**Лекция № 8-2. История Unix и Linux. Обзор системы Unix. Задачи. Интерфейс системы. Архитектура, особенности построения и функционирования операционной системы Unix. Структура ядра. Утилиты. Процессы в системе Unix. Загрузка Unix.**

 **Цель занятия:** Сформировать знания студентов об истории операционных систем Unix и Linux, а так же произвести обзор операционной системы Unix.

**ИСТОРИЯ**

 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 [Organick 72]. Перед системой Multics были поставлены задачи - обеспечить одновременный доступ к ресурсам ЭВМ большого количества пользователей, обеспечить достаточную скорость вычислений и хранение данных и 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" [Ritchie 84a]. Пытаясь усовершенствовать среду 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 [Richards 69]. Би был интерпретируемым языком со всеми недостатками, присущими подобным языкам, поэтому Ричи переделал его в новую разновидность, получившую название Си | 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 [Lycklama 78a]). 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 дальнейшем фирма Bell Laboratories добавила в версию III несколько новых особенностей, назвав новый продукт UNIX версия V (\*\*), и эта версия стала официально распространяться корпорацией AT&T с января 1983 года. В то же время сотрудники Калифорнийского университета в Бэркли разработали вариант системы UNIX, получивший название BSD 4.3 для машин серии VAX и отличающийся некоторыми новыми, интересными особенностями.

К началу 1984 года система UNIX была уже инсталлирована приблизительно на 100000 машин по всему миру, при чем на машинах с широким диапазоном вычислительных возможностей - от микропроцессоров до больших ЭВМ - и разных изготовителей. 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 Си 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, 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.

\* Наличие простого, последовательного интерфейса с периферийными устройствами.

\* Система является многопользовательской, многозадачной; каждый пользователь может одновременно выполнять несколько процессов.

\* 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, системаUNIX 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.

Хотя система UNIX и имеет более модульную структуру, чем MS DOS, ее архитектура ограничена функциональностью аппаратуры, для которой она были первоначально разработана, - миникомпьютеров. Поэтому первые версии UNIX имели ограниченное структурирование.

Система UNIX состоит из двух частей: **системные программы**и **ядро**.

Ядро содержит все модули, *уровень абстракции* которых ниже системных вызовов, но выше непосредственно аппаратных модулей.

UNIX обеспечивает поддержку файловой системы, диспетчеризацию процессора, *управление памятью* и другие основные функции ОС.

Архитектура UNIX изображена на рис. 1.



**Рис. 1.**  Структура системы UNIX

В архитектуре UNIX уже четко прослеживаются три уровня абстракции – пользовательский (*системные программы*), системных вызовов и низкоуровневых модулей взаимодействия с аппаратурой.

**Уровни абстракции**

В конце 1960-х гг., при разработке операционной системы THE (название – аббревиатура, означающая "Технический университет Эйндховена"), Э. Дейкстра предложил для своего времени весьма новый и прогрессивный принцип уровней абстракции, полезный при разработке любой сложной программной системы, в том числе – столь сложной, как операционная система. Согласно этому принципу, ОС (или другая сложная программа) реализуется в виде набора (иерархии) **уровней абстракции (abstraction layers)**,каждый из которых реализован на основе предыдущего уровня. **Уровень 0 (layer 0)**образует **аппаратура**( **hardware** ); самый высокий уровень N (layer N) является пользовательским интерфейсом с операционной системой. Каждый *уровень абстракции* N > 0 – это группа модулей, при реализации которого, согласно принципам *модульного программирования*, используются только модули предшествующего уровня (N-1).

"Перескакивание" через уровень (т.е., например, использование при реализации модуля уровня N вызовов модулей уровня N – 2) не рекомендуется и является нарушением технологии, которое может привести к ошибкам.

Подобный подход позволяет проектировать сложную программную систему шаг за шагом, снизу вверх, причем на каждом шаге (уровне K) используется все более и более удобная система обозначений уровня K-1. Это позволяет абстрагироваться от лишних деталей, что и является объяснением названия данного метода.

Заметим, что, по сути дела, уровни абстракции – движущая сила и принцип развития всего программного обеспечения в целом, а не только операционных систем. Каждая новая программа разрабатывается не с нуля, а на некотором достаточно высоком уровне абстракции, используя другие уже разработанные системы.

