**Архитектура UNIX**

**Первоисточник:** [**http://heap.altlinux.org/alt-docs/modules/unix\_base\_admin.dralex/ch01s02.html**](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html)

**Особенности архитектуры UNIX**

[Презентация 2-01: основные концепции UNIX](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#small-pres-2-01)

Знакомство с архитектурой UNIX начнем с рассмотрения таких неотъемлимых для неё характеристических понятий, как *стандартизация* и *многозадачность*:

Стандартизация

Несмотря на многообразие версий UNIX, основой всего семейства являются принципиально одинаковая архитектура и ряд стандартных интерфейсов (в UNIX стандартизовано почти всё – от расположения системных папок и файлов, до интерфейса системных вызовов и списка драйверов базовых устройств). Опытный администратор без особого труда сможет обслуживать другую версию, тогда как для пользователей переход на другую систему и вовсе может оказаться незаметным. Для системных же программистов такого рода стандарты позволяют полностью сосредоточиться на программировании, не тратя время на изучение архитектуры и особенностей конкретной реализации системы.

Многозадачность

В системе UNIX может одновременно выполняться множество процессов (задач), причем их число логически не ограничивается, и множество частей одной программы может одновременно находиться в системе. Благодаря специальному механизму управления памятью, каждый процесс развивается в своем защищенном адресном пространстве, что гарантирует безопасность и независимость от других процессов. Различные системные операции позволяют процессам порождать новые процессы, завершают процессы, синхронизируют выполнение этапов процесса и управляют реакцией на наступление различных событий.

**Два кита UNIX: файлы и процессы**

Существует два основных объекта операционной системы UNIX, с которыми приходиться работать пользователю – *файлы* и *процессы*. Эти объекты сильно связаны друг с другом, и в целом организация работы с ними как раз и определяет архитектуру операционной системы.

Все данные пользователя храняться в *файлах*; доступ к периферийным устройствам осуществляется посредством чтения и записи специальных файлов; во время выполнения программы, операционная система считывает исполняемый код из файла в память и передает ему управление.

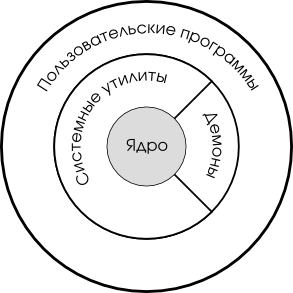
С другой стороны, вся функциональность операционная определяется выполнением соответствующих *процессов*. В частности, обращение к файлам на диске невозможно, если файловая подсистема операционной системы (совокупность процессов, осуществляющих доступ к файлам) не имеет необходимого для этого кода в памяти.

**Беглый взгляд на архитектуру UNIX**

[Презентация 2-02: беглый взгляд на архитектуру UNIX](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#small-pres-2-02)

Самый общий взляд на архитектуру UNIX позволяет увидеть *двухуровневую модель системы*, состоящую из *пользовательской* и *системной части (ядра)* (см. [Рисунок 1.20, «Архитектура операционной системы UNIX»](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#img-unix-arch)). Ядро непосредственно взаимодействует с аппаратной частью компьютера, изолируя прикладные программы (процессы в пользовательской части операционной системы) от особенностей ее архитектуры. Ядро имеет набор услуг, предоставляемых прикладным программам посредством системных вызовов. Таким образом, в системе можно выделить два уровня привилегий: *уровень системы* (привиегии специального пользователя root) и *уровень пользователя* (привилегии всех остальных пользователей). Подробнее об управлении доступом рассказывается в следующих главах ([Глава 3, *Безопасность операционной системы UNIX*](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch03.html)).

**Рисунок 1.20. Архитектура операционной системы UNIX**



Важной частью системных программ являются *демоны*. Демон – это процесс, выполняющий опеределенную функцию в системе, который запускается при старте системы и не связан ни с одним пользовательским терминалом. Демоны предоставляют пользователям определенные сервисы, примерами которых могут служить системный журнал, веб-сервер и т.п.. Аналогом демонов в операционной системе Windows NT и более поздних версиях являются *системные службы*.

**Ядро UNIX**

[Презентация 2-03: ядро UNIX](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#small-pres-2-03)

Операционная система UNIX обладает классическим монолитным ядром (см. [«Архитектура операционной системы»](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01.html#intro-concepts-arch)), в котором можно выделить следующие основные части:

Файловая подсистема

Доступ к структурам ядра осуществляется через файловый интерфейс.

