# Лекция 5. ОС UNIX



## 1 История создания системы UNIX

Исторически системы реального времени создавались в эпоху расцвета и бума UNIX'а и поэтому многие из них содержат те или иные заимствования из этой красивой концепции операционный системы (пользовательский интерфейс, концепция процессов и т.д.). Часть разработчиков операционных систем реального времени попыталась просто переписать ядро UNIX, сохранив при этом интерфейс пользовательских процессов с системой, насколько это было возможно. Реализация этой идеи не была слишком сложной, поскольку не было препятствия в доступе к исходным текстам ядра, а результат оказался замечательным. Получили и реальное время и сразу весь набор пользовательских приложений - компиляторы, пакеты, различные инструментальные системы. В этом смысле создателям систем первых двух классов пришлось потрудиться не только при создании ядра реального времени, но и продвинутых систем разработки.

Однако UNIX'ы реального времени не избавлены от следующих ***недостатков***: системы реального времени получаются достаточно большими и реактивность их ниже, чем реактивность систем первых двух классов. Наиболее популярным представителем систем этого класса является операционная система реального времени Lynx OS.

В 1965 году фирма Bell Telephone Laboratories, объединив свои усилия с компанией General Electric и проектом MAC Массачусетского технологического института, приступили к разработке новой операционной системы, получившей название Multics (англ. Multiplexed Information and Computing Service — «Мультиплексная информационная и вычислительная служба») [Organick 72].

Перед системой Multics были поставлены задачи:

- обеспечить одновременный доступ к ресурсам ЭВМ большого количества пользователей
- обеспечить достаточную скорость вычислений и хранение данных
- дать возможность пользователям в случае необходимости совместно использовать данные.

Многие разработчики, впоследствии принявшие участие в создании ранних редакций системы UNIX, участвовали в работе над системой Multics в фирме Bell Laboratories. Хотя первая версия системы Multics и была запущена в 1969 году на ЭВМ GE 645, она не обеспечивала выполнение главных вычислительных задач, для решения которых она предназначалась, и не было даже ясно, когда цели разработки будут достигнуты. Поэтому фирма Bell Laboratories прекратила свое участие в проекте.

По окончании работы над проектом Multics сотрудники Исследовательского центра по информатике фирмы Bell Laboratories остались без "достаточно интерактивного вычислительного средства". Пытаясь усовершенствовать среду(ОС) программирования, Кен Томпсон, Дэннис Ричи и другие набросали на бумаге проект файловой системы, получивший позднее дальнейшее развитие в ранней версии файловой системы UNIX. Томпсоном были написаны программы, имитирующие поведение предложенной файловой системы в режиме подкачки данных по запросу, им было даже создано простейшее ядро операционной системы для ЭВМ GE 645. В то же время он написал на Фортране игровую программу "Space Travel" ("Космическое путешествие") для системы GECOS (Honeywell 635), но программа не смогла удовлетворить пользователей, поскольку управлять "космическим кораблем" оказалось сложно, кроме того, при загрузке программа занимала много места. Позже Томпсон обнаружил малоиспользуемый компьютер PDP-

7, оснащенный хорошим графическим дисплеем и имеющий дешевое машинное время. Создавая программу "Космическое путешествие" для PDP-7, Томпсон получил возможность изучить машину, однако условия разработки программ потребовали использования кросс-ассемблера для трансляции программы на машине с системой GECOS и использования перфоленты для ввода в PDP-7. Для того чтобы улучшить условия разработки, Томпсон и Ричи выполнили на PDP-7 свой проект системы, включивший

- первую версию файловой системы UNIX
- подсистему управления процессами
- и небольшой набор утилит.

В конце концов, новая система больше не нуждалась в поддержке со стороны системы GECOS в качестве операционной среды разработки и могла поддерживать себя сама. Новая система получила название UNIX, по сходству с Multics его придумал еще один сотрудник Исследовательского центра по информатике Брайан Керниган.

