**Содержание**

[Введение 2](#_Toc133136346)

[Раздел 1. Работа с графическим интерфейсом KDE ASP Linux 3](#_Toc133136347)

[Тема 1.1. Вводное занятие 3](#_Toc133136348)

[1.1.1. Цели и задачи практики. Вводный инструктаж. Техника безопасности 3](#_Toc133136349)

[1.1.2. Семейство ОС Unix. ОС Linux 4](#_Toc133136350)

[Тема 1.2. Знакомство с графическим интерфейсом ASP Linux 8](#_Toc133136351)

[1.2.1. Графический интерфейс ASP Linux 8](#_Toc133136352)

[Раздел 2. Работа с терминалом ASP Linux 23](#_Toc133136353)

[Тема 2.1. Работа с файлами и папками ASP Linux 23](#_Toc133136354)

[2.1.1. Команды работы с каталогами 24](#_Toc133136355)

[2.1.4. Команды работы с файлами: управление файлами, генерация имен файлов 29](#_Toc133136357)

[2.1.7. Команды работы с файлами: ссылки, сортировка файлов 32](#_Toc133136358)

[Тема 2.2. Управление процессами 35](#_Toc133136359)

[2.2.1. Управление процессами 35](#_Toc133136360)

[Тема 2.3. Права доступа к файлам и каталогам 42](#_Toc133136361)

[2.3.1. Команды управления правами доступа 42](#_Toc133136362)

[Тема 2.4. Редактирование файлов средствами редактора vi 46](#_Toc133136363)

[2.4.1. Текстовый редактор vi 46](#_Toc133136364)

[Тема 2.5. Изучение сетевых команд shell 49](#_Toc133136365)

[2.5.1. Сетевые команды Shell 49](#_Toc133136366)

[Раздел 3. Сервисные и прикладные программы ASP Linux 54](#_Toc133136367)

[Тема 3.1. Архивация данных 54](#_Toc133136368)

[3.1.1. Архиваторы ASP Linux 54](#_Toc133136369)

[Тема 3.2. Знакомство с текстовым редактором ASP Linux пакта OpenOffice 57](#_Toc133136370)

[3.2.1. Работа с текстовым редактором OpenOffice.org Writer: основные принципы работы 57](#_Toc133136371)

[Тема 3.3. Итоговое зачетное занятие 60](#_Toc133136372)

[Итоговое задание по практике 61](#_Toc133136373)

[Литература 65](#_Toc133136374)

[Технические средства обучения 65](#_Toc133136375)

# Введение

Операционные системы разработаны для огромного количества разнообразных компьютерных систем. Наиболее известными операционными системами для компьютеров являются семейства операционных систем Microsoft Windows и UNIX.

Операционные системы, объединенные общим наименованием UNIX имеют ряд преимуществ перед обычно устанавливаемыми на персональные компьютеры ОС типа Windows. Изучаемая операционная система Linux с одной стороны относится к классу UNIX, а с другой — работает на персональных компьютерах на основе процессоров Intel (хотя сейчас уже существуют ее варианты и для других процессоров).

На сегодняшний момент Linux — самая современная, устойчивая и быстроразвивающаяся система, почти мгновенно вбирающая в себя самые последние технологические новшества.

Цель методических указаний – научить студента работать в среде ASP Linux, привить навыки работы с сервисным и прикладным программным обеспечением этой операционной систем, провести параллелизм в работе и функциональных возможностях изученной среды и ОС Windows, научить проводить аналогии между сходными по назначению программами, что поможет быстрее адаптироваться к новым инструментальным средствам, которые неизбежно придут на смену сегодняшнему программному обеспечению. Предполагается, что студенты уже знакомы с понятием компьютер, работали с MS-DOS или MS Windows. Методические указания содержат теоретические сведения по указанным темам и практические работы, в качестве закрепления полученного теоретического и практического материала по каждой теме приведены контрольные вопросы. В качестве итогового контроля в методических указаниях приведены примеры индивидуальных практических заданий студентов.

# Раздел 1. Работа с графическим интерфейсом KDE ASP Linux

# 

# Тема 1.1. Вводное занятие

## 

## Цели и задачи практики. Вводный инструктаж. Техника

## безопасности

Учебная практика «Операционные системы и среды» предназначена для реализации государственных требований к минимуму содержания и уровню подготовки выпускника по специальности 090108 «***Информационная безопасность»***среднего профессионального образования. Практика является продолжением изучения дисциплины «Операционные системы и среды».

**Цели и задачи практики:**

В результате прохождения практики студент должен приобрести навыки:

* закрепить знания, полученные на теоретических занятиях по предмету «Операционные системы и среды;
* получить практические навыки по работе в среде ASP Linux 9.0 (интерфейс KDE)
* получить навыки работы с текстовыми редакторами, такими как vi, OpenOffice.org Writer;
* получить навыки работы с программами-архиваторами tar, gzip, bzip, bzip2;
* изучить команды работы с файлами и каталогами, команды управления процессами программы Терминал.

**База практики**

Практика проводиться с использованием ПЭВМ типа IBM PC на базе процессора 386 и выше со следующими периферийными устройствами:

* монитор SVGA? Hfphtitybt 800\*600, цветной (16 цветов);
* манипулятор типа «Мышь»;
* клавиатура стандартная;

И с использованием следующих программных средств:

* операционной системы ОС ASP Linux
* KDE 3.3.1.
* Офисный пакет OpenOffice

**Организация практики**

Практика рассчитана на 72 часа и проводится в течение двух недель. В начале практики проводиться инструктаж по соблюдению правил техники безопасности и промсанитарии, разъясняются цели и задачи практики. В течение всей практики проводиться выдача ежедневных практических заданий и проверка их выполнения. По окончании практики проводиться итоговый контроль работы практикантов.

Ежедневные занятия (6 часов) построены следующим образом:

- теоретическая часть: закрепление материала, полученного ранее, освоение нового материала, выдача задания и объяснение его выполнения (2 часа);

- практическая часть с применением ПЭВМ: выполнение постановки задачи, декомпозиция задачи на подзадачи, решение подзадач на ПЭВМ.

Теоретическая часть проводиться в аудитории. Практическая часть с применением ПЭВМ проводиться в лаборатории УВЦ АКВТ.

Для оптимизации усвоения материала группа разбивается на две подгруппы, в каждой из которых работает отдельный преподаватель.

**Организация итогового контроля практикантов**

Итоговый контроль работы студентов проводиться в последний день практики. Студент предоставляет отчет о проделанной работе и выполняет индивидуальное задание, выданное преподавателем.

## 1.1.2. Семейство ОС Unix. ОС Linux

Операционная система — это комплекс программ, который обеспечивает управление аппаратными средствами компьютера, организует работу с файлами (в том числе запуск и управление выполнением программ), реализует взаимодействие с пользователем, т.е. интерпретацию вводимых пользователем команд и вывод результатов обработки этих команд.

Наиболее известными операционными системами для компьютеров являются семейства операционных систем Microsoft Windows и UNIX. Первые ведут свою родословную от операционной системы MS-DOS, которой оснащались первые персональные компьютеры фирмы IBM. Операционная система UNIX была разработана группой сотрудников Bell Labs под руководством Денниса Ричи, Кена Томпсона и Брайана Кернигана (Dennis Ritchie, Ken Thompson, Brian Kernighan) в 1969 году. В конце 70-х годов сотрудники Калифорнийского университете в Беркли внесли ряд усовершенствований в исходные коды UNIX, включая работу с протоколами семейства TCP/IP. Их разработка стала известна под именем BSD ("Berkeley Systems Distribution"). Она распространялась под лицензией, которая позволяла дорабатывать и усовершенствовать продукт, и передавать результат третьим лицам (с исходными кодами или без них) при условии, что будет указано, какая часть кода разработана в Беркли.

Операционные системы типа UNIX, в том числе и BSD, изначально разрабатывались для работы на больших многопользовательских компьютерах — мейнфреймах. Но персональные компьютеры постепенно наращивали мощь своего аппаратного обеспечения, и в наши дни они уже превосходят по возможностям те мейнфреймы, для которых в 70-х годах разрабатывалась ОС UNIX. В начале 90-х годов студент хельсинкского университета Линус Торвальдс (Linus Torvalds) приступил к разработке UNIX-подобной ОС для IBM-совместимых персональных компьютеров.

### *Основные характеристики ОС Linux*

На сегодняшний момент Linux — самая современная, устойчивая и быстроразвивающаяся система, почти мгновенно вбирающая в себя самые последние технологические новшества. Исходные коды Linux распространяются свободно и общедоступны. Linux обладает всеми возможностями, которые присущи современным полнофункциональным операционным системам типа UNIX, такими как:

**1). Реальная многозадачность**

Ядро осуществляет режим разделения времени центрального процессора, поочередно выделяя каждому процессу интервалы времени для выполнения.

**2). Многопользовательский доступ**

Linux  поддерживает возможность одновременной работы многих пользователей. При этом Linux может предоставлять все системные ресурсы пользователям, работающим с хостом через различные удаленные терминалы.

**3). Свопирование оперативной памяти на диск**

Свопирование оперативной памяти на диск позволяет работать при ограниченном объеме физической оперативной памяти; для этого содержимое некоторых частей (страниц) оперативной памяти записываются в выделенную область на жестком диске, которая трактуется как дополнительная оперативная память. Это несколько снижает скорость работы, но позволяет организовать работу программ, требующих большего объема ОЗУ, чем фактически имеется в компьютере.

**4). Страничная организация памяти**

Системная память Linux организована в виде страниц объемом 4K. Если оперативная память полностью исчерпана, ОС будет искать давно не использованные страницы памяти для их перемещения из памяти на жесткий диск. Если какие-либо из этих страниц становятся нужны, Linux восстанавливает их с диска. Некоторые старые Unix-системы и некоторые современные платформы (включая Microsoft Windows) переносят на диск все содержимое ОП, относящееся к неработающему в данный момент приложению, (т. е. ВСЕ страницы памяти, относящиеся к приложению, сохраняются на диске при нехватке памяти) что менее эффективно.

**5). Загрузка выполняемых модулей "по требованию"**

Ядро Linux поддерживает выделение страниц памяти по требованию, при котором только необходимая часть кода исполняемой программы находится в оперативной памяти, а не используемые в данный момент части остаются на диске.

**6). Совместное использование исполняемых программ**

Если необходимо запустить одновременно несколько копий какого-то приложения (либо один пользователь запускает несколько идентичных задач, либо разные пользователи запускают одну и ту же задачу), то в память загружается только одна копия исполняемого кода этого приложения, которая используется всеми одновременно исполняющимися идентичными задачами.

**7). Общие библиотеки**

Библиотеки — наборы процедур, используемых программами для обработки данных. Существует некоторое количество стандартных библиотек, используемых одновременно более чем одним процессом. В старых системах такие библиотеки включались в каждый исполняемый файл, одновременное выполнение которых приводило к непродуктивному использованию памяти. В новых системах (в частности, в Linux), обеспечивается работа с динамически и статически разделяемыми библиотеками, что позволяет сократить размер отдельных приложений.

**8). Динамическое кэширование диска**

Кеширование диска — это использование части оперативной памяти для хранения часто используемых данных с диска, что существенно ускоряет доступ к часто используемым программам и задачам. Пользователи MS-DOS работают со SmartDrive, который резервирует фиксированные области системной памяти для кеширования диска. Linux использует более динамичную систему кеширования: память, зарезервированная под кеш, увеличивается, когда память не используется, и уменьшается, если системе или процессу пользователя требуется больше памяти.

**9). 100%-ное соответствие стандарту POSIX 1003.1. Частичная поддержка возможностей System V и BSD**

POSIX 1003.1 (Portable Operating System Interface — интерфейс мобильной операционной системы) задаeт стандартный интерфейс Unix-систем, который описывается набором процедур языка Си. Сейчас он поддерживается всеми новыми ОС. Microsoft Windows NT также поддерживает POSIX 1003.1. Linux 100%-но соответствует POSIX. Дополнительно поддерживаются некоторые возможности System V и BSD для увеличения совместимости.

**10). System V IPC**

Linux использует технологию IPC (InterProcess Communication) для обмена сообщениями между процессами, использования семафоров и общей памяти.

**11). Возможность запуска исполняемых файлов других ОС**

Linux не является первой в истории операционной системой. Для ранее разработанных ОС, включая DOS, Windows 95, FreeBSD или OS/2, разработана масса различного, в том числе очень полезного и очень неплохого программного обеспечения. Для запуска таких программ под Linux разработаны эмуляторы DOS, Windows 3.1 и Windows 95. Более того, фирмой Vmware разработана система "виртуальных машин", представляющая собой эмулятор компьютера, в котором можно запустить любую операционную систему. Имеются аналогичные разработки и у других фирм. ОС Linux способна также выполнять бинарные файлы других Intel-ориентированных Unix-платформ, соответствующих стандарту iBCS2 (intel Binary Compatibility).

**12). Поддержка различных форматов файловых систем**

Linux поддерживает большое число форматов файловых систем, включая файловые системы DOS и OS/2, а также современные журналируемые файловые системы. При этом и собственная файловая система Linux, которая называется Second Extended File System (ext2fs), позволяет эффективно использовать дисковое пространство.

**13). Сетевые возможности**

Linux можно интегрировать в любую локальную сеть. Поддерживаются все службы Unix, включая Networked File System (NFS), удалeнный доступ (telnet, rlogin), работа в TCP/IP сетях, dial-up-доступ по протоколам SLIP и PPP, и т. д.. Также поддерживается включение Linux-машины как сервера или клиента для другой сети, в частности, работает общее использование (sharing) файлов и удаленная печать в Macintosh, NetWare и Windows.