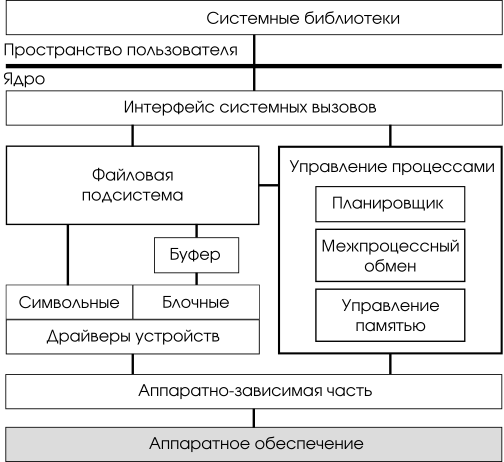
Управление процессами

Сюда входит управление параллельным выполнением процессов (планирование и диспетчеризация), виртуальной памятью процесса, и взаимодействием между процессами (сигналы, очереди сообщений и т.п.).

Драйверы устройств

Драйверы устройств делятся на символьные и блочные по типу внешнего устройства. Для каждого из устройств определен набор возможных операций (открытие, чтение и т.д.). Блочные устройства кэшируются с помощью специального внутреннего механизма управления буферами.

**Рисунок 1.21. Ядро операционной системы UNIX**



Благодаря тому, что в UNIX аппаратно-независимая часть явно отделена, это семейство операционных систем может быть с минимальными затратами перенесено на новые аппаратные платформы.

Видно, что ядро операционной системы UNIX является классическим для многозадачной многопользовательской операционной системы, поэтому оно широко используется в обучении системному программированию и теории операционных систем.

**Файловая система UNIX**

Термин *файловая система* по историческим причинам обозначает одновременно и иерархию каталогов и файлов, и часть ядра, управляющую доступом к каталогам и файлам.

**Особенности файловой системы**

[Презентация 2-04: файловая система UNIX](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#small-pres-2-04)

*Первое* значение термина упирается в рассмотрение структур, в которые могут быть организованы файлы на носителях данных. Существует несколько видов таких структур: линейные, древовидные, объектные и другие, но в настоящее время широко распространены только древовидные структуры.

Каджый файл в древовидной структуре расположен в определенном хранилище файлов – *каталоге*, каждый каталог, в свою очередь, также расположен в некотором каталоге. Таким образом, по принципу вложения элементов файловой системы (файлов и каталогов) друг в друга строится дерево, вершинами которого являются непустые каталоги, а листьями – файлы или пустые каталоги. Корень такого дерева имеет название *корневой каталог* и обозначается каким-либо специальным символом или группой символов (например, «C:» в операционной системе Windows). Каждому файлу соответствует некоторое *имя*, отпределяющее его расположение в дереве файловой системы. Полное имя файла состоит из имен всех вершин дерева файловой системы, через которые можно пройти от корня до данного файла (каталога), записывая их слева-направо и разделяя специальными символами-разделителями.

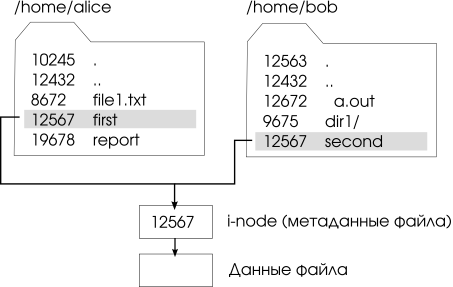
В настоящее время существует огромное количество файловых систем, каждая из которых используется для определенной цели: для быстрого доступа к данным, для обеспечения целостности данных при сбоях системы, для простоты реализации, для компактного хранения данных, и т.д. Однако среди всего множества файловых систем можно выделить такие, которые обладают рядом схожих признаков, а именно:

* [Презентация 2-05: индексный узел](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html" \l "small-pres-2-05" \o "Рисунок 1.32. Презентация 2-05: индексный узел)

Файлы и каталоги идентифицируются не по именам, а по *индексным узлам* (i-node) – индексам в общем массиве файлов для данной файловой системе. В этом массиве хранится информация об используемых блоках данных на носителе, а также – длина файла, владелец файла, права доступа и другая служебная информация под общим названием «метаданные о файле». Логические же связки типа «имя–i-node» – есть ни что иное как содержимое каталогов.

Таким образом, каждый файл характеризуется одним i-node, но может быть связан с несколькими именами – в UNIX это называют *жёсткими ссылками* (см. [Рисунок 1.22, «Пример жесткой ссылки»](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#img-hard-link)). При этом, удаление файла происходит тогда, когда удаляется последняя жёсткая ссылка на этот файл.

**Рисунок 1.22. Пример жесткой ссылки**



Важной особенностью таких файловых систем является то, что имена файлов зависят от регистра, другими словами файлы test.txt и TEST.txt отличаются (т.е. являются разными строками в файле директории).