Несмотря на то, что эта ранняя версия системы UNIX уже была многообещающей, она не могла реализовать свой потенциал до тех пор, пока не получила применение в реальном проекте. Так, для того, чтобы обеспечить функционирование системы обработки текстов для патентного отдела фирмы Bell Laboratories, в 1971 году система UNIX была перенесена на ЭВМ PDP-11. Система отличалась небольшим объемом: 16 Кбайт для системы, 8 Кбайт для программ пользователей, обслуживала диск объемом 512 Кбайт и отводила под каждый файл не более 64 Кбайт. После своего первого успеха Томпсон собрался было написать для новой системы транслятор с Фортрана, но вместо этого занялся языком Би (B), предшественником которого явился язык BCPL. Би был интерпретируемым языком со всеми недостатками, присущими подобным языкам, поэтому Ричи переделал его в новую разновидность, получившую название ***Си*** (C) и разрешающую генерировать машинный код, объявлять типы данных и определять структуру данных.

В 1973 году система была написана заново на Си, это был шаг, неслыханный для того времени, но имевший огромный резонанс среди сторонних пользователей. Количество машин фирмы Bell Laboratories, на которых была инсталлирована система, возросло до 25, в результате чего была создана группа по системному сопровождению UNIX внутри фирмы.

В то время корпорация AT&T не могла заниматься продажей компьютерных продуктов в связи с соответствующим соглашением, подписанным ею с федеральным правительством в 1956 году, и распространяла систему UNIX среди университетов, которым она была нужна в учебных целях. Следуя букве соглашения, корпорация AT&T не рекламировала, не продавала и не сопровождала систему. Несмотря на это, популярность системы устойчиво росла. В 1974 году Томпсон и Ричи опубликовали статью, описывающую систему UNIX, в журнале Communications of the ACM [Thompson 74], что дало еще один импульс к распространению системы.

К 1977 году количество машин, на которых функционировала система UNIX, увеличилось до 500, при чем 125 из них работали в университетах. Система UNIX завоевала популярность среди *телефонных компаний*, поскольку обеспечивала хорошие условия для разработки программ, обслуживала работу в сети в режиме диалога и работу в реальном масштабе времени (с помощью системы MERT). Помимо университетов, лицензии на систему UNIX были переданы коммерческим организациям.

В 1977 году корпорация Interactive Systems стала первой организацией, получившей права на перепродажу системы UNIX с надбавкой к цене за дополнительные услуги, которые заключались в адаптации системы к функционированию в автоматизированных системах управления учрежденческой деятельностью. 1977 год также был отмечен "переносом" системы UNIX на машину, отличную от PDP (благодаря чему стал возможен запуск системы на другой машине без изменений или с небольшими изменениями), а именно на Interdata 8/32.

С ростом популярности микропроцессоров другие компании стали переносить систему UNIX на новые машины, однако ее простота и ясность побудили многих разработчиков к самостоятельному развитию системы, в результате чего было создано несколько вариантов базисной системы. За период между 1977 и 1982 годом фирма Bell Laboratories объединила несколько вариантов, разработанных в корпорации AT&T, в один, получивший коммерческое название UNIX версия III.

В начале 1983 года компания American Telephone and Telegraph Bell Laboratories (AT&T Bell Labs) объявила о выпуске UNIX System V. Впервые в истории Bell Labs было также объявлено, что AT&T будет поддерживать этот и все будущие выпуски System V. Кроме того, была обещана совместимость выпущенной версии System V со всеми будущими версиями. ОС UNIX System V включала много новых возможностей, но почти все они относились к повышению производительности (хеш-таблицы и кэширование данных). На самом деле UNIX System V являлась развитым вариантом UNIX System III. К наиболее важным оригинальным особенностям UNIX System V относится появление семафоров, очередей сообщений и разделяемой памяти.

В 1984 году USG была преобразована в Лабораторию по развитию системы UNIX (UNIX System Development Laboratories - USDL). В 1984 году USDL выпустила UNIX System V Release 2 (SVR2). В этом варианте системы появились возможности блокировок файлов и записей, копирования совместно используемых страниц оперативной памяти при попытке записи (copy-on-write), страничного замещения оперативной памяти (реализованного не так, как в BSD) и т.д. К этому времени ОС UNIX была установлена на более чем 100 000 компьютеров.