**14). Работа на разных аппаратных платформах**

Хотя ОС Linux первоначально была разработана для ПК на базе Intel 386/486, сейчас она может работать на всех версиях Intel-овских микропроцессоров, начиная с 386 и кончая многопроцессорными системами на Pentium III (с Pentium IV возникли определенные трудности, но, судя по сообщениям в Интернете, они были вызваны ошибками в реализации процессора). Так же успешно Linux работает на различных клонах Intel от других производителей; в Интернете встречаются сообщения о том, что на процессорах Athlon и Duron от AMD Linux работает даже лучше, чем на Intel. Кроме того, разработаны версии для других типов процессоров — ARM, DEC Alpha, SUN Sparc, M68000 (Atari и Amiga), MIPS, PowerPC и других (отметим, что в настоящей книге рассматривается только вариант для IBM-совместимых компьютеров).

### *Дистрибутивы Linux*

В любой операционной системе можно выделить 4 основных части: ядро, файловую структуру, интерпретатор команд пользователя и утилиты. Ядро — это основная, определяющая часть ОС, которая управляет аппаратными средствами и выполнением программ. Файловая структура — это система хранения файлов на запоминающих устройствах. Интерпретатор команд или оболочка — это программа, организующая взаимодействие пользователя с компьютером. И, наконец, утилиты — это просто отдельные программы, которые, ничем не отличаются от других программ, запускаемых пользователем, за исключением их основного назначения — они выполняют служебные функции.

Слово "Linux" обозначает только ядро. Поэтому правильнее было бы говорить "операционная система, основанная на ядре Linux". Ядро ОС Linux разрабатывается под общим руководством Линуса Торвальдса и распространяется свободно (на основе лицензии GPL), как и огромное количество другого программного обеспечения, утилит и прикладных программ. Одним из следствий свободного распространения ПО для Linux явилось то, что большое число разных фирм и компаний, а также просто независимых групп разработчиков стали выпускать так называемые дистрибутивы Linux.

*Дистрибутив* — это набор программного обеспечения, включающий все 4 основные составные части ОС, т. е. ядро, файловую систему, оболочку и совокупность утилит, а также некоторую совокупность прикладных программ. В мире существует уже более сотни различных дистрибутивов Linux, и все время появляются новые. На сегодняшний день заслуживают внимания только три дистрибутива: Red Hat Linux Cyrillic Edition, Linux Mandrake Russian Edition (и его потомок ALTLinux) и ASPLinux. Для данных дистрибутивов характерно:

* принадлежность к семейству дистрибутивов, строящихся на основе Red Hat Linux (Red Hat — это самый распространенный в мире дистрибутив).
* Эти дистрибутивы изначально русифицированы.
* В каждом из них имеется достаточно отлаженная процедура установки, автоматически распознающая большинство компонент аппаратного обеспечения, что очень облегчает процедуру инсталляции системы.
* Легко устанавливается (добавляется) дополнительное программное обеспечение, так как оно поставляется в RPM-пакетах (технология распространения ПО, аналог программы setup под Windows).
* Эти дистрибутивы поддерживаются сформировавшимися командами разработчиков и постоянно обновляются.

### *Нумерация версий.*

Различают номера версий дистрибутивов и номера версий ядра. Обычно под версией Linux имеют в виду версию ядра (версии ядра развиваются последовательно, а не ветвятся и множатся, как дистрибутивы).

Версии ядра Linux принято обозначать тремя числами, разделенными точкой. Например, дистрибутив Black Cat версии 5.2 был построен на основе ядра версии 2.0.36, т. е. это был Linux версии 2.0.36. Версии ядра с нечетным значением второй цифры обычно не используются для создания дистрибутивов, потому что являются экспериментальными (отладочными). Версии с четной второй цифрой являются (считаются) устойчиво работающими.

# Тема 1.2. Знакомство с графическим интерфейсом ASP Linux

## 1.2.1. Графический интерфейс ASP Linux

Работать с ОС Linux только через интерфейс командной строки довольно трудно. Все необходимые операции в данном случае выполняются путем запуска отдельных команд, перечень которых огромен, и которые надо помнить наизусть.

Широко известной альтернативой интерфейсу командной строки является графический интерфейс. Графический интерфейс обеспечивает дополнительные удобства для пользователя, в частности, возможность запуска программ в отдельных окнах, обозначения программ (или других объектов) в виде маленьких картинок (пиктограмм, значков, иконок), возможность оперировать с объектами с помощью мыши, а также гораздо большую плотность информации на том же пространстве экрана.

Для ОС Linux существуют средства, обеспечивающие дружественный к пользователю графический интерфейс, который на первый взгляд очень похож на широко известный графический интерфейс Microsoft Windows, но его внутреннее устройство принципиально отличается.

### 1.2.1.1. XFree86 и его составные части

Графический интерфейс в Linux строится на основе стандарта X Window System или просто "X", первоначальный вариант которого был разработан в 1987 году в Массачусетском технологическом институте. Начиная со второй версии этот стандарт поддерживался консорциумом X, созданным в январе 1988 г. с целью унификации графического интерфейса для ОС UNIX. С 1997 года консорциум X преобразован в X Open Group. В настоящее время действует версия 11 выпуск 6 стандарта на графическую подсистему для UNIX-систем, которая кратко обозначается как X11R6.

Свободно распространяемая реализация стандарта X11R6 для UNIX-систем с процессорами 80386/80486/Pentium (в том числе для ОС Linux) была создана группой программистов, которую вначале возглавлял Дэвид Вексельблат (David Wexelblat). Эта реализация известна как XFree86, и может использоваться не только в Linux, но и в System V/386, 386BSD, FreeBSD и других версиях UNIX для систем на базе процессоров Intel x86. В настоящее время выпущена уже 4-ая версия XFree86, однако, и 3-я версия еще широко используется и входит в состав основных дистрибутивов Linux.

Система X Window построена на основе модели "клиент/сервер". X сервер работает на компьютере пользователя (а не на каком-то удаленном "сервере") и обеспечивает вывод изображения на экран монитора. X-сервер работает непосредственно с "железом": видеосистемой, устройствами ввода и динамиком. Эта программа захватывает оборудование и предоставляет его возможности другим программам как ресурсы по особому протоколу, который называется X-протокол, или протокол сетевой связи (X Network Protocol). Специализированный компьютер, на котором исполняется исключительно X-сервер, называется (аппаратным) *X-терминалом*.

X-сервер не формирует изображение, он только "доставляет" графику видео-драйверу. Если запустить только X-сервер, вы увидите просто серый экран с крестиком курсора посредине. С помощью мыши крестик можно перемещать по экрану. На нажатие кнопок мыши и клавиш реакции не следует — сервер готов передавать эти сигналы своим клиентам, а клиенты пока не запущены. Хотя некоторые комбинации клавиш X перехватывает и обрабатывает, к ним относятся <Ctrl>+<Alt>+<Backspace> — завершение работы сервера, <Ctrl>+<Alt>+<+> и <Ctrl>+<Alt>+<-> — "горячее" переключение доступных видеорежимов, и <Control>+<Alt>+<F#> — переключение в другую виртуальную консоль.

Таким образом одного X-сервера недостаточно для работы, необходим запуск менеджера окон и хотя бы одной программы-клиента, которая будет формировать изображение. В роли "клиентов" X-сервера выступают приложения, работающие с X Window, например графический редактор GIMP, текстовый редактор Corel WordPerfect, эмулятор терминала xterm и другие.

Между клиентами и сервером стоят еще два очень важных компонента графического интерфейса: библиотека графических функций X-lib и менеджер окон (рис. 1.2.1.1). X-Lib содержит графические функции, которые обеспечивают выполнение низкоуровневых операций с графическими образами. Менеджер окон вызывает функции из X-Lib для управления дисплеем и выполнения любых преобразований изображений в окнах.

При запуске X-приложение передает управление менеджеру окон, который обеспечивает выполнение всех операций с окнами: прорисовку рамок, меню, иконок, полос прокрутки и других элементов окна, а также предоставляет возможность изменять вид и положение окна в процессе работы в соответствии с потребностями пользователя. Менеджер окон также вызывает соответствующие функции для программ-клиентов в тех случаях, когда пользователь взаимодействует с приложением с помощью клавиатуры и мыши.

Расширенные графические среды типа Motif, CDE, KDE, GNOME, GNUStep и т. д. не замещают перечисленные выше компоненты системы X Window, а расширяют и дополняют их. KDE, например, добавляет библиотеку графических функций Qt в дополнение к X-Lib. Motif имеет собственный набор графических функций. GNOME использует библиотеку GTk+, которая составляет основу GIMP. Кроме того, в GNOME используется также CORBA (The Common Object Request Broker Architecture — универсальная архитектура посредничества при запросе объектов) и библиотека Imlib для дальнейшего расширения возможностей графической подсистемы.

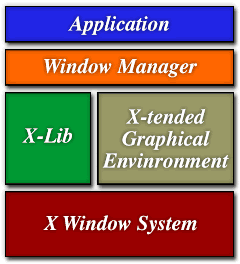


Рис. 1.2.1.1. Архитектура графической системы в Linux

Поскольку клиент и сервер являются отдельными процессами, они могут работать на разных компьютерах, взаимодействовуя по сети. Приложения можно запустить, например, на мейнфрейме, а картинка будет выводиться на экран персонального компьютера. Эта очень мощная особенность системы X Window является одним из основных отличий ее от MS Windows.

В Linux (и вообще в UNIX) нет жесткого деления между графическими и текстовыми программами, как в MS Windows или OS/2. С точки зрения системы нет разницы между программой, работающей в графике, и обычной. Программы для графического режима запускают как обычные программы, т. е. из командной строки, из Midnight Comander'а и т.п. Единственным необходимым условием для их работы является то, что должен работать X-сервер. При необходимости программа сама обращается к X-серверу.

На рис. 1.2.1.1 схематично представлена архитектура графической системы в Linux. Еще одним важным ресурсом графической подсистемы являются шрифты. Оперировать со шрифтами может как непосредственно X-сервер, так и специальная программа, которая называется сервер шрифтов.

Для каждого из типовых компонентов графической системы существует множество конкретных реализаций. В состав пакета XFree86 версии 3 входят несколько различных серверов, причем выбор конкретного сервера зависит от того, какие у вас видеоплата и монитор. Например, сервер XF86\_Mono — это сервер для монохромных видео-режимов; XF86\_S3 — сервер для карт, основанных на S3; XF86\_S3V — сервер для S3 ViRGE и ViRGE/VX; XF86\_SVGA — сервер для карт, работающих в режимах Super-VGA.

В четвертой версии XFree86 уже один X-сервер для большинства видеоадаптеров, называется он XFree86 и располагается в каталоге /usr/X11R6/bin/. Обычно на него делается ссылка с именем X, так что запустить X-сервер можно просто введя в командной строке букву X.

Различных менеджеров окон тоже существует очень много, например:

* **fvwm** — Free Virtual Window Manager.
* **fvwm2** — улучшенный вариант fvwm, позволяющий, в частности, использовать различные темы рабочего стола и динамические меню.
* **fvwm95** — менеджер окон с графическим интерфейсом в стиле Windows 95.
* **IceWM** — это оконный менеджер, который может эмулировать различные стандарты оконных систем (в частности, OS/2 Workplace Shell), имеет, по некоторым отзывам, довольно маленький обьем и быстро работает.
* **Enlightenment** — один из самых развитых менеджеров окон.

Разные менеджеры окон могут обеспечивать различный вид окон за счет использования различных рамок и оконных меню. Но все они используют одну и ту же базовую графическую утилиту X Window — X сервер.

Графический интерфейс в Linux строится по модульному принципу. Возможна самостоятельная сборка графического интерфейса из отдельных компонент. В настоящее время разработаны (и включены практически во все дистрибутивы) так называемые интегрированные графические среды. Наиболее известными представителями таких сред являются KDE и GNOME. При создании таких интегрированных графических сред различные и независимые разработчики используют единый подход, единую идеологию создания и оформления программных продуктов. В результате получается функционально полный пакет программ, в рамках которого решены практически все задачи управления компьютером и программной средой.

Основу интегрированной графической среды KDE (K Desktop Enviroment) образует расширенная библиотека графических функций Qt фирмы Troll Tech. С использованием этой библиотеки построены собственный оконный менеджер kwm, файл-менеджер kfm, центр управления KDE (аналог панели управления Windows) и множество других компонентов, вплоть до собственного офисного пакета KOffice.

В состав версии 2.1 KDE включен файловый менеджер Konqueror, который предоставляет уникальные возможности доступа к файлам. Кроме того, что он позволяет просматривать файлы большинства известных форматов на локальных дисках, он является и интернет-браузером, по своим возможностям вполне сравнимым с Internet Explorer или Netscape Navigator.

Другой графической средой того же класса, что и KDE, является пакет GNOME (GNU Network Object Model Environment), который разрабатывается в рамках проекта GNU, а значит, относится к классу свободно распространяемого ПО (KDE до недавнего времени не полностью соответствовал этому понятию, потому что библиотека Qt распространялась не на условиях GPL; хотя сейчас ситуация изменилась и KDE тоже является свободно распространяемым). GNOME строится на основе библиотеки графических функций GTK+.

Существуют и другие разработки интегрированных графических сред, которые, однако, пока не достигли той степени развития, как KDE или GNOME, например, Xfce, dfm и другие.

До недавнего времени установка и настройка графического интерфейса Linux представляла собой довольно непростую задачу. Однако программы инсталяции последних версий дистрибутивов Linux (например, русифицированой версии Red Hat Linux 7.1) уже вполне успешно с ней справляются, автоматически определяя тип монитора и видеоадаптера и выбирая подходящее разрешение экрана.