* В определенных (фиксированных для данной файловой системы) блоках физического носителя данных находится т.н. *суперблок*. Суперблок – это наиболее ответственная область файловой системы, содержащая информацию для работы файловой системы в целом, а также – для ёе идентификации. В суперблоке находится «магическое число» – идентификатор файловой системы, отличающий её от других файловых систем, список свободных блоков, список свободных i-node'ов и некоторая другая служебная информация.
* Помимо *каталогов* и *обычных файлов* для хранения информации, ФС может содержать следующие виды файлов:

Специальный *файл устройства*

Обеспечивает доступ к физическому устройству. При создании такого устройства указывается тип устройства (блочное или символьное), *старший номер* – индекс драйвера в таблице драйверов операционной системы и *младший номер* – параметр, передаваемый драйверу, поддерживающему несколько устройств, для уточнения о каком «подустройстве» идет речь (например, о каком из нескольких IDE-устройств или COM-портов).

*Именованный канал*

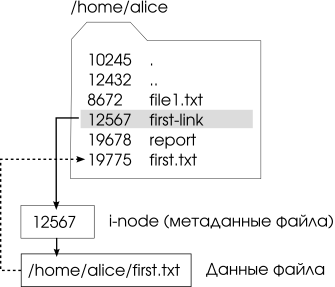
Используется для передачи данных между процессами, работает по принципу двунаправленной очереди (FIFO). Является одним из способов обмена между изолированными процессами (подробнее см. [«Межпроцессное взаимодействие»](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#intro-unix-proc-interproc)).

*Символическая ссылка*

Особый тип файла, содержимое которого – не данные, а имя какого-либо другого файла (см. [Рисунок 1.23, «Пример символической ссылки»](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#img-sym-link). Для пользователя такой файл неотличим от того, на который он ссылается.

Символическая ссылка имеет ряд преимуществ по сравнению с жёсткой ссылкой: она может использоваться для связи файлов в разных файловых системах (ведь номера индексных узлов уникальны только в рамках одной файловой системы), а также более прозрачно удаление файлов – ссылка может удаляться совершенно независимо от отсновного файла.

**Рисунок 1.23. Пример символической ссылки**



*Сокет*

Предназначен для взаимодействия между процессами через специальное API, схожее с TCP/IP-сокетеми (подробнее см. [«Межпроцессное взаимодействие»](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#intro-unix-proc-interproc)).

Такие файловые системы наследуют особенности оригинального UNIX. К ним можно отнести, например: s5 (используемая в версиях UNIX System V), ufs (BSD UNIX), ext2, ext3, reiserfs (Linux), qnxfs (QNX). Все эти файловые системы различаются форматами внутренних структур, но совместимы с точки зрения основных концепций.

**Дерево каталогов**

[Презентация 2-06: монтирование файловых систем](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#small-pres-2-06)

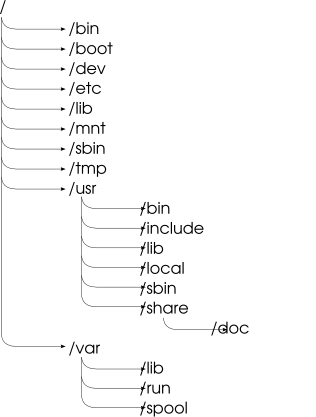
Рассмотрение *второго* значения термина ФС приводит нас к уже обозначенной ранее совокупности процедур, осуществляющих доступ к файлам на различных носителях. Особенностью операционных систем семейства UNIX является существование единого дерева файловой системы для любого количества носителей данных с одинаковыми или разными типами файловых систем на них. Это достигается путем *монтирования* – временной подстановкой вместо каталога одной файловой системы дерева другой файловой системы, вследствие чего система имеет не несколько деревьев никак не связанных друг с другом, а одно большое разветвленное дерево с единым корневым каталогом.

Файловая подсистема операционной системы UNIX имеет имеет уникальную систему обработки запросов к файлам – *переключатель файловых систем* или *виртуальная файловая система* (VFS). VFS предоставляет пользователю стандартный набор функций (интерфейс) для работы с файлами, вне зависимости от места их расположения и принадлежности к разным файловым системам.

В мире стандартов UNIX определено, что корневой каталог единого дерева файловой системы должен иметь имя /, как и символ-разделитель при формировании полного имени файла. Тогда полное имя файла может быть, например, /usr/share/doc/bzip2/README. Задача VFS – по полному имени файла найти его местоположение в дереве файловой системы, определить её тип в этом месте дерева и «переключить», т.е. передать файл на дальнейшую обработку драйверу конктретной файловой системы. Такой подход позволяет использовать практически неограниченое количество различных файловых систем на одном компьютере под управлением одной операционной системы, а пользователь даже не будет знать, что файлы физически находятся на разных носителях информации.

Использование общепринятых имен основных файлов и структуры каталогов существенно облегчает работу в операционной системе, её администрирование и переносимость. Некоторые из этих структур используются при запуске системы, некоторые – во время работы, но все они имеют большое значение для ОС вцелом, а нарушение этой структуры может привести к неработоспособности системы или ее отдельных компонентов.

