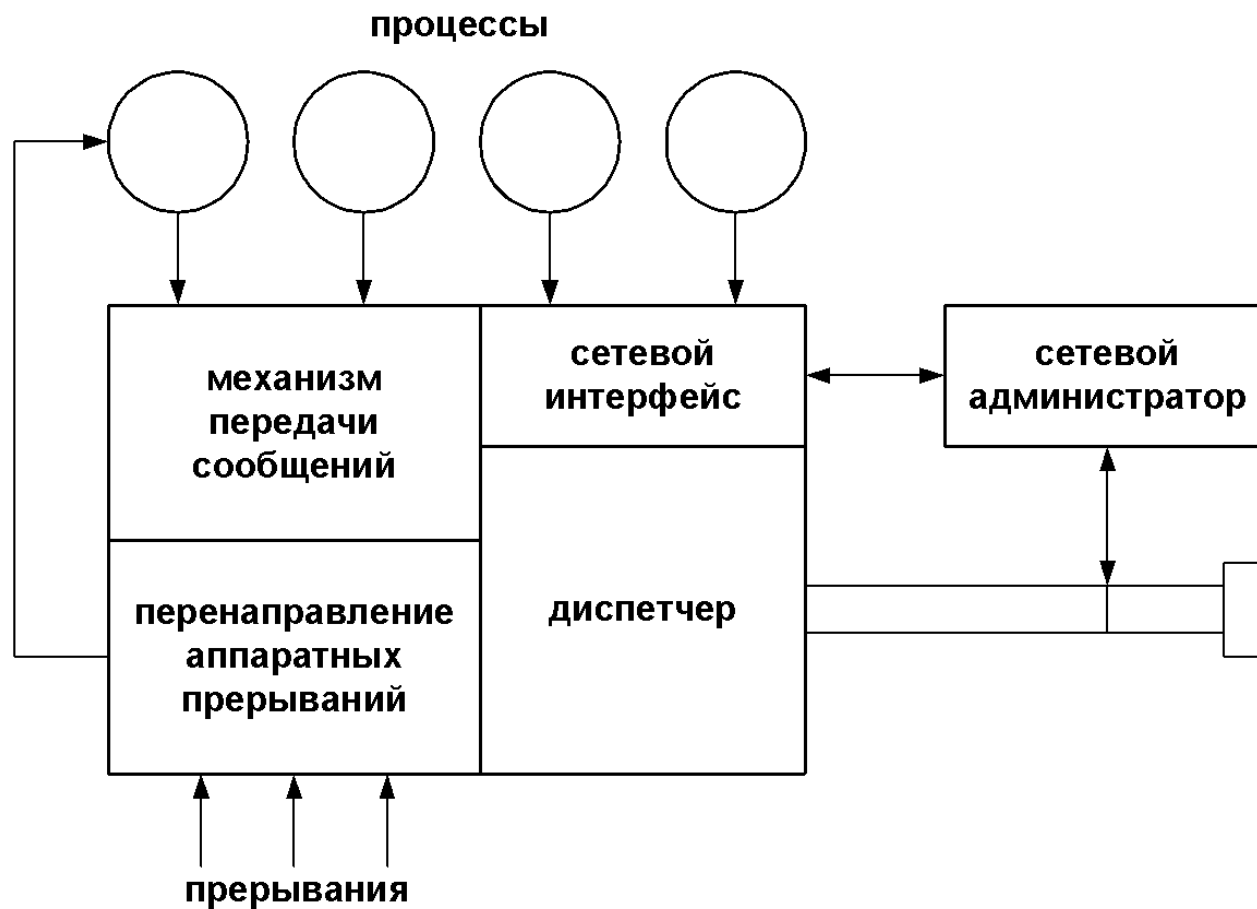
- распределенную обработку данных в реальном масштабе времени;
- передачу сообщений в качестве основного средства взаимодействия между процессами;

- сетевое взаимодействие “каждый с каждым” между любыми узлами сети;
- расширение сети простым добавлением узлов, не используя сложных файл-серверов или дополнительного сетевого оборудования.

QNX содержит:

1. **Администратор процессов** (Process Manager), отвечающий за распределение памяти, запуск и окончание задач в системе;
2. **Администратор периферийных устройств** (Device Manager), управляющий всем периферийным оборудованием:
3. **Администратор файловой системы** (File system Manager).
4. **Администратор сети** (Network Manager), обеспечивающий коммуникации в сети.

Структура ядра операционной системы QNX



Файлы в ОС QNX организованы по принципу набора участков, ссылки на которые находятся в дескрипторах файлов и в отдельных участках дисковой памяти.

ОС QNX создает следующую файловую структуру:

- для отслеживания свободного дискового пространства используется карт битов;
- для организации данных - массив расширений для каждого файла.

Задание

на самостоятельную работу

- **КОНСПЕКТ**
- **Л1 – с. 296-307**