### 1.2.1.2. Графическая среда KDE

KDE — это интегрированная графическая оболочка для Linux (и других версий UNIX), которая в настоящее время включает в себя более 100 графических приложений и поддерживает более 40 различных языков. Она разрабатывается в рамках движения Open Source, т. е. распространяется с открытыми исходными кодами. KDE позволяет совместить современную функциональность, удобство использования и отличный дизайн с технологическими преимуществами операционной системы класса UNIX.

### *Внешний вид*

Внешним видом экрана после запуска KDE представлен на рис. 1.2.1.2.1

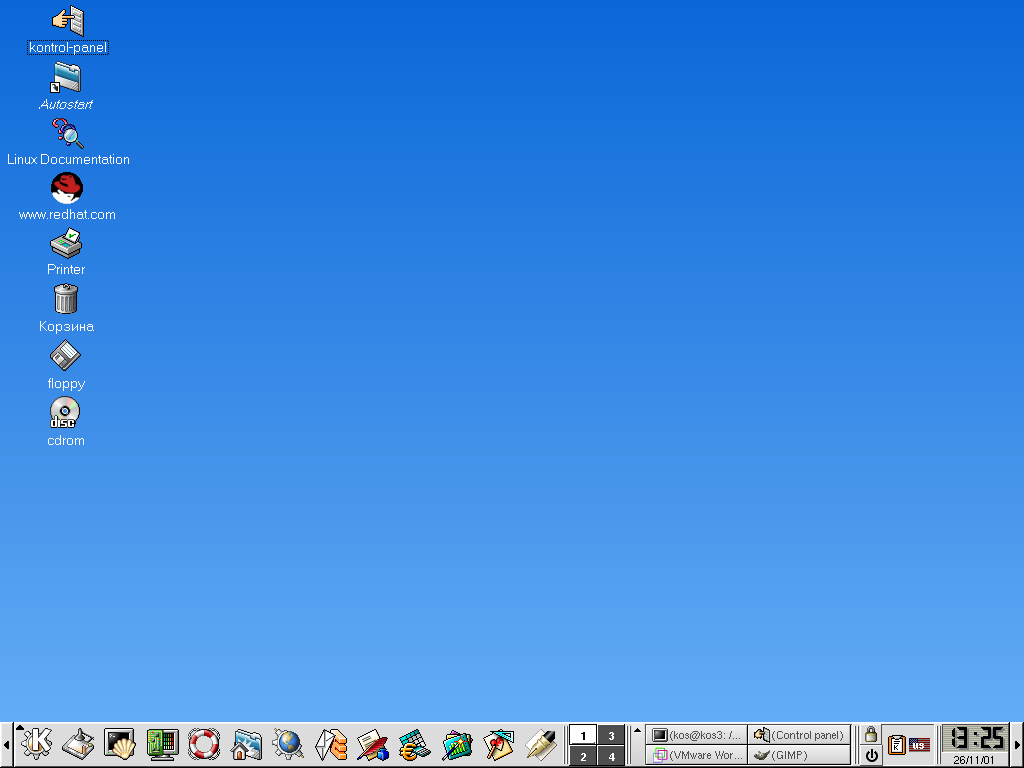


Рис. 1.2.1.2.1. Внешний вид графической среды KDE

Экран можно условно поделить на две части.

Панель в нижней части экрана служит для запуска приложений и переключения между рабочими столами. На паанели расположен значок с изображением буквы "К". Этот значок (аналог кнопки Start в Windows) служит для вызова иерархического меню, через которое можно запустить любое приложение из числа входящих в состав KDE, даже если значок приложения отсутствует на панели.

Вся оставшаяся поверхность экрана, на которой располагается некоторое число значков ("иконок"), которые тоже могут использоваться для запуска соответствующих им приложений. — это *рабочий стол* (Desktop)

KDE поддерживает несколько рабочих столов, переключаться между которыми можно с помощью пронумерованных клавиш, расположенных на панели (на рис. 1.2.1.2.1 изображены четыре такие клавиши, хотя это число можно изменить).

Правее переключателя рабочих столов располагается поле, которое называется панелью задач (taskbar). Оно служит для отображения перечня запущенных в данный момент приложений и переключения между ними. Каждому запущенному приложению соответствует кнопка, щелчок по которой переводит данное приложение в активное состояние.

Для тех, кто привык работать в Windows, может показаться непривычным то, что для запуска приложения достаточно щелкнуть по значку ("иконке") только один раз. Но к этому быстро привыкаешь, а и при желании можно настроить оболочку на два щелчка.

В правом конце панели задач находятся часы и небольшая вертикально вытянутая кнопка с треугольником-стрелкой. Такая же кнопка имеется и в левом конце панели. Щелчок по любой из этих кнопок приводит к тому, что панель сворачивается, как бы убегая за границу соответствующей стороны экрана. Видимой остается только такая же кнопка с треугольником. Щелчком по этой кнопке можно вернуть панель на экран.

Если вы подведете указатель мыши к любому значку или кнопке на панели задач и выждете некоторое время, появится подсказка, поясняющая назначение значка или название соответствующей задачи. А если щелкнуть правой кнопкой мыши по любому элементу на экране, в том числе и по пустому полю, появляется меню, в котором можно выбрать одно из действий, применимых к данному элементу. В частности, щелчок по пустому полю приводит к появлению меню настроек рабочего стола.

Панель можно настраивать по своему усмотрению. Но описанием способов настройки мы пока заниматься не будем. Давайте вначале посмотрим на главное меню KDE.

### *Главное меню KDE*

Доступ к главному меню KDE получаем, щелкнув по значку с буквой "K" (рис. 1.2.1.2.2).

Некоторые элементы меню имеют треугольник после названия. Это говорит о том, что данный элемент вызывает вложенное меню следующего уровня.

В меню RDE имеются аналоги команд, имеющихся в главном меню Windows: Поиск файла, Помощь, Запустить программу…

Команды Каталог и Домашний каталог служат для быстрого перехода в нужный каталог и просмотра его содержимого с помощью файлового менеджера Konqueror.

В верхней части главного меню появляется отделенный горизонтальной чертой список часто запускаемых или недавно вызывавшихся команд.

Смысл подавляющего числа команд меню не требует особых пояснений — они вызывают соответствующее приложение. Рассмотрим такие приложения как Центр управления и Настройки панели.

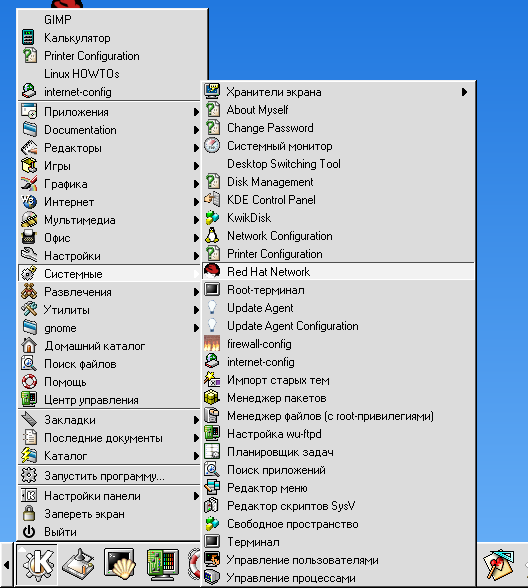


Рис 1.2.1.2.2. Главное меню KDE

### *Центр управления KDE*

Типичный вид окна при работе с Центром управления KDE изображен на рис. 1.2.1.2.3: слева расположено меню, а справа — поле вкладок, каждая из которых обычно служит для настройки какого-то конкретного элемента графической среды.

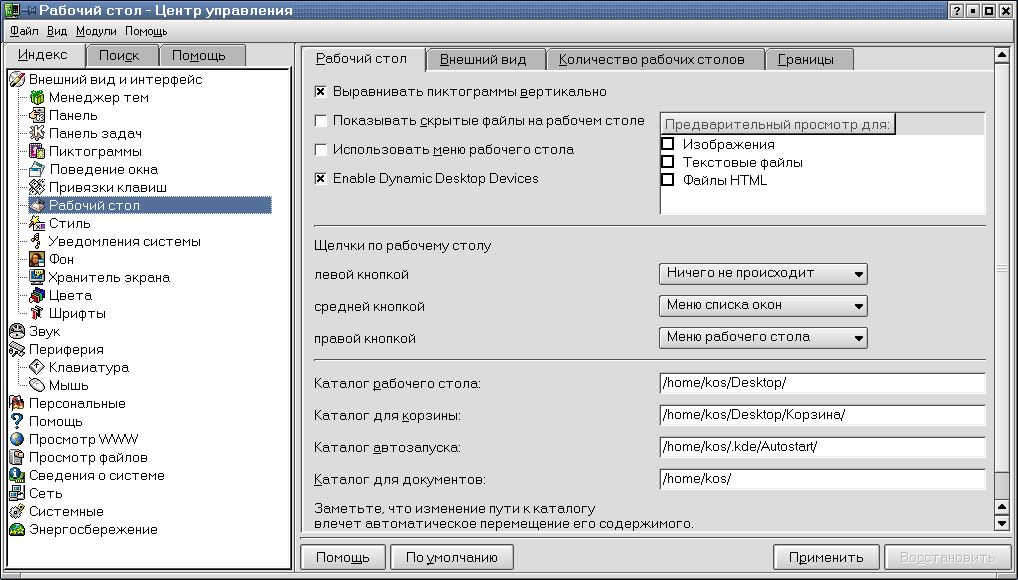


Рис. 1.2.1.2.3. Один из экранов Центра управления KDE (щелкните по рисунку для просмотра)

Настройка фона или темы рабочего стола выполняется с помощью команд Фон и Менеджер тем меню Внешний вид и интерфейс главного меню Центра управления. На приведенных выше рисунках вы можете видеть результат. Одновременно можно выбрать основные цвета (команда Цвета) и стиль оформления графических элементов (команда Стиль).

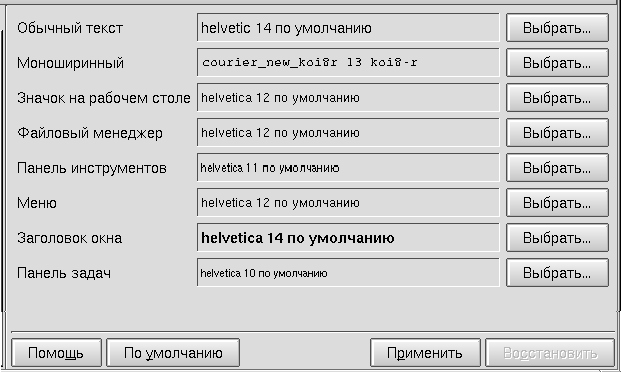


Рис. 1.2.1.2.4. Задание шрифтов для разных элементов экрана

. На рис. 1.2.1.2.4 вы видите правую панель Центра управления, соответствующую команде Шрифты, которая позволяет выбрать шрифты, которыми будут выводиться различные надписи

Для изменения какого-либо шрифта надо щелкнуть по экранной кнопке Выбрать. Появится окно, изображенное на рис. 1.2.1.2.5, в котором и осуществляется такой выбор. После того, как выбор шрифта произведен, нужно щелкнуть по кнопке Ok. Для того, чтобы изменения вступили в силу, надо нажать кнопку Применить на панели Центра управления (рис. 1.2.1.2.4). Это касается не только шрифтов, но и любых других изменений, производимых с помощью Центра управления.

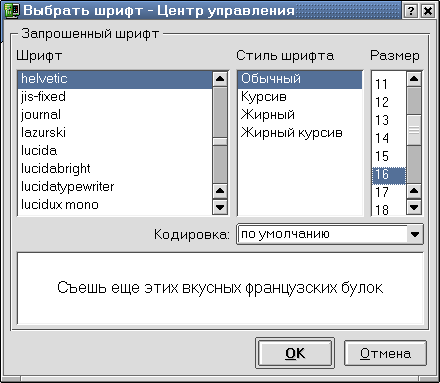


Рис. 1.2.1.2.5. Окно выбора шрифтов

Вид и содержание главного меню KDE тоже можно настраивать (см. рис. 1.2.1.2.6.) Как видно из рис. 1.2.1.2.6. состав и содержание элементов меню здесь изменить невозможно. Для этого можно вызвать отдельную программу — Редактор меню KDE, что делается через команду Настройка панели | Редактор меню того же главного меню.

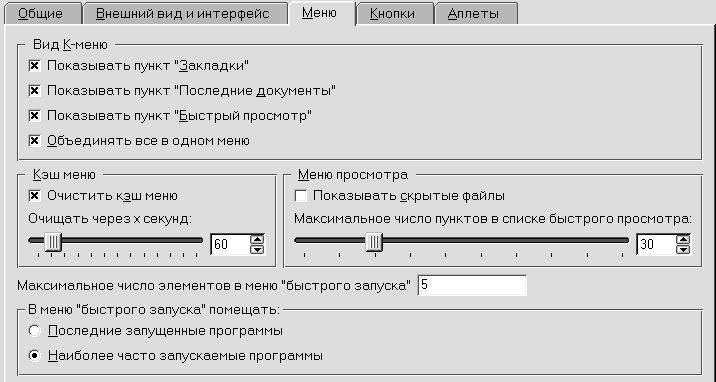


Рис. 1.2.1.2.6. Настройка главного меню

Окно программы Редактор меню KDE представлено на рис. 1.2.1.2.7. Здесь можно как создать новую команду меню, так и новое подменю, указать название приложения и имя запускаемого файла, а также назначить запуск программы от имени другого пользователя.