**Рисунок 1.24. Стандартные каталоги в файловой системе UNIX**



[Презентация 2-07: стандарт на файловую систему](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#small-pres-2-07)

Приведем краткое описание основных каталогов системы, формально описываемых специальным стандартом на *иерархию файловой системы* (Filesystem Hierarchy Standart). Все каталоги можно разделить на две группы: для статической (редко меняющейся) информации – /bin, /usr и динамической (часто меняющейся) информации – /var, /tmp. Исходя из этого администраторы могут разместить каждый из этих каталогов на собственном носителе, обладающем соответствующими характеристиками.

Корневой каталог

Корневой каталог / является основой любой ФС UNIX. Все остальные каталоги и файлы располагаются в рамках струтуры (дерева), порождённой корневым каталогом, независимо от их физического местонахождения.

/bin

В этом каталоге находятся часто употребляемые команды и утилиты системы общего пользования. Сюда входят все базовые команды, доступные даже если была примонтирована только корневая файловая система. Примерами таких команд являются: **ls**, **cp**, **sh** и т.п..

/boot

Директория содержит всё необходимое для процесса загрузки операционной системы: программу-загрузчик, образ ядра операционной системы и т.п..

/dev

Каталог содержит специальные файлы устройств, являющиеся интерфейсом доступа к периферийным устройствам. Наличие такого каталога не означает, что специальные файлы устройств нельзя создавать в другом месте, просто достаточно удобно иметь один каталог для всех файлов такого типа.

/etc

В этом каталоге находятся системные конфигурационные файлы. В качестве примеров можно привести файлы /etc/fstab, содержащий список монтируемых файловых систем, и /etc/resolv.conf, который задаёт правила составления локальных DNS-запросов. Среди наиболее важных файлов – скрипты инифиализации и деинициализации системы. В системах, наследующих особенности UNIX System V, для них отведены каталоги с /etc/rc0.d по /etc/rc6.d и общий для всех файл описания – /etc/inittab.

/home (необязательно)

Директория содержит домашние директории пользователей. Её существование в корневом каталоге не обязательно и её содержимое зависит от особенностей конкретной UNIX-подобной операционной системы.

/lib

Каталог для статических и динамических библиотек, необходимых для запуска программ, находящихся в директориях /bin и /sbin.

/mnt

Стандартный каталог для временного монтирования файловых систем – например, гибких и флэш-дисков, компакт-дисков и т.п..

/root (необязательно)

Директория содержит домашюю директорию суперпользователя. Её существование в корневом каталоге не обязательно.

/sbin

В этом каталоге находятся команды и утилиты для системного администратора. Примерами таких команд являются: **route**, **halt**, **init** и т.п.. Для аналогичных целей применяются директории /usr/sbin и /usr/local/sbin.

/usr

Эта директория повторяет структуру корневой директории – содержит каталоги /usr/bin, /usr/lib, /usr/sbin, служащие для аналогичных целей.

Каталог /usr/include содержит заголовочные файлы языка C для всевозможные библиотек, расположенных в системе.

Каталог /usr/local является следующим уровнем повторения корневого каталога и служит для хранения программ, установленных администратором в дополнение к стандартной поставке операционной системы.

Каталог /usr/share хранит неизменяющиеся данные для установленных программ. Особый интерес представляет каталог /usr/share/doc, в который добавляется документация ко всем установленным программам.

/var, /tmp

Используются для хранения временных данных процессов – системных и пользовательских соответственно.

**Управление процессами**

[Презентация 2-08: контекст процесса](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#small-pres-2-08)

В операционной системе UNIX традиционно поддерживается классическая схема мультипрограммирования. Система поддерживает возможность параллельного (или псевдопараллельного в случае наличия только одного аппаратного процессора) выполнения нескольких пользовательских программ. Каждому такому выполнению соответствует процесс операционной системы. Каждый процесс выполняется в собственной виртуальной памяти, и, тем самым, процессы защищены один от другого, т.е. один процесс не в состоянии неконтроллируемым образом прочитать что-либо из памяти другого процесса или записать в нее.

**Контекст процесса**

Каждому процессу соответствует *контекст*, в котором он выполняется. Этот контекст включает содержимое пользовательского адресного пространства – пользовательский контекст (т.е. содержимое сегментов программного кода, данных, стека, разделяемых сегментов и сегментов файлов, отображаемых в виртуальную память), содержимое аппаратных регистров – регистровый контекст (регистр счетчика команд, регистр состояния процессора, регистр указателя стека и регистры общего назначения), а также структуры данных ядра (контекст системного уровня), связанные с этим процессом. Контекст процесса системного уровня в ОС UNIX состоит из «статической» и «динамических» частей. Для каждого процесса имеется одна статическая часть контекста системного уровня и переменное число динамических частей.