В 1987 году подразделение USDL объявило о выпуске UNIX System V Release 3 (SVR3). В этой системе появились полные возможности межпроцессных взаимодействий, разделения удаленных файлов (Remote File Sharing - RFS), развитые операции обработки сигналов, разделяемые библиотеки и т.д. Кроме того, были обеспечены новые возможности по повышению производительности и безопасности системы. К концу 1987 года появилось более 750 000 установок ОС UNIX, и было зарегистрировано 4,5 млн. пользователей.

Т.о. за время, прошедшее с момента ее появления в 1969 году, система UNIX стала довольно популярной и получила распространение на машинах с различной мощностью обработки, от микропроцессоров до больших ЭВМ, обеспечивая на них общие условия выполнения программ. Ни о какой другой операционной системе нельзя было бы сказать того же. Популярность и успех системы UNIX объяснялись несколькими причинами:

- Система ***написана на языке высокого уровня***, благодаря чему ее легко читать, понимать, изменять и переносить на другие машины. По оценкам, сделанным Ричи, первый вариант системы на С имел на 20-40 % больший объем и работал медленнее по сравнению с вариантом на ассемблере, однако преимущества использования языка высокого уровня намного перевешивают недостатки.
- ***Наличие довольно простого пользовательского интерфейса*** с периферийными устройствами, в котором имеется возможность предоставлять все необходимые пользователю услуги.
- Наличие элементарных средств, позволяющих создавать сложные программы из более простых.
- ***Наличие иерархической файловой системы***, легкой в сопровождении и эффективной в работе.
- ***Обеспечение согласования форматов в файлах***, работа с последовательным потоком байтов, благодаря чему облегчается чтение прикладных программ.
- Система является ***многопользовательской, многозадачной***; каждый пользователь может одновременно выполнять несколько процессов.

Вся система UNIX делится на две части:

1. Одну часть составляют ***программы и сервисные функции***, то, что делает операционную среду UNIX такой популярной; эта часть легко доступна пользователям, она включает такие программы, как командный процессор, обмен сообщениями, пакеты обработки текстов и системы обработки исходных текстов программ.
2. Другая часть включает в себя собственно ***операционную систему***, поддерживающую эти программы и функции.

Т.о. архитектура машины скрыта от пользователя, благодаря этому облегчен процесс написания программ, работающих на различных конфигурациях аппаратных средств.

Простота и последовательность вообще отличают систему UNIX и объясняют большинство из вышеприведенных доводов в ее пользу.

Хотя операционная система и большинство команд написаны на Си, система UNIX поддерживает ряд других языков, таких как *Фортран, Бейсик, Паскаль, Ада, Кобол, Лисп и*

*Пролог*. Система UNIX может поддерживать любой язык программирования, для которого имеется компилятор или интерпретатор, и обеспечивать системный интерфейс, устанавливающий соответствие между пользовательскими запросами к операционной системе и набором запросов, принятых в UNIX.

Организации, получившие права на перепродажу с надбавкой к цене за дополнительные услуги, оснащают вычислительную систему прикладными программами, касающимися конкретных областей применения, стремясь удовлетворить требования рынка. Такие организации чаще продают прикладные программы, нежели операционные системы, под управлением которых эти программы работают.

## 2 Основные понятия, используемые в системе UNIX

Одним из достоинств ОС UNIX является то, что система базируется на небольшом числе интуитивно ясных понятий. Однако, несмотря на простоту этих понятий, к ним нужно привыкнуть. Без этого невозможно понять существо ОС UNIX.