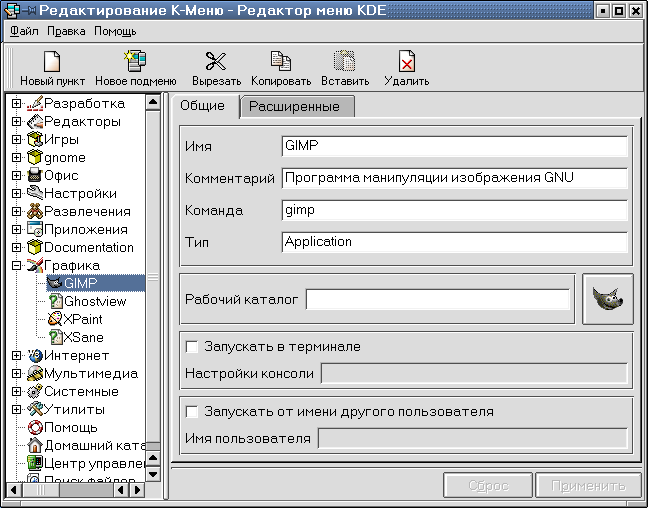


Рис. 1.2.1.2.7. Редактор меню KDE

На вкладке Расширенные можно назначить "горячую" клавишу, по которой можно будет вызывать приложение, не прибегая к помощи меню и мыши.

### *Настройка панели и значков на рабочем столе*

Для настройки панели главного меню можно щелкнуть правой клавишей мыши по свободному полю на панели, и появится выпадающее меню, изображенное на рис. 1.2.1.2.8, в котором тоже имеется команда Настройки, и которая по содержанию идентична команде Настройка панели главного меню. Если вызвать эту команду, появится окно, изображенное на рис. 1.2.1.2.9.

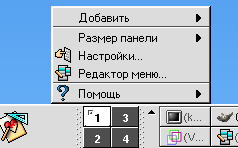


Рис. 1.2.1.2.8. Выпадающее меню для Панели

С помощью вкладки «Общие» можно переместить панель к любой границе экрана, изменить ее размер, задать скорость движения при автоскрытии и определить терминальное приложение. На вкладке «Внешний вид и интерфейс»» задаются еще несколько аналогичных параметров.

Первая команда контекстного меню, изображенного на рис. 1.2.1.2.8, «Добавить». Всплывающее подменю позволяет добавить можно как отдельный элемент (т. е. кнопку или иконку) на панель задач, так и четыре вида дополнительных панелей (или четыре вида расширений основной панели). Одна из таких панелей - дополнительная панель задач (на рис. 1.2.1.2.10 она расположена над основной панелью).

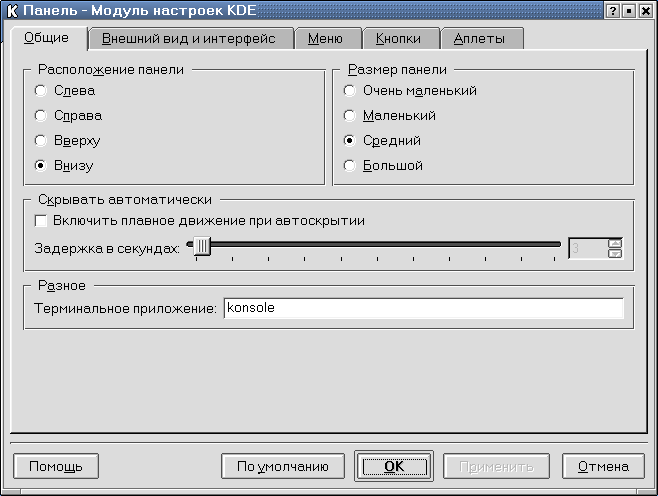


Рис. 1.2.1.2.9. Окно настроек панели

Щелчок на сером прямоугольнике в левом конце этой дополнительной панели дает возможность удалить дополнительную панель или добавить на нее что-то (кнопку мыши надо удерживать). С помощью таких же серых прямоугольников можно получить доступ к меню других элементов главной панели.

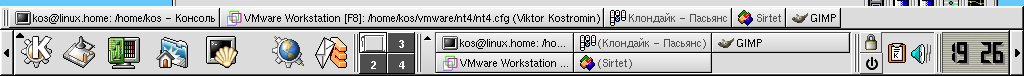


Рис. 1.2.1.2.10. Дополнительная панель задач (щелкните по рисунку для просмотра)

Меню управления кнопками на панели появляются после щелчка правой кнопкой мыши по самой кнопке (рис. 1.2.1.2.11).

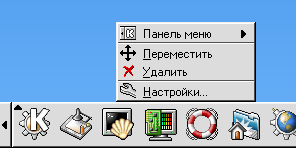


Рис. 1.2.1.2.11. Выпадающее меню для кнопки на Панели

Первый элемент в этом меню (Панель меню) вызывает меню настроек панели, следующие два служат для перемещения или удаления данной конкретной кнопки, а последний вызывает окно, изображенное на рис. 1.2.1.2.12. В этом окне можно изменить некоторые параметры той кнопки, щелчок по которой вызвал появление меню. Изменить можно рисунок на кнопке, вызываемую по ней программу, права, с которыми программа запускается, и список типов файлов, ассоциированных с данным приложением.

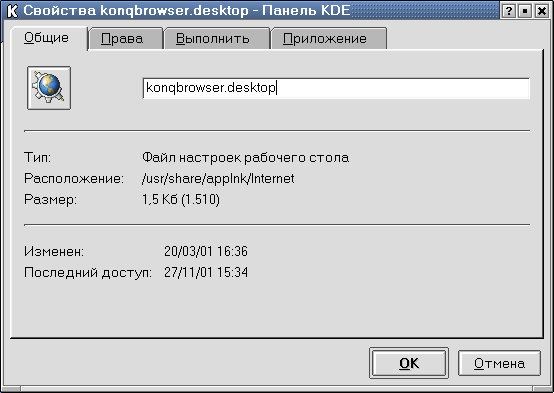


Рис. 1.2.1.2.12. Свойства ссылки

Значки ("иконки") на рабочем столе тоже могут использоваться для запуска приложений или быстрого доступа к некоторым файлам или каталогам. Создать их можно щелкнув по пустому пространству рабочего стола, и создав новую ссылку на приложение, файл или устройство, как это показано на рис. 1.2.1.2.13.

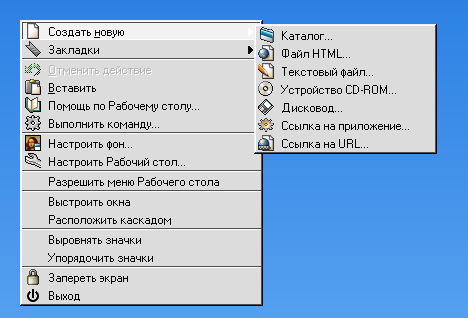


Рис. 1.2.1.2.13. Создание новой ссылки на рабочем столе.

Кроме этой программы KDE включает в себя еще массу разнообразных приложений, общим числом более сотни.

#### Практическая работа №1(1) Изучение графического интерфейса ASP Linux.

##### Цель работы:

1. Познакомиться с пользовательским интерфейсом ASP Linux.
2. Научиться работать со справочной системой ASP Linux

##### Порядок выполнения работы

1. Зарегистрируйтесь в ОС Linux. Какой графический интерфейс используется в ОС Linux?
2. Опишите составляющие части графического интерфейса и структуру главного меню KDE.
3. Вызовите Центр управления KDE. Какие возможности предоставляет Центр управления?
4. Измените фон рабочего стола, стиль оформления графических элементов, шрифты, для отображаемой информации.
5. Измените структуру главного меню: удалите пункт «Последние документы», включите опцию отображения скрытых файлов. Какие ещё изменения можно внести?
6. Назначьте для программы «Терминал» горячие клавиши.
7. Запустите утилиту настройки Рабочего стола, для этого выберите пункт меню «Настроить Рабочий стол…». Какие возможности предоставляет утилита? Измените настройки Рабочего стола. Какие изменения Вы внесли?
8. Откройте браузер Konqueor. Перейдите в корневой каталог диска. Какие файлы и папки расположены в корневом каталоге?
9. Вставьте в дисковод дискету и выполните операцию монтирования дискеты. Для выполнения операции монтирования модно воспользоваться контекстным меню либо утилитой монтирования для пользователей «KDE>Система>Управление дисками».
10. Просмотрите содержимое дискеты, для этого в браузере Konqueor перейдите в папку /mnt и выберите соответствующую подпапку fd01.
11. Выполните операцию размонтирование дискеты.
12. Отформатируйте вставленную дискету (дискета должна находиться в размонтированном состоянии), воспользовавшись утилитой Kfloppy. При форматировании задайте размер дискеты 1,2 Мб и файловую систему ext.
13. Отформатируйте вставленную дискету на 1,2 Мб, в качестве файловой системы задайте систему DOS, укажите начальную метку диска.
14. Просмотрите содержимое отформатированной дискеты.
15. Выполните быстрое форматирование дискеты. Чем отличается быстрое форматирование от полного?
16. Создайте на рабочем столе ссылку на Ваш личный каталог. Просмотрите свойства ссылки. Сравните свойства созданного ярлыка со свойствами элементов рабочего стола.
17. Завершите сеанс работы с системой

#### Практическая работа №1(2) Работа с поисковой системой

##### Цель работы:

1. Научиться осуществлять поиск файлов и папок в ОС Linux.
2. Научиться задавать маску файла для поиска.

##### Порядок выполнения работы

1. Зарегистрируйтесь в ОС Linux.
2. Запустите утилиту «Поиск файлов» в меню KDE.
3. Осуществите поиск в корневом каталоге всех файлов с расширениями txt.
4. Найдите на диске все графические файлы. Где они размещены?

Для поиска надо задать маску файлов. В маске разрешено использование символов «\*» (заменяет любую последовательность символов) и «?» (заменяет один символ). Кроме маски обязательным является указание каталога, в котором будет произведен поиск файлов. Для поиска не только в указанном каталоге, но и во всех его подкаталогах, следует поставить галочку «Включая подкаталоги». Что бы произвести поиск во всем файловом пространстве надо указать «file:/» в качестве значения «Искать в…» и поставить галочку поиска в подкаталогах. В этом случае поиск пройдет не только по жестким дискам, но и по всем сменным.

1. Найдите файлы, которые были созданы на прошлой неделе.
2. Найдите файлы созданные в промежутке между текущей и прошедшей неделей.
3. Осуществите поиск файла на диске с учетом регистра букв.
4. Найдите файлы, размер которых не превышает 10 Кб, а так же файлы, размер которых составляет 1 Мб. Откройте найденный файлы для просмотра.
5. Завершите сеанс работы.

### *Контрольные вопросы:*

1. Дайте определение понятию операционная система. Какие операционные системы Вы знаете?
2. Объясните понятия многозадачности и многопользовательскости.
3. Какие характеристики операционной системы (ОС) Linux вы знаете?
4. Что такое дистрибутив ОС? Какие дистрибутивы Linux Вы знаете?
5. Что означает понятие графический интерфейс? Какой графический интерфейс используется в Linux.
6. Какие графические среды Linux Вы знаете?
7. Опишите внешний вид графической среды KDE. В чем отличие графической оболочки OC Windows от среды KDE?
8. Опишите состав и структуру главного меню KDE. Какие утилиты, графические и текстовые программы входят в поставку КDE?
9. Что означает понятие ярлык? Как создать ярлык на рабочем столе?
10. Как осуществлять добавление и удаление панелей задач на рабочем столе?
11. Опишите способы создания папки?
12. Какие стандартные редакторы, позволяют создать текстовый документ?
13. Какова максимальная длина имени файла или папки в Linux?
14. Расскажите о способах открытия документов?
15. Как отобразить панель инструментов в окне программы Konqueor?
16. Опишите пиктограммы на панели инструментов программы Konqueor?
17. Какая информация отображается в строке состояния программы Konqueor?
18. Как определить размер папки в килобайтах?
19. С помощью какой клавиши можно перейти в «Родительскую папку»?
20. Как можно вызвать окно свойств объекта?
21. Способы упорядочения файлов в окне папки программы Konqueor?
22. Способы расположения и изменения размера значков файлов в окне папки?
23. Как установить автоматическое выравнивание значков файлов в программе Konqueor?
24. Как можно отменить последнюю выполненную команду?
25. Как переименовать файл или папку?
26. Как установить атрибуты файла?
27. Как скопировать файл из одной папки в другую?
28. Как переместить файл из одной папки в другую?
29. Как выделить в папке группу файлов, расположенных не подряд?
30. Как выделить сразу все объекты в папке?
31. Как снять выделение с одного или нескольких файлов, не отменяя выделение остальных?
32. Для чего используется команда Обновить в программе Konqueor?
33. Опишите способы удаления файлов и папок?
34. Опишите параметры настройки корзины?
35. Как найти файл на диске по его имени?
36. Объясните, для чего при поиске файла используются символы звездочка «\*» и знак вопроса «?»?
37. Как найти файлы по дате создания?
38. Как восстановить удаленный файл?
39. В каком случае восстановление удаленных файлов невозможно?
40. В чем отличие ярлыка от пиктограммы?
41. Опишите способы создания ярлыка?
42. Как изменить значок ярлыка?

# Раздел 2. Работа с терминалом ASP Linux

# 

# Тема 2.1. Работа с файлами и папками ASP Linux

Работа с системой Linux в текстовом режиме, т.е. с консоли или терминала, позволяет быстрее и удобнее выполнить многие действия в системе, по сравнению с графическим режимом. Даже находясь в графическом режиме, часто быстрее открыть окно эмулятора терминала, и выполнять необходимые действия в нем. Поскольку Linux — это многозадачная ОС, уже в текстовом режиме обеспечивается возможность работы в нескольких окнах. А для редактирования простого текста или html-файла вовсе не обязательно запускать достаточно тяжеловесную и медленную графическую оболочку.

Взаимодействие с пользователем организует специальная программа. Существует два вида таких программ — оболочка, или shell, для работы в текстовом режиме (интерфейс командной строки) и графический интерфейс пользователя GUI (Graphical User Interface), организующий взаимодействие с пользователем в графическом режиме.