Статическая часть контекста процесса системного уровня включает следующее:

*Идентификатор процесса (PID)*

Уникальный номер, идентифицирующий процесс. По сути, это номер строки в таблице процессов – специальной внутренней структуре ядра операционной системы, хранящей информацию о процессах.

В любой момент времени номера запущенных в ситеме процессов отличаются, однако после завершения процесса, его номер может быть в дальнейшем использован для идентификации вновь запущенного процесса.

*Идентификатор родительского процесса (PPID)*

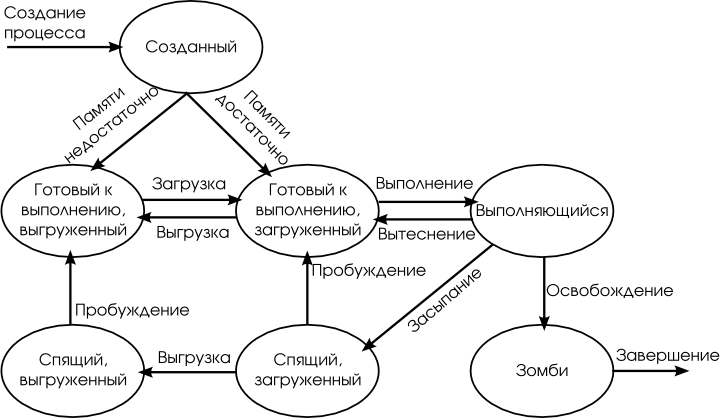
В операционнной системе UNIX процессы выстраиваются в *иерархию* – новый процесс может быть создан в рамках текущего, который выступает для него родительским.

Таким образом, можно построить дерево из процессов, в вершине которого находится *процесс init*, запускающийся при старте системы и являющийся прародителем для всех системных процессов. Подробнее об этом процессе сказано в разделе [«Процесс init»](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch06.html#base-admin-lect-sysinit-init).

*Состояние процесса*

Каждый процесс может находиться в одном из возможных состояний: инициализация, исполнение, приостановка, ожидание ввода-вывода, завершение и т.п. (см. [Рисунок 1.25, «Состояния процесса в UNIX»](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#img-process-states)).

**Рисунок 1.25. Состояния процесса в UNIX**



Большинство этих состояний совпадает с классическим набором состояний процессов в многозадачных операционных системах. Для операционной системы UNIX характерно особое состояние процесса – *зомби*. Это состояние имеет завершившийся процесс, родительский процесс которого еще не закончил работу, и служит для корректного завершния группы процессов, освобождения ресурсов и т.п..

Идентификаторы пользователя

Идентификатор пользователя и группы, от имени которых исполняется процесс. Подробнее о безопасности будет сказано в главе [Глава 3, *Безопасность операционной системы UNIX*](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch03.html).

*Приоритет процесса*

Число, используемое при *планировании* (см. [«Планирование процессов»](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#intro-unix-proc-schedule)) исполнения процесса в операционной системе. Традиционное решение операционной системы UNIX состоит в использовании динамически изменяющихся приоритетов. Каждый процесс при своем образовании получает некоторый устанавливаемый системой статический приоритет, который в дальнейшем может быть изменен с помощью системного вызова nice. Этот статический приоритет является основой начального значения динамического приоритета процесса, являющегося реальным критерием планирования. Все процессы с динамическим приоритетом не ниже порогового участвуют в конкуренции за процессор.

Таблица дескрипторов открытых файлов

Список структур ядра, описывающий все файлы, открытые этим процессом для ввода-вывода.

Другая информация, связанная с процессом

Динамическая часть контекста процесса – это один или несколько стеков, которые используются процессом при его выполнении в режиме ядра. Число ядерных стеков процесса соответствует числу уровней прерывания, поддерживаемых конкретной аппаратурой.

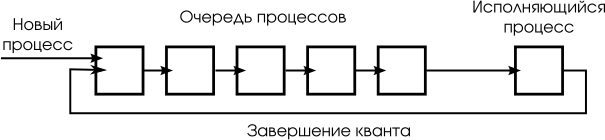
**Планирование процессов**

[Презентация 2-09: планирование процессов](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#small-pres-2-09)

Основной проблемой организации многопользовательского (правильнее сказать, мультипрограммного) режима в любой операционной системе является организация *планирования* «параллельного» выполнения нескольких процессов. Операционная система должна обладать четкими критериями для определения того, какому готовому к выполнению процессу и когда предоставить ресурс процессора.