Первая *версия многозадачной многопользовательской системы UNIX* была разработана в 1969 году сотрудниками подразделения *Bell Labs* фирмы AT&T Деннисом Ритчи (Dennis Ritchie) и Кеном Томпсоном (Ken Thompson). Она была написана на ассемблере для компьютера PDP-7. К лету 1972 года систему переписали на язык высокого уровня С, что позволяло перенести ее на любую аппаратную платформу. Создатели UNIX впервые предложили операционную систему, полностью написанную на языке высокого уровня.

Первым компьютером, которым управляла новая *версия* системы, стал компьютер PDP-11 компании Digital Equipment Corporation (DEC). Несколько десятилетий спустя DEC была переименована в Digital, затем поглощена компанией Compaq, которая, в свою очередь, слилась с Hewlett-Packard.

В начале 70-х годов XX века выпускники Университета в Беркли (University of California-Berkeley), вдохновленные новыми идеями UNIX, начали работу над собственной *версией UNIX*.

В 1977 году вышла первая версия Berkeley Software Distribution - известная университетская *версия UNIX*. Так как эта система была основана на коде AT&T, для ее использования была необходима лицензия AT&T. Разработчики BSD, подстегиваемые ростом цен на лицензии, полностью переписали исходный код системы к 1991 году.

После этого началось развитие UNIX в двух направлениях: одни команды разработчиков взяли за основу версию Беркли, другие - версию System V Release 4.2, которая была разработана AT&T.

**Внутреннее устройство UNIX**

Система состоит из ядра, демонов и утилит. В системе принят ряд соглашений, которые соблюдаются всеми разработчиками программ под UNIX, в частности, поддержка POSIX.

Ядро - это набор системных таблиц и подпрограмм работы с ними. В ядро также входят драйверы устройств - небольшие программы, обеспечивающие системе интерфейс для общения с аппаратурой компьютера - дисками, видеоподсистемой, сетевыми интерфейсами и т.п. Ядро состоит из статической части, которая загружается при старте системы, и модулей. Модули могут динамически загружаться при старте системы или во время работы, при необходимости поддержки той или иной функции. В частности, подсистема поддержки NFS и драйверы внешних устройств оформлены в виде модулей.

Демоны - это серверные приложения, отвечающие за обработку запросов от других (клиентских) программ.

Утилиты - это программы, которые нужны для выполнения разных базовых работ в системе: копирования файлов, управления процессами, восстановления файловой системы и т.п.

В разных вариантах UNIX используются разные файловые системы, причем часто поддерживается несколько разных файловых систем. Например, Linux умеет работать с ext2, ext3 (это ее родные файловые системы), *UFS*, *HPFS*, NTFS, FAT и другими. Solaris, как и некоторые другие ОС UNIX, использует файловую систему типа *UFS*.

UNIX - это многопользовательская многопроцессная система, т.е. в ней может одновременно быть запущено несколько процессов от имени разных *пользователей*. Число процессов, которые можно одновременно запустить, ограничивается размерами таблицы процессов в ядре и другими настройками ядра.

Каждый процесс в UNIX работает в своем собственном адресном пространстве, поэтому сбой в работе одного процесса никак не влияет на работу других. Подсистема виртуальной памяти, являющаяся частью ядра, запрещает процессам обращаться к чужим адресным пространствам.

Одной из функций ядра является планирование процессов, т.е. передача управления от одного процесса к другому. Для этого в ядре есть отдельная подпрограмма, называемая планировщиком задач. Процессы получают управление от планировщика задач в соответствии со своим приоритетом. Планировщик задач через определенное количество микросекунд решает, следует ли передать управление следующему в очереди процессу.

Ядро представляет собой отдельный процесс, выполняющийся с наивысшим приоритетом. В большинстве систем UNIX, включая Solaris, реализована возможность выполнять несколько параллельных подпроцессов внутри процесса. Эти подпроцессы называют потоками (threads). Более подробно этот аспект процессов рассмотрен в лекции 8.