### *Выполнение команд в shell*

Одна из основных функций оболочки состоит в том, чтобы организовать исполнение команд пользователя, вводимых им в командной строке.

Формат команд в ОС LINUX следующий:

**имя команды  [аргументы] [параметры] [метасимволы].**

*Имя команды* может содержать любое допустимое имя файла; *аргументы* - одна или несколько букв со знаком минус (-); *параметры* - передаваемые значения для обработки; *метасимволы* интерпретируются как специальные операции.

Оболочка должна найти код команды, загрузить его в память, передать команде аргументы, заданные в командной строке, а после завершения выполнения соответствующего процесса передать каким-то образом пользователю или другому процессу результаты выполнения данной команды.

На первом этапе осуществляется поиск кода команды. Команды бывают *встроенные* (те, код которых включен в код самой оболочки) и *внешние* (код которых расположен в отдельном файле на диске). Для поиска внешней команды пользователь, должен указать оболочке полный путь до соответствующего файла. Однако оболочка может находить внешние команды в каталогах, которые перечислены в специально заданных "путях поиска". Только если она не находит нужных файлов в таких каталогах, она решает, что пользователь ошибся при вводе имени команды.

Для организации задания команд в командной строке оболочка предоставляет пользователю два специальных оператора: ; и &.

### 1) Оператор ;

Хотя чаще всего пользователь задает команды в командной строке по одной, имеется возможность задать в одной строке несколько команд, которые будут выполнены последовательно, одна за другой. Для этого используется специальный символ -оператор ;. Если не поставить этот разделитель команд, то последующая команда может быть воспринята как аргумент предыдущей. Таким образом, если написать в командной строке что-то вроде:

[user]$ command1 ; command2

то оболочка вначале запустит на выполнение команду command1, дождется, пока ее выполнение завершится, после чего запустит command2, дождется ее завершения, после чего снова выведет приглашение командной строки, ожидая следующих действий пользователя.

### 2) Оператор *&*

Оператор & используется для того, чтобы организовать исполнение команд в фоновом режиме. Если поставить значок & после команды, то оболочка вернет управление пользователю сразу после запуска команды, не дожидаясь, пока выполнение команды завершится. Например, если задать в командной строке "command1 & command2 &", то оболочка запустит команду command1, сразу же затем команду command2, и затем немедленно вернет управление пользователю.

## 

## 2.1.1. Команды работы с каталогами

### *Понятие файла*

*Файл* - единица информации, поименованный набор логически связанных данных (таблицы, программа, текст и пр.) Обращение к файлу по его уникальному имени (собственное имя файла), структура имени:

имя-файла[.тип]

Возможные расширения (типы) для текстовых файлов: txt, doc; графических файлов: bmp, jpg и т.д. В UNIX расширения имен, определяющие тип файла, не обязательны. Все файлы имеют собственные описатели (дескрипторы) и хранятся во внешней памяти. Физические адреса файлов на внешней памяти концентрируются в специальных файлах - каталогах.

### *Структура системы каталогов Linux*

Строится в иерархической системе внешней памяти на блочных устройствах.

/

mail

bin

dev

tmp

etc

home

TTY

ksh

date

passwd

user

user n

lev

ira

petr

Lev1

f1

filira

fil2

1

2

3

4

5

vmunix

Рис. 2.1.1. Структура системных каталогов Linux

На рисунке 2.1.1. представлены следующие группы каталогов:

1 – коренной каталог системы (строится при инсталляции ОС)

2 – системные каталоги (инсталляция)

3 – каталоги группы пользователей (определяет администратор)

4 – головные (личные) каталоги пользователей (строятся при авторизации - HOME-каталоги)

5 – рабочие каталоги и файлы пользователей

**Некоторые системные каталоги:**

vmunix - ядро ОС;

bin - утилиты оболочки;

dev - файлы устройств;

tmp - адреса рабочих пространств памяти пользователей, выделяемых супервизором памяти ОС по запросам системных вызовов из программ пользователей;

etc - файлы администратора;

mnt - каталог монтирования новых файловых систем;

home - данные головных каталогов всех пользователей системы.

В UNIX как и в DOS используется простой способ описания пути к файлу в иерархической системе каталогов -

*полное маршрутное имя файла (абсолютное маршрутное):*

$ cat /home/user/lev/f1

*относительные маршрутные имена файлов:*

* от головного каталога группы пользователей - user/lev/f1
* от головного (личного) каталога пользователя - f1 или lev/f1
* от текущего (рабочего в данный момент) каталога,
* " . " - системное имя текущего каталога (скрытый файл)
* " .. " - системное имя вышележащего каталога (скрытый файл).

### *Основные команды работы с каталогами*

**man название\_команды**  - вызов электронного справочника об указанной команде.

*Например*: *man exit* – выведет справку по команде exit.

Выход из справочника - нажатие клавиши Q.

Команда  man man  сообщает информацию о том, как пользоваться справочником.

Помимо справочника можно воспользоваться командой **help**. Для этого нужно набрать:

*команда* --help

*Например*: exit --help – выведет справку по команде exit.

#### cd имя\_директории- смена текущего каталога. В качестве имени директории можно указывать элементы "." и "..".

#### mkdir имя\_директории - создание каталога. Вновь созданный каталог не содержит элементов, за исключением "." (ссылка на текущий каталог) и ".." (ссылка на вышележащий каталог).

#### rmdir имя\_директории - удаление каталога.

**ls [-alrstu] [namedir]** - вывод содержимого каталога; если в качестве *namedir* указано имя файла, то выдается вся информация об этом файле. Если аргумент не указан, выдается содержимое текущего каталога.

*Значения некоторых аргументов:*

**- l** - список включает всю информацию о файлах;

- **F** - добавление к имени каталога символа / и символа \* к имени файла, для которых разрешено выполнение;

**- a** - в список включаются все файлы, в том числе и те, которые начинаются с точки;

**- i** - указать идентификационный номер каждого файла;

- **R** - рекурсивный вывод содержимого подкаталогов заданного каталога.

**find список\_каталогов условия\_поиска -**определение полных имен файлов в поддереве каталогов, удовлетворяющих заданным условиям поиска**.**

В команде может быть задано до 18 условий\_поиска. Необходимые комбинации условий объединяются в булевское выражение с помощью элементарных логических операций ('элементарных логических функций). Таким образом формируется логическая функция, принимающая истинное значение если значения всех её составляющих истины.

На экран выводятся полные маршрутные имена файлов, свойства которых обеспечивают истинность этой функции.

*Элементарные логические функции:*

! <условие> отрицание условия;

<пробел> соответствует операции "И";

-o операция "ИЛИ";

\( выражение \) булевское выражение в скобках.

*При описании команды используются обозначения:*

n положительное десятичное число;

-n любое положительное десятичное число, строго меньшее n;

+n любое положительное десятичное число, строго большее n.

*Условия задаются следующими опциями:*

-name'имя\_файла' истинно для файлов с указанным именем.

-perm <8-ричный\_код> истинно для файлов с указанным кодом прав доступа.

-type {f|d|b|c|p} истинно для файлов указанного типа.

-links n истинно для файлов с числом ссылок n.

-user <имя\_пользователя> истинно для файлов, принадлежащих данному пользователю.

-size n[c] истинно для файлов с длиной n (число блоков) или с – в символах (байтах).

-exec <команда> {} \; истинно, если команда возвращает нулевой код завершения (true), т.е. опция выполняет <команда> для каждого найденного файла.

Все аргументы команды find разделяются пробелами.

*Примеры использования команды find:*

1) команда выводит на экран список файлов с именами *fil* или файлов у которых имя владельца *petr*:

*find / --name “fil.\*” –o –user petr –exec cat {} \;*

2) поиск файлов, не являющихся директориями в текущем каталоге и его подкаталогах:

*find \* !-type d*

3) команда осуществит поиск в корневом каталоге всех директорий, на которые существуют более трех жестких ссылок:

*find / -type d -links +3*

**grep [-ключи] 'шаблон' <список\_файлов> - поиск в файлах из списка\_файлов строк, содержащих указанный шаблон (подстрока символов).**

*Ключи определяют режимы поиска и вывода:*

-c выводятся имена всех просмотренных файлов и количество найденных строк, содержащих шаблон;

-n перед каждой строкой выводится ее относительный номер в файле и сама строка;

-i игнорируются регистры;

-l выводятся только имена файлов, содержащие найденные строки и др.

*Например:*

grep -c 'aaa' \*

””””””””

f.1: 10

f.2: 3

f.3: 1

””””””””

В рассмотренном примере выводятся все имена файлов текущего каталога, содержащих подстроку *aaa* и количество таких строк в каждом из них.

В нижеследующем примере выводятся только имена файлов, содержащих строки с шаблоном *aaa*.

$ grep -n bbb \* <NewLine>

””””””””””””””””””””””””””””

f.1: 5: aaaaaabbbbaaacc

f.2: 2: bbbbbbbbbbbbbbbbbbbb

f.6: 1: bbb

””””””””””””””””””””””””””””

#### Практическая работа №2(1) Основные команды работы с каталогами

##### Цель работы:

1. Изучить архитектуру и принципы функционирования многопользовательской многозадачной операционной системы Linux
2. Изучить основные команды работы с каталогами ОС Linux

##### Порядок выполнения работы

Изучаются команды:

mkdir - создание каталога;

rmdir - удаление каталога;

cd - перемещение по дереву каталогов;

pwd - определение текущего каталога;

ls - просмотр содержимого каталога;

more - постраничный вывод информации;

man - вызов руководства по команде.

1. Зарегистрируйтесь в ОС Linux. Запустите на выполнение программу Терминал.
2. Изучить команды работы с каталогами, для этого выведите на экран Терминала справку по следующим командам: *pwd, mkdir, rmdir, cd, ls*. Назначение и формат команд приведите в отчете.
3. Определите Ваш текущей каталог. Просмотрите его содержимое и приведите его в отчете.
4. Перейдите в каталог Вашей группы и создайте в нем личный каталог.
5. Определите уникальное имя (номер индексного указателя) Вашего личного каталога. Объясните структуру полного маршрутного имени каталога.
6. Создайте два поддерева из одного и из двух каталогов в каталоге Вашей группы. Используя команду ls, проверьте факт построения дерева подкаталогов.
7. Просмотрите содержимое пустых подкаталогов, т.е. новых подкаталогов, не содержащих файлов. Объясните их содержание.
8. Сделайте текущим последний каталог меньшего поддерева. Определите его полное маршрутное имя.
9. Уничтожьте все подкаталоги большего поддерева каталогов. Получите подтверждение выполнения команд.
10. Выведите на экран содержимое корневого каталога. Приведите его в отчете. Какие каталоги расположены в корневой директории, каково их назначение?
11. Завершите сеанс работы с системой.

#### Практическая работа №2(2) Поиск файлов в системе каталогов

##### Цель работы:

1. Изучить архитектуру и принципы функционирования многопользовательской многозадачной операционной системы Linux
2. Научиться осуществлять поиск файлов в системе каталогов

##### Порядок выполнения работы

Изучаются команды:

find - поиск файлов в системе каталогов

more - постраничный вывод информации;

man - вызов руководства по команде;

grep - поиск в файлах указанный шаблон;

еxit - выход из терминала;

1. Зарегистрируйтесь в ОС Linux. Запустите на выполнение программу Терминал.
2. Изучить команды работы с каталогами, для этого выведите на экран Терминала справку по следующим командам: *find, grep*. Назначение и формат команд приведите в отчете.
3. Установите в качестве текущего HOME-каталог.
4. Найдите обычные файлы с определением их полных маршрутных имен.
5. Осуществите поиск файлов типа директория в Вашем личном каталоге.
6. Найдите в текущем каталоге все файлы, на которые имеются жесткие ссылки.
7. Найдите файлы блок-ориентированного типа из каталога /dev и его подкаталогов.
8. Осуществите поиск обычных файлов на диске, размер которых не превышает 2 Кб.
9. Задайте различные комбинации известных Вам условий поиска файлов и каталогов.
10. Выведите на экран принадлежащую Вам регистрационную запись с использованием команды grep.
11. Завершите сеанс работы с системой.

## 2.1.4. Команды работы с файлами: управление файлами, генерация имен файлов

### *Управление файлами*

Ниже приведены форматы команд, использующихся для управления файлами:

**>маршрутное-имя-файла** - создание пустого файла.

**cat имя-файла**  - вывод содержимого файла на экран.

**cat>имя-файла**  - перенаправление информации, вводимой с клавиатуры с заменой содержимого файла.

**cat [входной файл 1] [входной\_файл2] […входной файл N] > [выходной\_файл] -** слияние файлов

**grep [-ключи] 'шаблон' <список\_файлов> -поиск в файлах из списка\_файлов строк, содержащих указанный шаблон (подстрока символов).**

*Ключи определяюobt режимы поиска и вывода:*

-c выводятся имена всех просмотренных файлов и количество

найденных строк, содержащих шаблон;

-n перед каждой строкой выводится ее относительный номер в

файле и сама строка;

-i игнорируются регистры;

-l выводятся только имена файлов, содержащие найденные строки и др.

**touch [-k] имя-файла -** замена времени модификации на текущее, если файл не существует, то создается новый пустой файл. Команда может использоваться для создания новых пустых файлов, если указать имя несуществующего файла.

**wc [-lwc] список\_файлов -** подсчет количества строк (-l), слов (-w), или символов (-c) в указанных файлах.

**cp [-k] вх\_файл\_1 [вх\_файл\_2 [... вх\_файл\_n]] вых\_файл –** копирование файлов.

В самом формате команды заложено два режима ее использования:

*1-й:* Если вых\_файл - обычный файл, то вх\_файл может быть только один; в этом случае содержимое вх\_файла копируется в вых\_файл. Если вых\_файл существовал, то его содержимое полностью заменяется на новое, атрибуты защиты сохраняются. Если создается новый вых\_файл, то ему присваиваются атрибуты копируемого.