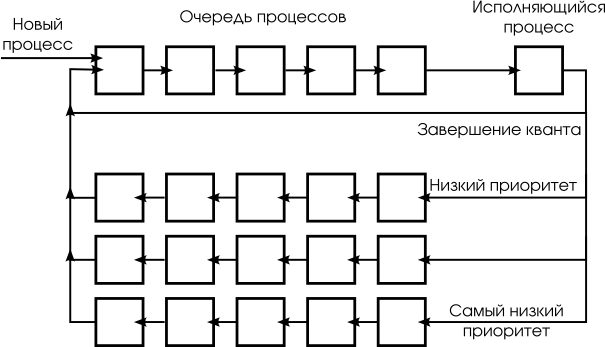
Наиболее распространенным алгоритмом планирования в системах разделения времени является *кольцевой режим* (Round Robin). Основной смысл алгоритма состоит в том, что время процессора делится на кванты фиксированного размера, а процессы, готовые к выполнению, выстраиваются в кольцевую очередь (см. [Рисунок 1.26, «Схема планирования с кольцевой очередью»](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#img-round-robin)). У этой очереди имеются два указателя – начала и конца. Когда процесс, выполняющийся на процессоре, исчерпывает свой квант процессорного времени, он снимается с процессора, ставится в конец очереди, а ресурсы процессора отдаются процессу, находящемуся в начале очереди. Если выполняющийся на процессоре процесс откладывается (например, по причине обмена с некоторым внешнем устройством) до того, как он исчерпает свой квант, то после повторной активизации он становится в конец очереди (не смог доработать - не вина системы). Это прекрасная схема разделения времени в случае, когда все процессы одновременно помещаются в оперативной памяти.

**Рисунок 1.26. Схема планирования с кольцевой очередью**



Однако операционная система UNIX всегда была рассчитана на то, чтобы обслуживать больше процессов, чем можно одновременно разместить в основной памяти. Другими словами, часть процессов, потенциально готовых выполняться, размещалась во внешней памяти (куда образ памяти процесса попадал в результате своппинга). Поэтому требовалась несколько более гибкая схема планирования разделения ресурсов процессора(ов). В результате было введено понятие приоритета (см. [Рисунок 1.27, «Схема планирования с кольцевой очередью и приоритетами»](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#img-round-robin-prio)). В операционной системе UNIX значение приоритета определяет, во-первых, возможность процесса пребывать в основной памяти и на равных конкурировать за процессор. Во-вторых, от значения приоритета процесса, вообще говоря, зависит размер временного кванта, который предоставляется процессу для работы на процессоре при достижении своей очереди. В-третьих, значение приоритета, влияет на место процесса в общей очереди процессов к ресурсу процессора.

**Рисунок 1.27. Схема планирования с кольцевой очередью и приоритетами**



**Межпроцессное взаимодействие**

[Презентация 2-10: межпроцессное взаимодействие](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#small-pres-2-10)

Полная изоляция процессов в операционной системе бессмысленна, так как им частно необходимо обмениваться данными в процессе работы. Операционной системой допускаются контролируемые взаимодействия процессов, в том числе за счет возможности разделения одного сегмента памяти между виртуальной памятью нескольких процессов. Для решения задачи *межпроцессного взаимодействия* в операционной системе существует набор специальных средств:

*Стандартные потоки ввода-вывода*

По умолчанию каждому процессу при запуске ставится в соответствие три открытых файла: стандартного ввода, стандартного вывода и стандартного вывода ошибок. С помощью средств командной строки (см. подробнее в [Глава 2, *Коммандная строка UNIX*](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch02.html)) такие потоки для разных процессов могут быть объединены так, что, к примеру, вывод одного процесса будет подаваться на ввод другого. В более общем смысле такие потоки называют *неименованными каналами*.

*Разделяемая память*

Специальный механизм, с помощью которого средствами операционной системы два процесса могут обращаться к общему участку физической памяти – каждый через свое адресное пространство.

*Сигналы*

Это способ информирования процесса со стороны ядра о происшествии некоторого события. Смысл термина «сигнал» состоит в том, что сколько бы однотипных событий в системе не произошло, по поводу каждой такой группы событий процессу будет подан ровно один сигнал. Т.е. сигнал означает, что определяемое им событие произошло, но не несет информации о том, сколько именно произошло однотипных событий. Сигналы могут инициироваться одними процессами по отношению к другим процессам с помощью специального системного вызова kill.

Именованные каналы и сокеты

Специальные файлы, через которые может осуществляться обмен данными между процессами.

**Резюме**

[Презентация 2-11: резюме](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#small-pres-2-11)

Во второй лекции была рассмотрена архитектура операционной системы UNIX.

Основными принципами операционной системы UNIX являются многозадачность и стандартизация. Пользователю приходиться работать с двумя основными объектами операционной системы: файлами и процессами.

Операционную систему UNIX можно представить в виде трёх уровней: монолитное ядро, системные утилиты и демоны, пользовательские программы. Первые два работают на уровне привилегий системы, третий – на уровне пользователя.

Все файловые системы семейства UNIX обладают схожей структурой. Основным элементом файловой системы является индексный узел, каталог связывает файл с именем. Помимо файлов и директорий существуют специальные виды файлов: устройства, каналы, символические ссылки и сокеты. Стандарт на файловую систему описывает основные директории иерархической файловой системы UNIX. Для объединения файловых систем нескольких устройств используется механизм монтирования.