### Пользователь

С самого начала ОС UNIX замышлялась как интерактивная система. Другими словами, UNIX предназначена для терминальной работы. Чтобы начать работать, человек должен "войти" в систему, введя со свободного терминала свое учетное имя (account name) и, возможно, пароль (password). Человек, зарегистрированный в учетных файлах системы, и, следовательно, имеющий учетное имя, называется ***зарегистрированным пользователем системы***. Регистрацию новых пользователей обычно выполняет администратор системы. Пользователь не может изменить свое учетное имя, но может установить и/или изменить свой пароль. Пароли хранятся в отдельном файле в закодированном виде. Не забывайте свой пароль, снова узнать его не поможет даже администратор!

Все пользователи ОС UNIX явно или неявно работают с файлами. Файловая система ОС UNIX имеет древовидную структуру. Промежуточными узлами дерева являются каталоги со ссылками на другие каталоги или файлы, а листья дерева соответствуют файлам или пустым каталогам. Каждому зарегистрированному пользователю соответствует некоторый каталог файловой системы, который называется "домашним" (home) каталогом пользователя. При входе в систему пользователь получает неограниченный доступ к своему домашнему каталогу и всем каталогам и файлам, содержащимся в нем. Пользователь может создавать, удалять и модифицировать каталоги и файлы, содержащиеся в домашнем каталоге. Потенциально возможен доступ и ко всем другим файлам, однако он может быть ограничен, если пользователь не имеет достаточных привилегий.

### Интерфейс пользователя

Традиционный способ взаимодействия пользователя с системой UNIX основывается на использовании командных языков (правда, в настоящее время все большее распространение получают графические интерфейсы). После входа пользователя в систему для него запускается один из командных интерпретаторов (в зависимости от параметров, сохраняемых в файле /etc/passwd). Обычно в системе поддерживается несколько командных интерпретаторов с похожими, но различающимися своими возможностями командными языками. Общее название для любого командного интерпретатора ОС UNIX - ***shell*** (оболочка), поскольку любой интерпретатор представляет внешнее окружение ядра системы.

Вызванный командный интерпретатор выдает приглашение на ввод пользователем командной строки, которая может содержать простую команду, конвейер команд или последовательность команд. После выполнения очередной командной строки и выдачи на экран терминала или в файл соответствующих результатов, shell снова выдает приглашение на ввод командной строки, и так до тех пор, пока пользователь не завершит свой сеанс работы путем ввода команды logout или нажатием комбинации клавиш Ctrl-d.

Командные языки, используемые в ОС UNIX, достаточно просты, чтобы новые пользователи могли быстро начать работать, и достаточно мощны, чтобы можно было использовать их для написания сложных программ. Последняя возможность опирается на механизм командных файлов (shell scripts), которые могут содержать произвольные последовательности командных строк. При указании имени командного файла вместо очередной команды интерпретатор читает файл строка за строкой и последовательно интерпретирует команды.

**Привилегированный пользователь**

Ядро ОС UNIX идентифицирует каждого пользователя по его идентификатору (UID - User Identifier), уникальному целому значению, присваиваемому пользователю при регистрации в системе. Кроме того, каждый пользователь относится к некоторой группе пользователей, которая также идентифицируется некоторым целым значением (GID - Group IDentifier). Значения UID и GID для каждого зарегистрированного пользователя сохраняются в учетных файлах системы и приписываются процессу, в котором выполняется командный интерпретатор, запущенный при входе пользователя в систему. Эти значения наследуются каждым новым процессом, запущенным от имени данного пользователя, и используются ядром системы для контроля правомочности доступа к файлам, выполнения программ и т.д.

Понятно, что администратор системы, который, естественно, тоже является зарегистрированным пользователем, должен обладать большими возможностями, чем обычные пользователи. В ОС UNIX эта задача решается путем выделения одного значения UID (нулевого). Пользователь с таким UID называется суперпользователем (superuser) или root. Он имеет неограниченные права на доступ к любому файлу и на выполнение любой программы. Кроме того, такой пользователь имеет возможность полного контроля над системой. Он может остановить ее и даже разрушить.