*2-й:* Если вых\_файл - каталог, то в него последовательно копируются все указанные вх\_файлы со своими атрибутами и именами, но каталог при этом автоматически не создается.

*Примеры использования команды cp:*

Создается новый файл FIL.c в том же текущем головном каталоге. Для копирования в другой каталог необходимо указать маршрутное имя выходного файла:

cd

cp fil.c FIL.c

Входной файл fil.c сохраняется в исходном текущем каталоге:

mkdir petr

mkdir lev1

cp fil.c lev1/prog.c

Групповые операции над файлами при копировании:

cp f.? fildir

cp f.[1-3] fildir

cp f.[123] fildir

**mv [-k] вх\_ф йл\_1 [вх\_ф йл\_2 [... вх\_ф йл\_n]] вых\_файл -** перемещение или переименование файлов

Отличия от предыдущей команды копирования заключаются в том, что перемещаемые файлы в исходном каталоге уничтожаются.

### *Генерация имен файлов и каталогов (Pathname Expansion)*

Подстановки имен путей и файлов (Pathname expansion) используются для того, чтобы с помощью краткого образца или шаблона указать несколько имен файлов (или каталогов), соответствующих данному шаблону.

Специальные символы шаблонов имеют следующее значение.

***Таблица 2.1.4.*** *Символы шаблонов*

|  |  |
| --- | --- |
| **Символ** | **Правила замены** |
| \* | Соответствует произвольной строке символов, включая пустую строку. Например, my\*.txt будет заменено на myday.txt, myweek.txt и mymonth.txt (если такие файлы существуют), а \*.jpg соответствует всем файлам с расширением jpg в указанном каталоге |
| ? | Соответствует любому одиночному символу. Например, вместо шаблона file?.txt будут подставлены имена file1.txt и filex.txt, но не file10.txt |
| [...] | Соответствует любому символу из числа символов, указанных в скобках. Пары символов, разделенные знаком минуса, обозначают интервал; любой символ стоящий лексически между этими двумя символами, включая и символы, задающие интервал, соответствует шаблону. Если первым символом внутри скобок является ! или ^, то считается, что шаблону (в данной позиции) соответствуют все символы, не указанные в скобках |

Шаблоны имен файлов очень часто применяются в командных строках, содержащих команду ls. Представьте себе, что вы хотите просмотреть информацию о содержимом каталога, в котором находится огромное количество разных файлов различных форматов, например, файлов с изображениями форматов gif, jpeg, avi и т. д.. Чтобы получить только список файлов формата jpeg, вы можете использовать команду

[user]$ ls \*.jpg

Если в каталоге имеется множество файлов, имена которых представлены четырехзначными номерами, то следующей командой можно вывести только список файлов с номерами от 0200 до 0499:

[user]$ ls -l 0[2-4]??.\*

#### Практическая работа №3(1). Команды управления фалами

##### Цель работы:

1. Изучить основные команды работы с файлами ОС Linux
2. Научиться создавать, объединять, удалять, переименовывать файлы.

##### Порядок выполнения работы

Изучаются команды:

cat - слияние и вывод файлов на стандартное устройство вывода(конкатенация файлов);

rm - удаление файла;

mv - переименование файла;

сp - копирование файлов;

wc - подсчет количества строк и слов в файле.

1. Зарегистрируйтесь в ОС Linux. Запустите на выполнение программу Терминал.
2. Изучите команды работы с файлами, для этого выведите на экран Терминала справку по следующим командам: *cat, rm, mv, cp, wc*. Назначение и формат команд приведите в отчете.
3. Перейдите в Ваш личный каталог и выведете на экран его содержимое.
4. Создайте 3-4 текстовых файла с частично совпадающими именами. Проанализируйте значения атрибутов Ваших файлов.
5. Создайте еще один файл методом слияния из существующих. Как изменились атрибуты нового файла?
6. Создайте два новых параллельных подкаталога.
7. В один подкаталог скопируйте имеющиеся файлы HOME-каталога с изменением имен, а в другой – переместите. Проанализируйте как изменилось содержание всех трех каталогов и каковы атрибуты всех полученных файлов.
8. Удалите созданные текстовые файлы и каталоги. Выйдите из программы Терминал.

#### Практическая работа №3(2) Использование шаблонов в именовании файлов

##### Цель работы:

1. Изучить понятие шаблона, способы задания шаблонов файлов
2. Научиться осуществлять просмотр директорий, копирование, удаление, и поиск файлов, используя шаблоны файлов.

##### Порядок выполнения работы

1. Зарегистрируйтесь в ОС Linux. Запустите на выполнение программу Терминал.
2. Осуществите поиск всех файлов в домашнем каталоге, которые имеют расширение *txt*.
3. Осуществите поиск директорий, в составе имени которых, имеется любой однозначное число.
4. Осуществите в корневом каталоге поиск всех файлов, содержащих в составе имени слово linux.
5. Создайте в Вашем личном каталоге структуру каталогов, содержащих совпадающие имена.
6. Просмотрите содержимое файлов домашнего каталога с частично совпадающими именами с использованием механизма генерации имен файлов.
7. Просмотрите содержимое файлов каталога /dev с частично совпадающими именами с использованием механизма генерации имен файлов.
8. С использованием механизма генерации имен файлов слейте содержимое всех файлов в один и поместите его в HOME-каталог. Проанализируйте содержание файла.
9. С использованием механизма генерации имен файлов слейте содержимое всех файлов в один и поместите его в HOME-каталог. Проанализируйте содержание файла.
10. С использованием механизма генерации имен скопируйте из Вашего личного каталога все файлы с частично совпадающими именами в домашний каталог.
11. С использованием механизма генерации имен удалите в Вашем личном каталоге все файлы с частично совпадающими именами.
12. С использованием механизма генерации имен удалите в Вашем личном каталоге все каталоги с частично совпадающими именами.
13. Удалите из домашнего каталога скопированные ранее файлы и завершите работу с программой Терминал.

## 2.1.7. Команды работы с файлами: ссылки, сортировка файлов

**ln [-ключ] вх\_файл\_1 [вх\_файл\_2 [... вх\_файл\_n]] вых\_файл -** организация новых ссылок на файл.

В формате команды заложено 2 режима ее использования:

*1-й:* Если вых\_файл - обычный файл, то допускается только один вх\_файл\_1; в этом случае на н равным именам: вх\_файл\_1 и вых\_файл.

Количество ссылок на файл в его описателе увеличивается на 1.

*2-й:* Если вых\_файл - каталог, то в нем создаются элементы, включающие имена перечисленных в команде входных файлов и ссылки на них, после чего в каталоге вых\_файл можно работать с этими файлами, как с файлами этого каталога.

Возможно создание символьных и жестких ссылок.

При создании жесткой ссылки ключ в команде не указывается. Символьная ссылка создается командой ln с ключом *s*. Символьная ссылка ссылается на полное имя файла, жесткая – на его инод (числовой номер, присеваемый файлу при его создании).

*Обобщение свойств команды ln при создании жесткой ссылки:*

* каждая ссылка на входной файл добавляет +1 к числу ссылок входного файла, после чего к упомянутому файлу можно обращаться по двум именам;
* все связанные файлы имеют совпадающие описатели в каталогах;
* если изменяется содержимое одного из связанных файлов, то меняется содержимое всех связанных файлов, т.к. связанные файлы разделяют одни и те же данные на диске;
* если удалить один из связанных файлов, то сокращается число взаимных ссылок на -1.

**ls [-alrstu] [namedir]**  - вывод содержимого каталога; если в качестве *namedir* указано имя файла, то выдается вся информация об этом файле. Если аргумент не указан, выдается содержимое текущего каталога.

*Значения некоторых аргументов:*

- t - сортировка по времени модификации файлов;

- v - сортировка файлов по времени последнего доступа;

- с - использовать время создания файла при сортировке;

- s - размеры файлов указываются в блоках;

**sort [-k номер-поля-в-строке] [список-файлов] -** алфавитная и числовая сортировка файлов или строк файлов; режим сортировки определяется ключами. Команда многофункциональная, предусматривает много режимов сортировки. Некоторые ключи команды:

-d - сортировка по алфавиту (или по умолчанию);

-n - числовая;

-u - исключает повторяющиеся строки и т.д.

#### Практическая работа №3(3) Создание ссылок на файлы.

##### Цель работы:

1. Изучить понятие ссылки на файл.
2. Научиться создавать жесткие и символьные ссылки на файлы.

##### Порядок выполнения работы

Изучаются команды:

ln - создание ссылки на файл;

ls - просмотр атрибутов файла.

1. Зарегистрируйтесь в ОС Linux. Запустите на выполнение программу Терминал.
2. Выведите на экран Терминала справку по следующим командам: *ln, ls*. Назначение и формат команд приведите в отчете.
3. Перейдите в Ваш личный каталог и создайте в нем текстовый файл и любой подкаталог.
4. Создайте жесткую ссылку с другим именем в созданном подкаталоге на созданный файл. Проанализируйте и сравните все атрибуты связанных файлов. Объясните отличия.
5. Уничтожьте один из связанных файлов. Проанализируйте и сравните все атрибуты связанных с ним файлов. Объясните изменения.
6. Просмотрите содержание связанных файлов. Каков был результат просмотра?
7. Создайте символьную ссылку с другим именем в созданном ранее подкаталоге. Проанализируйте и сравните все атрибуты связанных файлов. Объясните отличия.
8. Создайте новый подкаталог в Вашем личном каталоге.
9. Создайте ссылки на несколько файлов, принадлежащих одному из каталогов домашней директории, в созданном ранее каталоге одной командной строкой. Проанализируйте и сравните все атрибуты связанных файлов. Объясните отличия.
10. Завершите работу с программой Терминал.

#### Практическая работа №3(4) Сортировка файлов

##### Цель работы:

1. Научиться сортировать отображаемую информацию при просмотре директории.
2. Научиться осуществлять сортировку данных в файле.

##### Порядок выполнения работы

Изучаются команды:

sort - сортировка файлов;

ls - просмотр содержимого каталога;

touch - замена времени модификации файла на текущее.

1. Зарегистрируйтесь в ОС Linux. Запустите на выполнение программу Терминал.
2. Выведите на экран Терминала справку по следующим командам: *sort, touch*. Назначение и формат команд приведите в отчете.
3. Просмотрите содержимое корневого каталога с выводом размеров файлов, содержащихся в директории, в блоках.
4. Упорядочите по алфавиту строки суммарного файла HOME-каталога.
5. Просмотрите содержимое любой директории отсортировав отображаемую информацию по времени последнего доступа к файлу.
6. Обновите временные характеристики одного из существующих файлов. Проанализируйте результат.
7. Просмотрите содержимое директории, содержащей файл с обновленными временными характеристиками отсортировав его по времени модификации файлов
8. Обновите временные характеристики несуществующего файла. Проанализируйте результат работы команды.
9. Просмотрите содержимое директории, содержащей созданный файл, используя сортировку по времени создания файла
10. Проанализируйте с использованием команды history содержание лабораторной работы, продумайте ответы на нижеприведенные контрольные вопросы и сдайте выполненную работу преподавателю.

### *Контрольные вопросы:*

1. Какие системные имена каталогов Вам известны?
2. Каким образом можно построить отдельный каталог или цепочку каталогов?
3. Для чего и каким образом переопределяются текущие каталоги?
4. Как обратиться к файлам параллельных ветвей дерева

каталогов? К вышележащему каталогу?

1. Какие условия поиска файлов Вы знаете? Как комбинируются условия поиска? Как осуществить поиск по дереву каталогов?
2. Какова последовательность действий при удалении одного каталога? Цепочки каталогов?
3. Объясните назначение и содержание каждого поля каталога.
4. Как отличить по содержимому каталога типы файлов, содержащихся в Ваших каталогах.
5. Какую информацию содержит «пустой» вновь созданный каталог?
6. Как осуществить поиск файлов в системе каталогов по фрагментам текста файлов?
7. Назовите известные Вам способы создания пустых файлов.
8. Как создать текстовый файл?
9. Какие возможности сокращения записи имен файлов Вы знаете с использованием механизма генерации имен файлов?
10. Какие три команды этой лабораторной работы можно использовать для переименования файлов? Как в этом случае надо использовать команды?
11. Какими способами можно объединить несколько текстовых файлов в один?
12. В чем разница работы команд cp и mv?
13. Сколько ссылок можно создать на единственный файл из разных каталогов?
14. Как создать несколько ссылок с совпадающими именами на несколько файлов в другом каталоге?
15. Какое соответствие атрибутов имеют связанные между собой файлы?
16. На какой атрибут и как влияет удаление одного из связанных файлов?
17. Как отражается на содержимом связанных файлов изменение содержания одного из них и почему?
18. Какими возможностями обладает команда sort?

# Тема 2.2. Управление процессами

## 

## 2.2.1. Управление процессами

### *Отображение информации о процессах*

В Linux (как и во всех UNIX-системах) имеется команда ps, которая позволяет определить, какие процессы в системе запущены. Если ее запустить без всяких параметров, то она выдает список процессов, запущенных в текущей сессии.