Каждый *пользователь* в UNIX имеет свою собственную учетную запись *пользователя* (account), которая содержит имя *пользователя*, пароль, идентификатор*пользователя* (UID), идентификатор главной *группы пользователя* (GID), описание *пользователя*, его домашний каталог и путь к командному процессору, который следует запустить при интерактивном входе *пользователя* в систему.

*Пользователь*, работающий в UNIX, имеет уникальное имя *пользователя* и уникальный идентификатор. Идентификатор *пользователя* (User ID, UID) - это целое число от 0 до 2147483647. Обычные *пользователи* в Solaris имеют идентификатор в диапазоне от 100 до 60000.

*Пользователю* не надо знать свой идентификатор, потому что он используется только системой, а для входа в систему *пользователь* указывает свое имя (username). Имя *пользователя* может состоять из латинских букв и цифр; длина имени должна быть от 2 до 8 символов. Первый символ должен быть буквой, и по крайней мере один символ должен быть буквой нижнего регистра. Имя *пользователя* назначает системный администратор при создании новой учетной записи. Имя*пользователя* в руководствах часто называют "логин" (login) или "username".

*Пользователи* объединены в *группы*. Каждая *группа* имеет свое имя и уникальный идентификатор (Group ID, GID). В *группе* может быть сколько угодно*пользователей*, и каждый из них может быть участником любого количества *групп*. Однако у каждого *пользователя* есть главная *группа* - она указывается в свойствах любого файла, который создает *пользователь*. Идентификатор *группы* имеет значение от 100 до 60000, если только это не специальная *группа*. Для специальных (предопределенных) *групп* зарезервирован диапазон от 0 до 99.

*Пользователей* объединяют в *группы* для того, чтобы было удобнее администрировать систему.

Например, если право редактировать содержимое web-сайта надо предоставить нескольким ответственным, то достаточно создать специальную *группу*, например, webedit, и включить их всех в эту *группу*.

Предположим, что эти ответственные - *пользователи* ivan, lena и kira. Тогда в файл /etc/group следует добавить новую *группу*:

webedit::101:ivan,lena,kira

Концепция прав доступа в UNIX требует объединять *пользователей* в *группы* всегда, когда нужно предоставить одинаковые права доступа к файлам или каталогам группе людей.

Домашний каталог и путь к командному процессору играют роль только при интерактивном входе *пользователя* в систему. В UNIX каждый *пользователь* может работать с системой как непосредственно (набирая команды ОС на клавиатуре), так и обращаясь через сеть к тем или иным службам, запущенным на компьютере под UNIX.

При входе в систему *пользователь* набирает имя в ответ на приглашение login:. Затем в ответ на password: следует набрать пароль. При вводе пароль никак не отображается, в том числе и звездочками.

Пароль обычно назначается системным администратором при добавлении *пользователя* в систему.

*Пользователь* или системный администратор имеют возможность изменить пароль с помощью команды passwd имя\_пользователя. Системный администратор может изменить пароль любого *пользователя*, все остальные - только свой собственный. Команда passwd без параметров меняет пароль того *пользователя*, который ее запустил. При изменении своего пароля потребуется набрать старый пароль (чтобы подтвердить, что вы - действительно тот, за кого себя выдаете) и затем - новый пароль. Новый пароль потребуется набрать дважды, чтобы система была уверена в вашем выборе.

Пароль может содержать любые символы. Допустимая длина пароля в Solaris составляет от 6 до 8 символов. Символы разных регистров в паролях, как и повсюду в UNIX, считаются различными.

Пароль не должен легко отгадываться. Идеальный пароль в системах UNIX состоит из восьми символов, среди которых есть латинские буквы разных регистров, цифры и знаки препинания. Не следует набирать пароль латинскими буквами, соответствующими русскому слову на клавиатуре, - эти комбинации давно попали в словари взломщиков, так же, как и комбинации последовательных клавиш типа qazwsx.

Так же, как на военном заводе нельзя одолжить чужой пропуск на денек, *пользователям* нельзя делиться с коллегами своим паролем. Строго запрещено писать пароль на бумажке и приклеивать ее к столу, на стену или монитор.

Если под одним именем и паролем работает несколько человек, нельзя выяснить, кто конкретно выполнил в системе действие от этого имени.