В мире UNIX считается, что человек, получивший статус суперпользователя, должен понимать, что делает. Суперпользователь должен хорошо знать базовые процедуры администрирования ОС UNIX. Он отвечает за безопасность системы, ее правильное конфигурирование, добавление и исключение пользователей, регулярное копирование файлов и т.д.

Еще одним отличием суперпользователя от обычного пользователя ОС UNIX является то, что на суперпользователя не распространяются ограничения на используемые ресурсы. Для обычных пользователей устанавливаются такие ограничения как максимальный размер файла, максимальное число сегментов разделяемой памяти, максимально допустимое пространство на диске и т.д. Суперпользователь может изменять эти ограничения для других пользователей, но на него они не действуют.

**Программы**

ОС UNIX одновременно является операционной средой использования существующих прикладных программ и средой разработки новых приложений. Новые программы могут писаться на разных языках (Фортран, Паскаль, Модула, Ада и др.). Однако стандартным языком программирования в среде ОС UNIX является язык С (который в последнее время все больше заменяется на С++). Это объясняется тем, что, во-первых, сама система UNIX написана на языке С, а, во-вторых, язык С является одним из наиболее качественно стандартизованных языков.

Поэтому программы, написанные на языке С, при использовании правильного стиля программирования обладают весьма высоким уровнем мобильности, т.е. их можно достаточно просто переносить на другие аппаратные платформы, работающие как под управлением ОС UNIX, так и под управлением ряда других операционных систем (например, DEC Open VMS или MS Windows NT).

Выполняемая программа может быть запущена в интерактивном режиме как команда shell или выполнена в отдельном процессе, образуемом уже запущенной программой.

**Команды**

Любой командный язык семейства shell фактически состоит из трех частей:

1. ***служебных конструкций***, позволяющих манипулировать с текстовыми строками и строить сложные команды на основе простых команд;
2. ***встроенных команд***, выполняемых непосредственно интерпретатором командного языка;
3. ***команд, представляемых отдельными выполняемыми файлами***.

**Процессы**

***Процесс*** в ОС UNIX - это программа, выполняемая в собственном виртуальном адресном пространстве. Когда пользователь входит в систему, автоматически создается процесс, в котором выполняется программа командного интерпретатора. Если командному интерпретатору встречается команда, соответствующая выполняемому файлу, то он создает новый процесс и запускает в нем соответствующую программу, начиная с функции main. Эта запущенная программа, в свою очередь, может создать процесс и запустить в нем другую программу (она тоже должна содержать функцию main) и т.д.

**Перенаправление ввода/вывода**

Механизм перенаправления ввода/вывода является одним из наиболее элегантных, мощных и одновременно простых механизмов ОС UNIX. Цель, которая ставилась при разработке этого механизма, состоит в следующем. Поскольку UNIX - это интерактивная система, то обычно программы вводят текстовые строки с терминала и выводят результирующие текстовые строки на экран терминала. Для того чтобы обеспечить более гибкое использование таких программ, желательно уметь обеспечить им ввод из файла или из вывода других программ и направить их вывод в файл или на ввод другим программам.

Реализация механизма основывается на следующих свойствах ОС UNIX.

1. Во-первых, любой ввод/вывод трактуется как ввод из некоторого файла и вывод в некоторый файл. Клавиатура и экран терминала тоже интерпретируются как файлы (первый можно только читать, а во второй можно только писать).
2. Во-вторых, доступ к любому файлу производится через его дескриптор (положительное целое число). Фиксируются три значения дескрипторов файлов. Файл с дескриптором ***1 - называется файлом стандартного ввода*** (stdin), файл с дескриптором ***2 - файлом стандартного вывода*** (stdout), и файл с дескриптором ***3 - файлом стандартного вывода диагностических сообщений*** (stderr).
3. В-третьих, программа, запущенная в некотором процессе, "наследует" от породившего процесса все дескрипторы открытых файлов.

**Именованные программные каналы.**

Программный канал (pipe) - это одно из наиболее традиционных средств межпроцессных взаимодействий в ОС UNIX.