**ps [-опции] –** определение запущенных в системе процессов

*Краткая характеристика наиболее важных опций:*

*-* ax - список всех процессов, запущенных в системе;

–u – отображение % ЦПУ и памяти занимаемых запущенными процессами;

–l – длинный формат отображения информации о процессах (с выводом приоритета процесса и значениния nice)

Независимо от наличия опций команда ps выдает для каждого процесса отдельную строку, но содержимое этой строки может быть разным. В зависимости от заданных опций могут присутствовать следующие поля:

* USER — имя владельца процесса;
* PID — идентификатор процесса в системе;
* PPID — идентификатор родительского процесса;
* %CPU — доля времени центрального процессора (в процентах), выделенного данному процессу;
* %MEM — доля реальной памяти (в процентах), используемая данным процессом;
* VSZ — виртуальный размер процесса (в килобайтах);
* RSS — размер резидентного набора (количество 1K-страниц в памяти);
* STIME — время старта процесса;
* TTY — указание на терминал, с которого запущен процесс;
* S или STAT — статус процесса;
* PRI — приоритет планирования;
* NI — значение nice (см. описание командыniceниже);
* TIME — сколько времени центрального процессора занял данный процесс;
* CMD или COMMAND — командная строка запуска программы, выполняемой данным процессом;

а также и другие поля, полный список которых приведен на man-странице, посвященной команде ps.

В поле **Статус процесса** могут стоять следующие значения:

* R — выполнимый процесс, ожидающий только момента, когда планировщик задач выделит ему очередной квант времени;
* S — процесс "спит";
* D — процесс находится в состоянии подкачки на диске;
* T — остановленный процесс;
* Z — процесс-зомби.

Рядом с указателем статуса могут стоять дополнительные символы из следующего набора:

* W — процесс не имеет резидентных страниц;
* < — высоко-приоритетеный процесс;
* N — низко-приоритетный процесс;
* L — процесс имеет страницы, заблокированные в памяти.

Например, результат выполнения нижеприведенной команды показал, что в системе работали на момент снятия данных два процесса sh (с идентификатором 927, идентификатор процесса предка - 1) и ps (с идентификатором 1001, идентификатор процесса предка – 927, т.е. sh порождает процесс ps). Терминалы с которых запущены процессы – tty5.

*ps -flu lev*

PID PPID TTY PRI TIME CMD

927 1 tty5 0:04 sh

1001 927 tty5 0:02 ps

Для вывода листинга процессов, принадлежащих пользователю можно воспользоваться конструкцией:

ps -f|grep <имя\_пользователя>.

**Команда top**

Команда ps позволяет сделать как бы "моментальный снимок" процессов, запущенных в системе. В отличие от ps команда top отображает состояние процессов и их активность "в реальном режиме времени". На рисунке 2.2.1. изображено окно терминала, в котором запущена программа top.

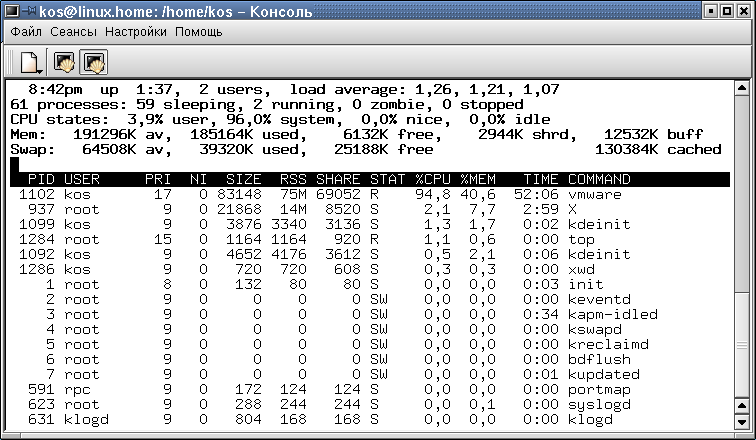


Рис. 2.2.1. Вывод команды top

В верхней части окна отображается астрономическое время, время, прошедшее с момента запуска системы, число пользователей в системе, число запущенных процессов и число процессов, находящихся в разных состояниях, данные об использовании ЦПУ, памяти и свопа. А далее идет таблица, характеризующая отдельные процессы. Число строк, отображаемых в этой таблице, определяется размером окна: сколько строк помещается, столько и выводится. Графы таблицы обозначены так же, как поля вывода команды ps. Содержимое окна обновляется каждые 5 секунд.

### *Приоритеты, значение nice и команда renice*

Приоритет для каждого процесса устанавливается в тот момент, когда процесс порождается. Приоритет процесса определяется так называемым "значением nice", которое лежит в пределах от +20 (наименьший приоритет, процесс выполняется только тогда, когда ничто другое не занимает процессор), до -20 (наивысший приоритет).

Значение nice устанавливается для каждого процесса в момент порождения этого процесса и при обычном запуске команд или программ принимается равным приоритету родительского процесса. Но существует специальная команда nice, которая позволяет изменять значение nice при запуске программы. Формат использования этой программы:

**nice [- adnice] command [args]**

где adnice — значение (от –20 до +19), добавляемое к значению nice процесса-родителя. Полученная сумма и будет значением nice для запускаемого процесса. Отрицательные значения может устанавливать только суперпользователь. Если опция — adnice не задана, то по умолчанию для процесса-потомка устанавливается значение nice, увеличенное на 10 по сравнению со значением nice родительского процесса. Очевидно, что если вы не суперпользователь, то применять эту команду имеет смысл только тогда, когда вы хотите запустить некий процесс с низким значением приоритета.

Другая команда, renice, служит для изменения значения nice для уже выполняющихся процессов. Ее формат таков:

**renice priority [[-p] PID] [[-g] grp] [[-u] user]**

*Например*, команда

*renice -1 987 –u daemon –p 32*

увеличивает на 1 приоритет процессов с PID 987 и 32, а также всех процессов пользователя daemon.

Суперпользователь может изменить приоритет любого процесса в системе. Другие пользователи могут изменять значение приоритета только для тех процессов, для которых данный пользователь является владельцем. При этом обычный пользователь может только уменьшить значение приоритета (увеличить значение nice), но не может увеличить приоритет, даже для возврата значения nice к значению, устанавливаемому по умолчанию. Поэтому процессы с низким приоритетом не могут породить "высокоприоритетных детей".

### *Сигналы и команда kill*

**Сигналы** — это средство, с помощью которого процессам можно передать сообщения о некоторых событиях в системе.

Сами процессы тоже могут генерировать сигналы, с помощью которых они передают определенные сообщения ядру и другим процессам. С помощью сигналов можно осуществлять такие акции управления процессами, как приостановка процесса, запуск приостановленного процесса, завершение работы процесса. Всего в Linux существует 63 разных сигнала, их перечень можно посмотреть по команде

*kill –l*

Сигналы принято обозначать номерами или символическими именами. Все имена начинаются на SIG, но эту приставку иногда опускают: например, сигнал с номером 1 обозначают или как SIGHUP, или просто как HUP.

Когда процесс получает сигнал, то возможен один из двух вариантов развития событий. Если для данного сигнала определена подпрограмма обработки, то вызывается эта подпрограмма. В противном случае ядро выполняет от имени процесса действие, определенное по умолчанию для данного сигнала. Вызов подпрограммы обработки называется перехватом сигнала. Когда завершается выполнение подпрограммы обработки, процесс возобновляется с той точки, где был получен сигнал.

Можно заставить процесс игнорировать или блокировать некоторые сигналы. Игнорируемый сигнал просто отбрасывается процессом и не оказывает на него никакого влияния. Блокированный сигнал ставится в очередь на выдачу, но ядро не требует от процесса никаких действий до разблокирования сигнала. После разблокирования сигнала программа его обработки вызывается только один раз, даже если в течение периода блокировки данный сигнал поступал несколько раз.

В табл. 2.2.1. приведены некоторые из часто встречающихся сигналов.

Таблица 2.2.1. Сигналы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Имя** | **Описание** | **Можно перехватывать** | **Можно блокировать** | **Комбинация клавиш** |
| 1 | HUP | Hangup. Отбой | Да | Да |  |
| 2 | INT | Interrupt. В случае выполнения простых команд вызывает прекращение выполнения, в интерактивных программах — прекращение активного процесса | Да | Да | <Ctrl>+<C> или <Del> |
| 3 | QUIT | Как правило, сильнее сигнала Interru | Да | Да | <Ctrl>+<\> |
| 4 | ILL | Illegal Instruction. Центральный процессор столкнулся с незнакомой командой (в большинстве случаев это означает, что допущена программная ошибка). Сигнал отправляется программе, в которой возникла проблема | Да | Да |  |
| 8 | FPE | Floating Point Exception. Вычислительная ошибка, например, деление на ноль | Да | Да |  |
| 9 | KILL | Всегда прекращает выполнение процесса | Нет | Нет |  |
| 11 | SEGV | Segmentation Violation. Доступ к недозволенной области памяти | Да | Да |  |
| 13 | PIPE | Была предпринята попытка передачи данных с помощью конвейера или очереди FIFO, однако не существует процесса, способного принять эти данные | Да | Да |  |
| 15 | TERM | Software Termination. Требование закончить процесс (программное завершение) | Да | Да |  |
| 17 | CHLD | Изменение статуса порожденного процесса | Да | Да |  |
| 18 | CONT | Продолжение выполнения приостановленного процесса | Да | Да |  |
| 20 | TSTR | Сигнал останова, генерируемый клавиатурой. Переводит процесс в фоновый | Да | Да | <Ctrl>+<Z> |

Как видно из таблицы 2.2.1, некоторые сигналы можно сгенерировать с помощью определенных комбинаций клавиш. Но такие комбинации существуют не для всех сигналов. Зато имеется команда kill, которая позволяет послать заданному процессу любой сигнал.

Для посылки сигнала процессу (или группе процессов) можно воспользоваться командой kill в следующем формате:

*kill [-сигн] PID [PID..]*

где *сигн* — это номер сигнала, причем если указание сигнала опущено, то посылается сигнал 15 (TERM — программное завершение процесса). Чаще всего используется сигнал 9 (KILL), с помощью которого суперпользователь может завершить любой процесс. Но сигнал этот очень "грубый", если можно так выразиться, поэтому его использование может привести к нарушению порядка в системе. Поэтому в большинстве случаев рекомендуется использовать сигналы TERM или QUIT, которые завершают процесс более "мягко".

Наиболее часто команду kill вынужден применять суперпользователь. Он должен использовать ее для уничтожения процессов-зомби, зависших процессов (они показываются в листинге команды ps как <exiting>), процессов, которые занимают слишком много процессорного времени или слишком большой объем памяти и т. д. Особый случай — процессы, запущенные злоумышленником.

### *Перевод процесса в фоновый режим*

Если вы запускаете какой-то процесс путем запуска программы из командной строки, то обычно процесс запускается, как говорят, "на переднем плане". Это значит, что процесс "привязывается" к терминалу, с которого он запущен, воспринимая ввод с этого терминала и осуществляя на него вывод. Но можно запустить процесс в фоновом режиме, когда он не связан с терминалом. Для запуска процесса в фоновом режиме в конце командной строки запуска программы добавляют символ & (амперсанд).

*Например*, в фоновом режиме запускается процесс cc prog.c:

$ cc prog.c &

2388

$

Shell выводит номер этого процесса (PID) и разрешает ввод следующей команды.

Фоновые процессы обладают некоторыми *недостатками*:

* не допускают ввода с клавиатуры;
* обспечивают вывод на экран, но при этом нарушают целостность вывода диалогового процесса.

Общепринятый прием *исключения влияния фонового вывода на интерактивную работу*:

**ком\_строка>имя\_файла.out &**

* ком\_строка планирует задание для фонового режима
* перенаправляет вывод вместо экрана в указанный файл головного каталога пользователя.

*Пример:* Запуск программы, осуществляющей поиск файлов по шаблону «ааа\*», в фоновом режиме, результаты поиска перенаправляются в файлgrep.out*:*

$grep ааа\* > grep.out &

194

$ps

PID TTY TIME CMD

194 tty5 0:02 grep

200 tty5 0:01 ps

В оболочке bash имеются две встроенные команды, которые служат для перевода процессов на передний план или возврата их в фоновый режим. Но прежде, чем рассказывать об этих командах, надо рассказать о команде **jobs**. Она всегда вызывается без аргументов и показывает задания, запущенные из текущего экземпляра shell. В начале каждой строки вывода этой команды указывается порядковый номер задания в виде числа в квадратных скобках. После номера указывается состояние процесса: stopped (остановлен), running (выполняется) или suspended (приостановлен). В конце строки указывается команда, которая исполняется данным процессом. Один из номеров выполняющихся заданий помечен знаком +, а еще один — знаком -. Процесс, помеченный знаком +, будет по умолчанию считаться аргументом команд fg или bg, если они вызываются без параметров. Процесс, помеченный знаком -, получит знак +, если только завершится по какой-либо причине процесс, который был помечен знаком +.

Команды **fg** и **bg** служат для перевода процессов на передний план или возврата их в фоновый режим. В качестве аргумента обеим этим командам передаются номера тех заданий, которые присутствуют в выводе команды jobs. Если аргументы отсутствуют, то подразумевается задание, помеченное знаком +. Команда fg переводит указанный в аргументе процесс на передний план, а команда bg — переводит процесс в фоновый режим. Одной командой bg можно перевести в фоновый режим сразу несколько процессов, а вот возвращать их на передний план необходимо по одному.

#### Практическая работа №4(1) Команды управления процессами. Создание фоновых процессов

##### Цель работы:

1. Закрепить представление о возможностях командного языка UNIX по управлению процессами, которым выделяются все необходимые ресурсы вычислительной системы
2. Научиться запускать на выполнение фоновые процессы и процессы переднего плана, получать моментальный снимок процессов в системе.

##### Порядок выполнения работы

Изучаются команды:

ps - запрос информации о процессах текущего терминала;

top получение информации о "состоянии процессов в режиме реального времени;

& - запуск фонового процесса;

jobs - запрос листинга списка заданий;

bg - перевод оперативного в фоновый режим;

fg перевод фонового процесса в оперативный.