Список *пользователей* системы хранится в файлах /etc/passwd (в нем нет зашифрованных паролей, и его можно читать всем) и /etc/shadow (здесь есть зашифрованные пароли и дополнительная информация о параметрах пароля, читать можно только *пользователю* root). Список *групп* хранится в /etc/group.

**Руководство по системе**

В любом варианте UNIX есть главная команда - man. Это команда работы с *руководством по системе*. Подробнее структура *руководства по Solaris* будет рассмотрена в лекции 9. Однако уже сейчас, приступая к работе в системе, вы должны знать, что она вас не бросит в трудную минуту. По каждой программе, установленной в системе, по каждому системному вызову из системных библиотек, по многим понятиям и файлам конфигурации, а также драйверам устройств в стандартном комплекте установки системы есть страница *руководства*.

Для вызова руководства по команде следует набрать

man команда

Для вызова руководства по самой программе man, которая и показывает страницы *руководства*, наберите

man man

Это всегда верный и рекомендуемый, а иногда - единственно возможный способ получения справки о системе для тех, кто хочет разобраться в хитростях работы программ под UNIX или в их настройках.

### Пользовательский интерфейс

Интерфейс пользователя в UNIX может быть текстовым и графическим. Текстовый интерфейс является основным для большинства систем UNIX. Однако в Solaris часто используется графический интерфейс. Мы рассмотрим оба типа интерфейсов. Фактически, все команды, которые выполняются в текстовом интерфейсе, могут быть выполнены в текстовом окне в графическом интерфейсе. Графический интерфейс редко применяется для встроенных систем Solaris, например, там, где Solaris управляет автоматической телефонной станцией или технологическим процессом на заводе. Поэтому системный администратор должен уметь работать, используя любой интерфейс.

### Вход в систему и выход из системы

Для *входа в систему* следует набрать имя пользователя (login) и пароль (password). После загрузки система работает самостоятельно, независимо от того, вошел кто-то в нее или нет. Под "войти в систему" мы подразумеваем начало интерактивного сеанса работы с системой, когда пользователь отдает ей команды, сидя перед клавиатурой и экраном.

После *входа пользователя в систему* для него запускается программа - командный процессор. Эта программа также часто называется интерпретатором команд, оболочкой или "шеллом" (shell). В среде этой программы проходит весь сеанс работы пользователя с системой. При *входе* с графической консоли Solaris вся работа проходит в среде *программы-менеджера окон*. Графический интерфейс пользователя и программы, которые его обеспечивают, в Solaris называется CDE (*Common Desktop Environment*).

Для *выхода из системы* следует дать команду logout или нажать кнопку Exit в центре внизу экрана, если вы работаете в CDE. При работе в командном процессоре в текстовом режиме достаточно нажать Ctrl-D или дать команду exit (это эквивалентно Ctrl-D). Если в процессе работы вы запустили несколько командных процессоров, то команду exit или Ctrl-D придется давать до тех пор, пока, выходя из запущенных командных процессоров, вы не доберетесь до самого первого, который запустился при вашем *входе в систему*.

### Режимы работы системы

UNIX может работать в однопользовательском *режиме* (*single-user* mode) или в многопользовательском *режиме* (multi-*user mode*).

Для обычной работы система загружается в многопользовательском *режиме*. В нем пользователи могут одновременно *входить в систему* локально или через сеть, посылать программам, работающим в системе, запросы различного характера по сети. В этом *режиме* множество пользователей одновременно могут работать в системе. Их число ограничивается размером таблиц ядра. Ограничение числа одновременно работающих пользователей связано не с числом одновременно запущенных командных процессоров или сеансов связи, а с количеством запущенных процессов и потребляемых ими ресурсов. Поэтому в каждый момент времени максимальное число пользователей, имеющих возможность работать с системой, может меняться.

Однопользовательский *режим* используется системным администратором для настройки, резервного копирования или ремонта системы (например, запуска программы fsck, которая выполняет проверку и исправление ошибок в файловых системах дисков, или программы dump, которая выполняет резервное копирование).

Для загрузки системы в однопользовательском *режиме* следует дать команду

boot -s

или

b -s

программе-загрузчику.