Каждый процесс в системе имеет уникальный идентификатор и состояние. Планирование исполнения процессов производится на основе динамических приоритетов. Для межпроцессного взяимодействия используются специальные средства, основными среди которых являются неименованные каналы и сигналы.

*Ключевые термины:*[файл](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-unix-file), [процесс](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-unix-process), [двухуровневая модель системы](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-unix-two-levels), [демон](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-daemon), [файловая система](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-fs), [каталог](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-directory), [корневой каталог](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-root-dir), [имя файла](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-file-name), [индексный узел](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-index-node), [жёсткая ссылка](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-hard-link), [суперблок](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-superblock), [файл устройства](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-device-file), [именованый канал](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html" \l "def-named-pipe), [символическая ссылка](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-sym-link), [сокет](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-fs-socket), [монтирование](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-mount), [виртуальная файловая система](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-vfs), [иерархия файловой системы](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-fs-hierarchy), [контекст процесса](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-process-context), [идентификатор процесса](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-pid), [родительский процесс](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-parent-process), [процесс init](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-init-process), [состояние процесса](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-process-state), [зомби](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-zombie), [приоритет процесса](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-process-prio), [планирование](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-scheduling), [Round Robin](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html" \l "def-round-robin), [межпроцессное взаимодействие](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html" \l "def-interprocess-communication), [стандартный ввод-вывод](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-stdio), [неименованный канал](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-anonymous-pipe), [разделяемая память](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-shmem), [сигнал](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#def-signal)

**Дополнительные материалы**

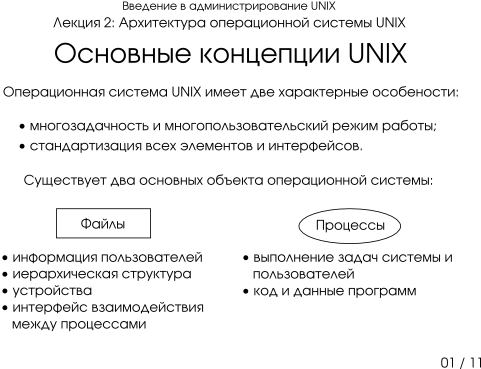
1. Бах Дж.М. *Архитектура операционной системы UNIX*. – <http://www.opennet.ru/docs/RUS/unix/>
2. Курячий Г.В. *Операционная система UNIX*. – М.:Интуит.Ру, 2004. – 292 с.: ил.
3. Робачевский А.М. *Операционная система UNIX*. – СПб.: БХВ-Петербург, 2002. – 528 с.: ил.
4. *Стандарт на файловую систему*. – <http://www.pathname.com/fhs/>
5. Максвелл С. *Ядро Linux в комментариях*. – К.: Издательство «Диа-Софт», 2000. – 488 с.: ил.

**Вопросы**

1. [В чём заключается особенность архитектуры UNIX?](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#quest-2-1)
2. [Какие программы называют демонами? Приведите примеры.](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#quest-2-2)
3. [Из каких частей состоит ядро UNIX?](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#quest-2-3)
4. [Какими отличительными особенностями обладает файловая система UNIX?](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#quest-2-4)
5. [Назовите типы файлов в UNIX. В чём все они схожи, каковы отличия между ними?](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#quest-2-5)
6. [Чем жёсткие ссылки отличаются от символических? Какими преимуществами обладают символические ссылки?](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#quest-2-6)
7. [Какие каталоги стандартизованы в UNIX? Что обычно хранится в каждом из них?](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#quest-2-7)
8. [Что такое контекст процесса? Из чего состоит контекст процесса в UNIX?](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#quest-2-8)
9. [Что такоае планирование и диспетчеризация процессов? Какие в UNIX применяются алгоритмы планирования?](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#quest-2-9)
10. [Какие средства межпроцессного обмена предоставляются в UNIX?](http://heap.altlinux.org/alt-docs/modules/unix_base_admin.dralex/ch01s02.html#quest-2-10)

**Презентация**

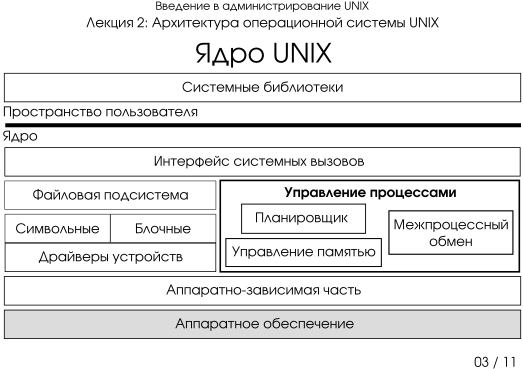
**Рисунок 1.28. Презентация 2-01: основные концепции UNIX**