Основной принцип работы программного канала состоит в буферизации байтового вывода одного процесса и обеспечении возможности чтения содержимого программного канала другим процессом в режиме FIFO (т.е. первым будет прочитан байт, который раньше всего записан). В любом случае интерфейс программного канала совпадает с интерфейсом файла (т.е. используются те же самые системные вызовы read и write). Однако различаются два подвида программных каналов - неименованные и именованные.

***Неименованный программный канал*** создается процессом-предком, наследуется процессами-потомками, и обеспечивает тем самым возможность связи в иерархии порожденных процессов. Интерфейс неименованного программного канала совпадает с интерфейсом файла. Однако, поскольку такие каналы не имеют имени, им не соответствует какой-либо элемент каталога в файловой системе.

***Именованному программному каналу*** обязательно соответствует элемент некоторого каталога и даже собственный i-узел. Другими словами, именованный программный канал выглядит как обычный файл, но не содержащий никаких данных до тех пор, пока некоторый процесс не выполнит в него запись. После того, как некоторый другой процесс прочитает

записанные в канал байты, этот файл снова становится пустым. В отличие от неименованных программных каналов, именованные программные каналы могут использоваться для связи любых процессов (т.е. не обязательно процессов, входящих в одну иерархию родства).

**Принципы защиты.**

Поскольку ОС UNIX - многопользовательская операционная система, в ней всегда была актуальна проблема авторизации доступа различных пользователей к файлам файловой системы. Под авторизацией доступа понимают действия системы, которые допускают или не допускают доступ данного пользователя к данному файлу в зависимости от прав доступа пользователя и ограничений доступа, установленных для файла.

Идентификаторы пользователя и группы пользователей. С каждым выполняемым процессом в ОС UNIX связываются ***реальный идентификатор пользователя*** (real user ID), ***действующий идентификатор пользователя*** (effective user ID) и ***сохраненный идентификатор пользователя*** (saved user ID). Все эти идентификаторы устанавливаются с помощью системного вызова, который можно выполнять только в режиме суперпользователя. Аналогично, с каждым процессом связываются три идентификатора группы пользователей - ***real group ID***, ***effective group ID*** и ***saved group ID***. Эти идентификаторы устанавливаются привилегированным системным вызовом.

При входе пользователя в систему программа login проверяет, что пользователь зарегистрирован в системе и знает правильный пароль (если он установлен), образует новый процесс и запускает в нем требуемый для данного пользователя shell. Но перед этим login устанавливает для вновь созданного процесса идентификаторы пользователя и группы, используя для этого информацию, хранящуюся в файлах /etc/passwd и /etc/group. После того, как с процессом связаны идентификаторы пользователя и группы, для этого процесса начинают действовать ограничения для доступа к файлам. Процесс может получить доступ к файлу или выполнить его (если файл содержит выполняемую программу) только в том случае, если хранящиеся при файле ограничения доступа позволяют это сделать. Связанные с процессом идентификаторы передаются создаваемым им процессам, распространяя на них те же ограничения.

**Управление устройствами.**

Для доступа к внешним устройствам в ОС UNIX используется универсальная абстракция файла. Помимо настоящих файлов (обычных файлов или каталогов), которые реально занимают память на магнитных дисках, файловая система содержит так называемые специальные файлы, для которых, как и для настоящих файлов, отводятся отдельные i-узлы, но которым на самом деле соответствуют внешние устройства. Это решение позволяет естественным образом работать в одном и том же интерфейсе с любым файлом или внешним устройством.

**Драйверы устройств.**

***Драйвер устройства*** - это многовходовой программный модуль со своими статическими данными, который умеет инициировать работу с устройством, выполнять заказываемые пользователем обмены (на ввод или вывод данных), терминировать работу с устройством и обрабатывать прерывания от устройства. В ОС UNIX различаются символьные, блочные и потоковые драйверы.

***Символьные драйверы*** являются простейшими и предназначаются для обслуживания устройств, которые реально ориентированы на прием или выдачу произвольных последовательностей байтов (например, простой). Такие драйверы используют минимальный набор стандартных функций ядра UNIX, которые главным образом заключаются в возможности взять данные из виртуального пространства пользовательского процесса и/или поместить данные в такое виртуальное пространство.