Для перехода из многопользовательского *режима* в однопользовательский дайте команду

init s

или

init S

Тот же эффект даст выполнение команды shutdown без параметров.

Для того чтобы перевести систему из однопользовательского *режима* в многопользовательский, нужно перезагрузить компьютер или выйти из командного процессора однопользовательского *режима* ( exit или Ctrl-D, работает в большинстве случаев). Кроме этого, можно запустить программу init с параметром-названием *режима работы*, например

init 3

Cуществует несколько *режимов работы* ( *runlevels* ) операционной системы. *Режим* с номером 1 соответствует однопользовательскому *режиму*, с номером 3 - многопользовательскому. *Режим* номер 0 - это останов (на этот уровень систему переводит команда shutdown ).

Подробнее о *режимах работы системы* говорится в *руководстве по системе*:

man init

 Система делится на две части. Одну часть составляют программы и сервисные функции – это делает операционную среду UNIX такой популярной; данная часть ОС легко доступна пользователям, она включает такие программы, как командный процессор, обмен сообщениями, пакеты обработки текстов и системы обработки исходных текстов программ. Другая часть включает в себя собственно операционную систему, поддерживающую эти программы и функции.

UNIX – традиционно сетевая операционная система.

**1. Функциональные характеристики**

К основным функциям ядра ОС UNIX принято относить следующие:

1. Инициализация системы - функция запуска и раскрутки. Ядро системы обеспечивает средство раскрутки (bootstrap), которое обеспечивает загрузку полного ядра в память компьютера и запускает ядро.
2. Управление процессами и нитями - функция создания, завершения и отслеживания существующих процессов и нитей ("процессов", выполняемых на общей виртуальной памяти). Поскольку ОС UNIX является мультипроцессной операционной системой, ядро обеспечивает разделение между запущенными процессами времени процессора (или процессоров в мультипроцессорных системах) и других ресурсов компьютера для создания внешнего ощущения того, что процессы реально выполняются в параллель.
3. Управление памятью - функция отображения практически неограниченной виртуальной памяти процессов в физическую оперативную память компьютера, которая имеет ограниченные размеры. Соответствующий компонент ядра обеспечивает разделяемое использование одних и тех же областей оперативной памяти несколькими процессами с использованием внешней памяти.
4. Управление файлами - функция, реализующая абстракцию файловой системы, - иерархии каталогов и файлов. Файловые системы ОС UNIX поддерживают несколько типов файлов. Некоторые файлы могут содержать данные в формате ASCII, другие будут соответствовать внешним устройствам. В файловой системе хранятся объектные файлы, выполняемые файлы и т.д. Файлы обычно хранятся на устройствах внешней памяти; доступ к ним обеспечивается средствами ядра. В мире UNIX существует несколько типов организации файловых систем. Современные варианты ОС UNIX одновременно поддерживают большинство типов файловых систем.
5. Коммуникационные средства - функция, обеспечивающая возможности обмена данными между процессами, выполняющимися внутри одного компьютера (IPC - Inter-Process Communications), между процессами, выполняющимися в разных ‑­
6. узлах локальной или глобальной сети передачи данных, а также между процессами и драйверами внешних устройств.
7. Программный интерфейс - функция, обеспечивающая доступ к возможностям ядра со стороны пользовательских процессов на основе механизма системных вызовов, оформленных в виде библиотеки функций.

**2. Особенности архитектуры ОС UNIX**

Архитектура ОС UNIX – многоуровневая (рис.1). На нижнем уровне, непосредственно над оборудованием, работает ядро операционной системы. Функции ядра доступны через интерфейс системных вызовов, образующих второй уровень. На следующем уровне работают командные интерпретаторы, команды и утилиты системного администрирования, коммуникационные драйверы и протоколы, - все то, что обычно относят к системному программному обеспечению. Наконец, внешний уровень образуют прикладные программы пользователя, сетевые и другие коммуникационные службы, СУБД и утилиты.

**3. Способы управления процессами и ресурсами**

Файлы и процессы, являются центральными понятиями модели операционной системы UNIX. Рисунок 1. представляет блок-схему ядра системы, отражающую состав модулей, из которых состоит ядро, и их взаимосвязи друг с другом. Слева изображена файловая подсистема, а справа подсистема управления процессами – две главные компоненты ядра.