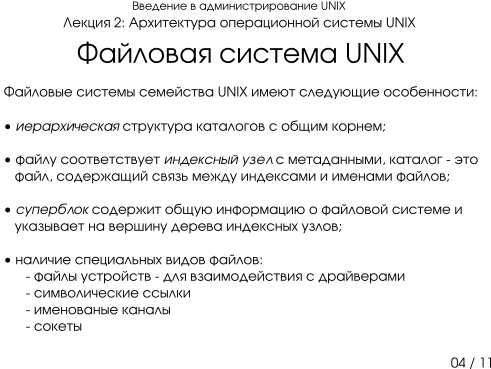
**Рисунок 1.29. Презентация 2-02: беглый взгляд на архитектуру UNIX**



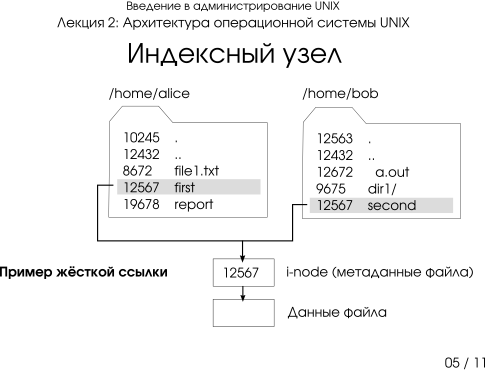
**Рисунок 1.30. Презентация 2-03: ядро UNIX**



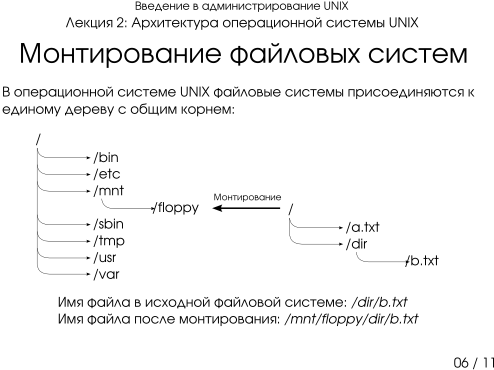
**Рисунок 1.31. Презентация 2-04: файловая система UNIX**



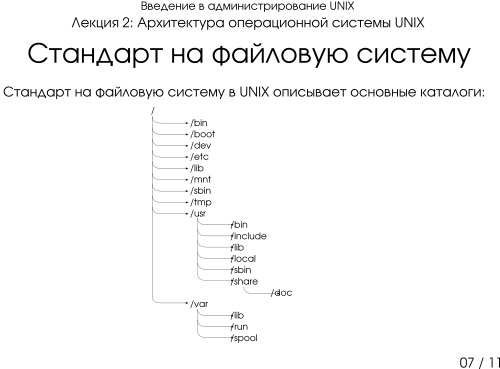
**Рисунок 1.32. Презентация 2-05: индексный узел**



**Рисунок 1.33. Презентация 2-06: монтирование файловых систем**



**Рисунок 1.34. Презентация 2-07: стандарт на файловую систему**



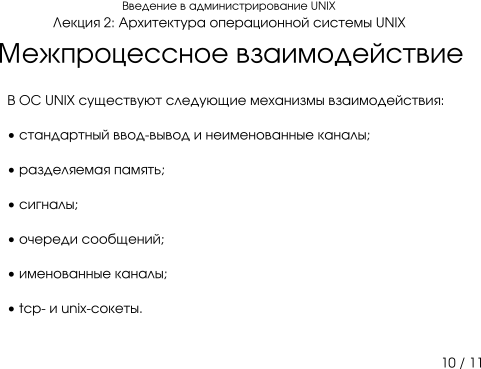
**Рисунок 1.35. Презентация 2-08: контекст процесса**



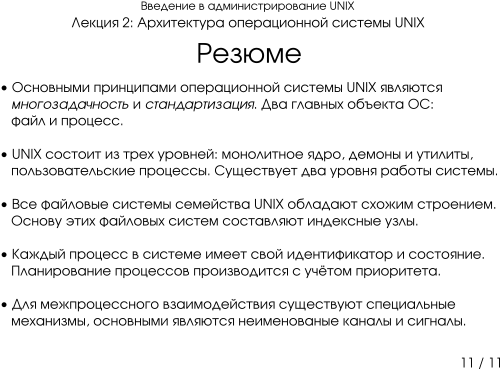
**Рисунок 1.36. Презентация 2-09: планирование процессов**



**Рисунок 1.37. Презентация 2-10: межпроцессное взаимодействие**



**Рисунок 1.38. Презентация 2-11: резюме**



ПОДРОБНОЕ ОПИСАНИЕ СИСТЕМЫ: <https://www.opennet.ru/docs/RUS/unix/>