***Блочные драйверы*** работают с использованием возможностей системной буферизации блочных обменов ядра ОС UNIX. В число функций такого драйвера входит включение соответствующего блока данных в систему буферов ядра ОС UNIX и/или взятие содержимого буферной области в случае необходимости.

Наиболее сложной организацией отличаются ***потоковые драйверы***. Фактически, такой драйвер представляет собой конвейер модулей, обеспечивающий многоступенчатую обработку

запросов пользователя. Потоковые драйверы в среде ОС UNIX в основном предназначены для реализации доступа к сетевым устройствам, которые должны работать в соответствии с многоуровневыми сетевыми протоколами.

**Внешний и внутренний интерфейсы устройств.**

Независимо от типа файла (обычный файл, каталог, связь или специальный файл) пользовательский процесс может работать с файлом через стандартный интерфейс, включающий системные вызовы open, close, read и write. Ядро само распознает, нужно ли обратиться к его стандартным функциям или вызвать подпрограмму драйвера устройства. Другими словами, если процесс пользователя открывает для чтения обычный файл, то системные вызовы open и read обрабатываются встроенными в ядро подпрограммами open и read соответственно. Однако, если файл является специальным, то будут вызваны подпрограммы open и read, определенные в соответствующем драйвере устройства.

## 3 Структура системы

Операционная система UNIX - это набор программ, который управляет компьютером, осуществляет связь между вами и компьютером и обеспечивает вас инструментальными средствами, чтобы помочь вам выполнить вашу работу. Разработанная, чтобы обеспечить легкость, эффективность и гибкость программного обеспечения, система UNIX имеет несколько полезных функций:

- ***основная цель системы*** - это выполнять широкий спектр заданий и программ;
- ***интерактивное окружение***, которое позволяет вам связываться напрямую с компьютером и получать немедленно ответы на ваши запросы и сообщения;
- ***многопользовательское окружение***, которое позволяет вам разделять ресурсы компьютера с другими пользователями без уменьшения производительности. Этот метод называется разделением времени. Система UNIX взаимодействует с пользователями поочередно, но так быстро, что кажется, что взаимодействует со всеми пользователями одновременно;
- ***многозадачное окружение***, которое позволяет вам выполнять более одного задания в одно и тоже время.

В некоторых реализациях системы UNIX операционная система взаимодействует с собственной операционной системой, которая, в свою очередь, взаимодействует с аппаратурой и выполняет необходимые функции по обслуживанию системы. В таких реализациях допускается инсталляция других операционных систем с загрузкой под их управлением прикладных программ параллельно с системой UNIX. Классическим примером подобной реализации явилась система MERT [Lycklama 78a]. Более новым примером могут служить реализации для компьютеров серии IBM 370 и UNIVAC 1100.

Система UNIX имеет 4 основных компонента:

- ***ядро*** –
  это программа, которая образует ядро операционной системы; она координирует внутренние функции компьютера (такие как размещение системных ресурсов). Ядро работает невидимо для вас;
- ***shell*** –
  это программа, которая осуществляет связь между вами и ядром, интерпретируя и выполняя ваши команды. Так как она читает ваш ввод и посылает вам сообщения, то описывается как интерактивная;
- ***commands*** –
  это имена программ, которые компьютер должен выполнить. Пакеты программ называются инструментальными средствами. Система UNIX обеспечивает инструментальными средствами для таких заданий как создание и изменение текста, написание программ, развитие инструментария программного обеспечения, обмен информацией с другими посредством компьютера;
- ***file system*** –

файловая система - это набор всех файлов, возможных для вашего компьютера. Она помогает вам легко сохранять и отыскивать информацию.

Самый общий взляд на архитектуру UNIX позволяет увидеть *двухуровневую модель системы*, состоящую из *пользовательской* и *системной части (ядра)* 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).