Обращения к операционной системе выглядят так же, как обычные вызовы функций в программах на языке Си, и библиотеки устанавливают соответствие между этими вызовами функций и элементарными системными операциями. При этом программы на ассемблере могут обращаться к операционной системе непосредственно, без использования библиотеки системных вызовов. Программы часто обращаются к другим библиотекам, таким как библиотека стандартных подпрограмм ввода-вывода, достигая тем самым более полного использования системных услуг. Для этого во время компиляции библиотеки связываются с программами и частично включаются в программу пользователя. Совокупность обращений к операционной системе разделена на те обращения, которые взаимодействуют с подсистемой управления файлами, и те, которые взаимодействуют с подсистемой управления процессами. Файловая подсистема управляет файлами, размещает записи файлов, управляет свободным пространством, доступом к файлам и поиском данных для пользователей. Процессы взаимодействуют с подсистемой управления файлами, используя при этом совокупность специальных обращений к операционной системе, таких как open (для того, чтобы открыть файл на чтение или запись),close, read, write, stat (запросить атрибуты файла), chown (изменить запись с информацией о владельце файла) и chmod (изменить права доступа к файлу).

Подсистема управления файлами обращается к данным, которые хранятся в файле, используя буферный механизм, управляющий потоком данных между ядром и устройствами внешней памяти. Буферный механизм, взаимодействуя с драйверами устройств ввода-вывода блоками, инициирует передачу данных к ядру и обратно. Драйверы устройств являются такими модулями в составе ядра, которые управляют работой периферийных устройств. Устройства ввода-вывода блоками относятся программы пользователя к типу запоминающих устройств с произвольной выборкой; их драйверы построены таким образом, что все остальные компоненты системы воспринимают эти устройства как запоминающие устройства с произвольной выборкой. Например, драйвер запоминающего устройства на магнитной ленте позволяет ядру системы воспринимать это устройство как запоминающее устройство с произвольной выборкой. Подсистема управления файлами также непосредственно взаимодействует с драйверами устройств "неструктурированного" ввода-вывода, без вмешательства буферного механизма. К устройствам неструктурированного ввода-вывода, иногда именуемым устройствами посимвольного ввода-вывода (текстовыми), относятся устройства, отличные от устройств ввода-вывода блоками.

Подсистема управления процессами отвечает за синхронизацию процессов, взаимодействие процессов, распределение памяти и планирование выполнения процессов. Подсистема управления файлами и подсистема управления процессами взаимодействуют между собой, когда файл загружается в память на выполнение: подсистема управления процессами читает в память исполняемые файлы перед тем, как их выполнить.

Примерами обращений к операционной системе, используемых при управлении процессами, могут служить fork (создание нового процесса), exec (наложение образа программы на выполняемый процесс), exit (завершение выполнения процесса), wait (синхронизация продолжения выполнения основного процесса с моментом выхода из порожденного процесса), brk (управление размером памяти, выделенной процессу) и signal (управление реакцией процесса на возникновение экстраординарных событий.

Модуль распределения памяти контролирует выделение памяти процессам. Если в какой-то момент система испытывает недостаток в физической памяти для запуска всех процессов, ядро пересылает процессы между основной и внешней памятью с тем, чтобы все процессы имели возможность выполняться. Существует два способа управления распределением памяти: выгрузка (подкачка) и замещение страниц. Программу подкачки иногда называют планировщиком, т.к. она "планирует" выделение памяти процессам и оказывает влияние на работу планировщика центрального процессора. «Планировщик» планирует очередность выполнения процессов до тех пор, пока они добровольно не освободят центральный процессор, дождавшись выделения какого-либо ресурса, или пока ядро системы не выгрузит их после того, как их время выполнения превысит заранее определенный квант времени. Планировщик выбирает на выполнение готовый к запуску процесс с наивысшим приоритетом; выполнение предыдущего процесса (приостановленного) будет продолжено тогда, когда его приоритет будет наивысшим среди приоритетов всех готовых к запуску процессов. Существует несколько форм взаимодействия процессов между собой, от асинхронного обмена сигналами о событиях до синхронного обмена сообщениями.