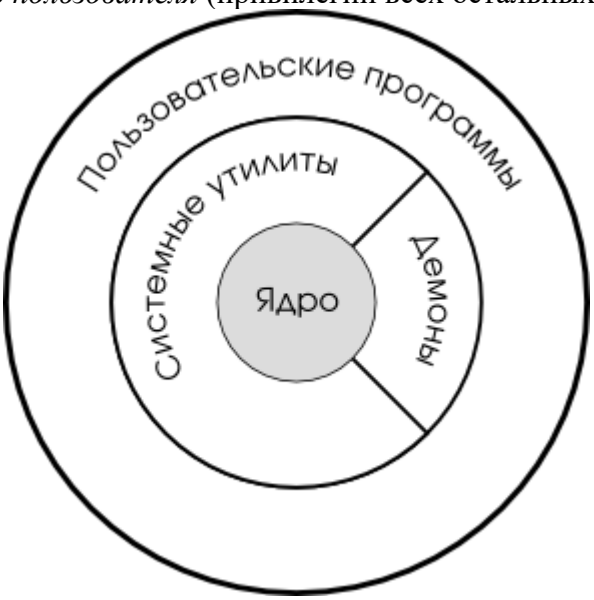


Рисунок 1 - Архитектура операционной системы UNIX

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*.

**Программные гнезда (Sockets).**

Механизм программных гнезд (Sockets) реализован в качестве развитого средства межпроцессных взаимодействий. Это средство, вообще говоря, позволяет любому процессу обмениваться сообщениями с любым другим процессом, независимо от того, выполняются они на одном компьютере или на разных, соединенных сетью.

Программные гнезда входят в число обязательных компонентов стандартной среды ОС UNIX, однако реализуются в разных системах по-разному.

**Вызовы удаленных процедур (RPC).**

Основными идеями механизма вызова удаленных процедур (RPC - Remote Procedure Calls) являются следующие:

(а) Во многих случаях взаимодействие процессов носит ярко выраженный асимметричный характер. Один из процессов ("клиент") запрашивает у другого процесса ("сервера") некоторую услугу (сервис) и не продолжает свое выполнение до тех пор, пока эта услуга не будет выполнена (и пока процесс-клиент не получит соответствующие результаты).

(б) Свойства переносимости позволяют, в частности, предельно просто создавать "операционно однородные" сети, включающие разнородные компьютеры, однако, остается проблема разного представления данных в компьютерах разной архитектуры. Поэтому второй идеей RPC является автоматическое обеспечение преобразования форматов данных при взаимодействии процессов, выполняющихся на разнородных компьютерах.

Независимость от конкретного машинного представления данных обеспечивается отдельно специфицированным протоколом XDR (EXternal Data Representation - внешнее представление данных). Этот протокол определяет стандартный способ представления данных, скрывающий такие машинно-зависимые свойства, как порядок байтов в слове, требования к выравниванию начального адреса структуры, представление стандартных типов данных и т.д. По существу, XDR реализуется как независимый пакет, который используется не только в RPC, но и других продуктах (например, в NFS).

**Упражнения к лекции.**

1. Какие основные достоинства ОС Unix?
2. На какие две большие части делится ОС Unix?
3. Пользователи ОС Unix.
4. Интерфейс пользователя ОС Unix.
5. Привилегированный пользователь ОС Unix.
6. Программы ОС Unix.
7. Команды ОС Unix.
8. Процессы ОС Unix.
9. Перенаправление ввода/вывода ОС Unix.
10. Именованные программные каналы ОС Unix.

**Литература к лекции.**

3.1 А.В.Гордеев, А.Ю.Молчанов. Системное программное обеспечение. — "Питер", 2002. — 736с.

3.2 Кристиан К. Введение в операционную систему Unix: пер. с англ. — М. Финансы и статистика, 1985. – 360с.

3.3 Робачевский А.М., Немнюгин С.А., Стесик О.Л. Операционная система Unix. 2-е изд.– СПб.: БХВ – Петербург, 2005. – 635с.