Наконец, аппаратный контроль отвечает за обработку прерываний и за связь с машиной. Такие устройства, как диски и терминалы, могут прерывать работу центрального процессора во время выполнения процесса. При этом ядро системы после обработки прерывания может возобновить выполнение прерванного процесса. Прерывания обрабатываются не самими процессами, а специальными функциями ядра системы, перечисленными в контексте выполняемого процесса.

**4. Условия эксплуатации**

UNIX - многопользовательская операционная система. Пользователи, занимающиеся общими задачами, могут объединяться в группы. Каждый пользователь обязательно принадлежит к одной или нескольким группам. Все команды выполняются от имени определенного пользователя, принадлежащего в момент выполнения к определенной группе.

В многопользовательских системах необходимо обеспечивать защиту объектов (файлов, процессов), принадлежащих одному пользователю, от всех остальных. ОС UNIX предлагает базовые средства защиты и совместного использования файлов на основе отслеживания пользователя и группы, владеющих файлом, трех уровней доступа (для пользователя-владельца, для пользователей группы-владельца, и для всех остальных пользователей) и трех базовых прав доступа к файлам (на чтение, на запись и на выполнение). Базовые средства защиты процессов основаны на отслеживании принадлежности процессов пользователям.

Для отслеживания владельцев процессов и файлов используются числовые идентификаторы. Идентификатор пользователя и группы - целое число (обычно) в диапазоне от 0 до 65535. Присвоение уникального идентификатора пользователя выполняется при заведении системным администратором нового регистрационного имени. Значения идентификатора пользователя и группы - не просто числа, которые идентифицируют пользователя, - они определяют владельцев файлов и процессов. Среди пользователей системы выделяется один пользователь - системный администратор или суперпользователь, обладающий всей полнотой прав на использование и конфигурирование системы. Это пользователь с идентификатором 0 и регистрационным именем root.

При представлении информации человеку удобнее использовать вместо соответствующих идентификаторов символьные имена - регистрационное имя пользователя и имя группы. Соответствие идентификаторов и символьных имен, а также другая информация о пользователях и группах в системе (учетные записи), как и большинство другой информации о конфигурации системы UNIX, по традиции, представлена в виде текстовых файлов. Эти файлы - /etc/passwd, /etc/group и /etc/shadow.

5. Достоинства и недостатки

Широкое распространение UNIX породило проблему несовместимости его многочисленных версий. Для пользователя весьма неприятен тот факт, что пакет, купленный для одной версии UNIX, отказывается работать на другой версии UNIX. Периодически делались и делаются попытки стандартизации UNIX, но они пока имеют ограниченный успех. Процесс сближения различных версий UNIX и их расхождения носит циклический характер. Перед лицом новой угрозы со стороны какой-либо другой операционной системы различные производители UNIX-версий сближают свои продукты, но затем конкурентная борьба вынуждает их делать оригинальные улучшения и версии снова расходятся. В этом процессе есть и положительная сторона - появление новых идей и средств, улучшающих как UNIX, так и многие другие операционные системы, перенявшие у него за долгие годы его существования много полезного. Наибольшее распространение получили две несовместимые линии версий UNIX: линия AT&T - UNIX System V, и линия университета Berkeley-BSD. Многие фирмы на основе этих версий разработали и поддерживают свои версии UNIX: SunOS и Solaris фирмы Sun Microsystems, UX фирмы Hewlett-Packard, XENIX фирмы Microsoft, AIX фирмы IBM, UnixWare фирмы Novell (проданный теперь компании SCO), и список этот можно еще долго продолжать.

Наибольшее влияние на унификацию версий UNIX оказали такие стандарты как SVID фирмы AT&T, POSIX, созданный под эгидой IEEE, и XPG4 консорциума X/Open. В этих стандартах сформулированы требования к интерфейсу между приложениями и ОС, что дает возможность приложениям успешно работать под управлением различных версий UNIX

Одним из основных преимуществ семейства операционных систем типа UNIX и возникшего на их основе подхода к стандартизации интерфейсов операционных систем (важная часть общего подхода открытых систем) является то, что они обеспечивают единую операционную среду на компьютерах с разной архитектурой.