1. Зарегистрируйтесь в ОС Linux. Запустите на выполнение программу Терминал.
2. Выведите на экран Терминала справку по следующим командам: *ps, top, jobs, bg, fg*. Назначение и формат команд приведите в отчете.
3. Выведите на экран листинг характеристик (в длинном и коротком форматах) процессов, инициализированных с Вашего терминала. Проанализируйте и объясните содержание каждого поля сообщения
4. Запустите утилиту получения информации о процессах в режиме реального времени. Проанализируйте и объясните содержание каждого поля сообщения
5. Запустите простейшую процедуру в фоновом режиме с бесконечным циклом выполнения, предусматривающую, например процесс, выводящий на экран бесконечный поток символов «y»:

уes &

1. Выведите на экран листинг характеристик (в длинном и коротком форматах) процессов, инициализированных с Вашего терминала. Объясните изменения в листинге характеристик процессов. Объясните содержание PID и PPID. Какой процесс является родительским для запущенной процедуры.
2. Выйдите из системы и войдите заново. Проанализируйте листинг процессов. Объясните изменения в системе.
3. Запустите процесс уes в оперативном режиме с перенаправлением вывода в устройство /dev/null («черная дыра»).
4. Переведите процесс в оперативный режим и проанализируйте сообщение на экране.
5. Приостановите выполнение процесса и переведите его в фоновый режим. Проанализируйте сообщение на экране.
6. Завершите работу всех запущенных вами процессов, переведя их на передний план и нажав клавиши Ctrl+C.
7. Завершите работу с Терминалом.

#### Практическая работа №4(2) Управление приоритетами процессов. Завершение работы процессов

##### Цель работы:

1. Закрепить представление о возможностях командного языка UNIX по управлению процессами, которым выделяются все необходимые ресурсы вычислительной системы
2. Научиться осуществлять управление существующими в системе процессами, изменяя их приоритеты и посылая им сигналы управления.

##### Порядок выполнения работы

Изучаются команды:

nice - понижение приоритета процесса;

kill - посылка сигнала процессу.

1. Зарегистрируйтесь в ОС Linux. Запустите на выполнение программу Терминал.
2. Выведите на экран Терминала справку по следующим командам: *nice, kill*. Назначение и формат команд приведите в отчете.
3. Запустите процесс уes в оперативном режиме с перенаправлением вывода в устройство /dev/null («черная дыра»).
4. Приостановите выполнение процесса, используя команду kill.
5. Переведите процесс в фоновый режим. Проанализируйте сообщение на экране.
6. Убейте процесс, послав командой kill сигнал завершения процесса.
7. Запустите процесс поиска директорий в корневом каталоге в фоновом режиме (выводимые данные перенаправьте в файл или «черную дыру»).
8. Выведите на экран листинг характеристик (в длинном формате) процессов, инициализированных с Вашего терминала. Каково значение приоритета запущенного процесса? Какой процесс является родительским для запущенной процедуры.
9. Понизьте значение приоритета процедуры поиска. На что и как повлияет эта операция при управлением вычислительным процессом системы? Как отразятся её результаты в описателях процессов?
10. Убейте все созданные ранее процессы и завершите работу с Терминалом.

### *Контрольные вопросы:*

1. Объясните понятия процесса и ресурса. Какое их значение в организации вычислительного процесса в ОС UNIX?
2. Какая информация содержится в описателях процессов? Как просмотреть их содержание в процессе работы с системой?
3. Какими способами можно организовать выполнение программ в фоновом режиме?
4. Какие особенности выполнения программ в фоновом режиме? Как избежать вывода фоновых сообщений на экран и прерывания выполнения фоновых программ при прекращении сеанса работы с системой?
5. Как пользователь может повлиять на распределение ресурсов между активными процессами?
6. Как можно прервать выполнение активных процессов? Какая информация для этого необходима и откуда она извлекается?

# Тема 2.3. Права доступа к файлам и каталогам

## 

## 2.3.1. Команды управления правами доступа

Каждый файл принадлежит конкретному пользователю. Владелец файла имеет абсолютный контроль над теми, кто из пользователей системы может иметь доступ к файлу. Владельцу предоставлены средства командного языка, позволяющими разрешать или запрещать доступ к своим файлам и каталогам.

Права процессов пользователей при доступе к файлу кодируются в атрибутах защиты файла. Атрибуты сопровождают каждый файл, хранятся в описателях файлов, на которые в каталоге имеются ссылки, и доступны для анализа и изменения посредством специальных команд ОС UNIX. Атрибуты защиты файла определяют права доступа трем видам процессов: процессам пользов ателя - владельца файла (u - user), процессам группы владельца файла (g - group) и процессам остальных пользователей (o - other), не попавших ни в одну из двух предыдущих категорий. Код атрибутов прав доступа пользователей трех перечисленных категорий для каждого файла отображается в полном листинге каталога символьным кодом в виде комбинации следующих символов:

**r -** разрешение на чтение или на выполнение файла, для каталога - *просмотр содержимого каталога (список всех файлов)*;

**w -** разрешение модификации или удаления файла, для каталога - *включение или удаление файлов***;**

**x -** разрешение выполнения файла (совместно с - r), для каталога - *поиск по каталогу конкретных отдельных файлов или сделать каталог текущим.*

Например, полный листинг каталога /udd/user1/lev может иметь следующий вид:

-rwxr-xr-x 1 lev user1 171 Mar 4 14:20 fil1.c

drwxr-xr-x 2 lev user1 32 Mar 4 14:51 hh

Здесь файл fil1.c, владельцем которого является пользователь со входным именем lev, является обычным, содержит исходный текст программы на языке Си длиной 171 байт, доступен владельцу для чтения, записи и выполнения, членам группы и прочим пользователям - только для чтения и выполнения. Директория hh защищена для включения новых и удаления существующих файлов.

Для изменения значений кодов защиты только указанных в команде файлов служит команда:

*chmod <коды защиты> <список\_файлов>*

Коды защиты (r, w, x) могут быть заданы только владельцем файла в символьном или числовом виде. Атрибуты задаются для владельца (u), его группы (g) и остальных пользователей (o) или для всех категорий пользователей одновременно (a).

Над символьными атрибутами защиты можно выполнять три следующие операции отдельно для владельца, для группы-владельца и для всех остальных пользователей:

= - присвоить значения кодов доступа (замена существующих);

+ - добавить значения кодов доступа;

- - отобрать права доступа.

Необходимо отметить, что новый файл обычно создается по умолчанию как невыполняемый, со стандартным набором прав доступа:

rw-rw-rw- - для файла;

rwxrwxrwx - для каталога.

Например, необходимо сделать некоторый файл shproc1 выполняемым, если он был создан как обычный. Для этого можно использовать следующую команду:

$ chmod u+x shproc1

$ shproc1

< Выполнение программы из файла shproc1 >

$

Эти действия необходимы и при формировании и выполнении shell-процедуры.

chmod a+x f1 - в данном случае файл f1 становится доступным для исполнения всем пользователям;

chmod a=rwx f2 - предоставляются все права всем категориям пользователей.

Числовые значения кодов защиты кодируются трехразрядным восьмеричным числом, где существование соответствующего кода соответствует наличию единицы в двоичном эквиваленте восьмеричной цифры этого числа, отсутствие атрибута - нулю.

*Например:*

Символьное представление: rwx r-x r--

Двоичное представление: 111 101 100

Восьмеричное представление: 7 5 4

Поэтому следующая команда:

chmod 0754 f3

- эквивалентна команде:

chmod u=rwx,g=rx,o=r f3

В результате выполнения команд в любой из приведенных форм коды доступа файла f3 приобретут следующий вид:

ls -l f3

”””””””””””””””””””””””””””””

-rwxr-xr-- ....................... f3

”””””””””””””””””””””””””””””

Таким образом, файл f3 является выполняемым для владельца и группы, чтение его разрешено всем пользователям, модифицировать файл может только владелец.

С целью защиты файла от удаления надо отобрать право (w) как у файла, так и у каталога, в котором находится файл.

Стандартные значения кодов прав доступа устанавливает администратор системы. Однако **пользователь в rsh может изменить временно (до конца сеанса работы) значение кода для всех своих новых файлов с помощью команды:**

*$umask [-r] <режим-доступа>*

Собственно *маска* - это двоичный код, с этим кодом и двоичным кодом установленным ране выполняются некоторые логические операции - в результате операции вычисляются новые коды защиты. В ksh - возможно символьное представление кодов защиты в umask, а собственно числовое значение маски просчитывается системой автоматически и используется для вычисления результирующих заданных в команде кодов доступа; ключ -S -выводит на экран текущие символьные значения кодов; без ключа - команда выводит числовое значение маски.

Имеются и другие возможности управления правами доступа. Приведенные ниже функции может выполнять только владелец файла или администратор.

**chown нов\_владелец имя\_файла –** владелец передает права владения данным файлом другому пользователю или группе.

**chgrp нов\_группа имя\_файла -** передача прав другой группе (сменить группу).

#### Практическая работа №5(1) Управление правами доступа к файлам

##### Цель работы:

1. Изучить принципы защиты файлов от несанкционированного доступа
2. Изучить влияние задаваемых прав доступа к файлу на выполнение различных команд по обработке этих файлов .

##### Порядок выполнения работы

Изучаются команды:

chmod - изменить права доступа к указанному файлу.

1. Зарегистрируйтесь в ОС Linux. Запустите на выполнение программу Терминал.
2. Выведите на экран Терминала справку по следующим командам: *chmod*. Назначение и формат команд приведите в отчете.
3. Создайте в Вашем HOME-каталоге один текстовый файл, например с именем f1. Выведите на экран полный листинг каталога.
4. Проанализируйте и умейте объяснить какие права доступа к f1 имеет владелец файла, его группа и остальные пользователи.
5. Выведите на экран содержимое файла f1. Объясните почему операция выполнилась успешно.
6. Запретите права на чтение f1 владельцу и группе. Попытайтесь вывести на экран текст файла. Объясните почему операция не выполняется.
7. Удалите права на запись в файл. Попытайтесь добавить к файлу текст и удалить его. Объясните результат.
8. Завершите работу с Терминалом

#### Практическая работа №5(2) Управление правами доступа к каталогам

##### Цель работы:

1. Изучить принципы защиты каталогов от несанкционированного доступа
2. Изучить влияние задаваемых прав доступа к каталогу на выполнение различных команд по обработке этих каталогов.

##### Порядок выполнения работы

Изучаются команды:

chmod - изменить права доступа к указанному каталогу.

1. Зарегистрируйтесь в ОС Linux. Запустите на выполнение программу Терминал.
2. Выведите на экран Терминала справку по следующим командам: *chmod*. Назначение и формат команд приведите в отчете.
3. Проанализируйте права доступа к Вашему личному каталогу. Есть ли ограничения на работу с файлами в этом каталоге?
4. Удалите право на модификацию каталога. Выполните операцию удаления файла f1, созданного в ходе выполнения работы 5(1). Объясните результат.
5. Создайте подкаталог. Разместите в нем текстовый файл. Проанализируйте права доступа к подкаталогу и объясните возможности по использованию подкаталога.
6. Удалите право владельца на «выполнение» подкаталога.
7. Попытайтесь сделать подкаталог текущим. Объясните результат.
8. Просмотрите содержимое подкаталога. Объясните результат.
9. Попытайтесь вывести длинный листинг подкаталога только для одного из файлов (поиск файла по подкаталогу). Объясните результат.
10. Попытайтесь вывести на экран содержимое файла. Объясните результат.
11. Верните право для подкаталога на «выполнение», удалите право на «чтение» и сохраните право на «модификацию».
12. Выполните п.п. 7, 8, 9, 10. Проанализируйте и объясните результаты
13. Верните право для подкаталога на «выполнение», удалите право на «чтение» и сохраните право на «модификацию»
14. Выполните п.п. 7, 8, 9, 10. Проанализируйте и объясните результаты
15. Завершите работу с Терминалом

### *Контрольные вопросы:*

* 1. Как кодируются в атрибутах файла и каталога права доступа?
  2. Кто может пользоваться и изменять права доступа к файлам?
  3. Какие команды для изменения символьных кодов прав доступа Вы знаете? Перечислите и расскажите о назначении каждой из команд.
  4. В чем разница в применении команд chmod и umask?
  5. Какие команды обработки файлов разрешают (или запрещают) права на чтение, модификацию и исполнение?
  6. Какие команды обработки каталогов разрешают (или запрещают) эти же права?
  7. Что означает право на выполнение, применительно к каталогу?
  8. Какими правами надо обладать, чтобы удалить файл или каталог?
  9. Какие команды для защиты файлов Вы знаете?

# Тема 2.4. Редактирование файлов средствами редактора vi

## 

## 2.4.1. Текстовый редактор vi

Во всех версиях ОС UNIX имеется встроенный текстовый редактор vi экранного типа. Он может применяться практически на любом терминале, начиная с телетайпа. Редактор является универсальным средством, доступным в среде любой UNIX-подобной ОС.

Редактор богат своими функциональными возможностями. Мы ограничимся наиболее часто используемыми функциями при составлении обычных текстов, текстов процедур и пр..

*Основная особенность – при редактировании текста в качестве управляющих используются обычные клавиши алфавитно-цифровой клавиатуры, а позиционирование осуществляется только относительно текущего положения курсора.*

Начало работы с редактором:

$vi имя\_файла [\*] вызов редактора и файла

Если файл не существует, то создается новый пустой

Редактируемый файл переписывается в буфер. Далее все изменения в редактируемом файле соответственно осуществляются в буфере в КОМАНДНОМ РЕЖИМЕ - нажатые клавиши интерпретируются как команды редактирования текста. В командный режим редактор переводится автоматически сразу же после входа в программу vi.

По завершении редактирования надо выйти из редактора с сохранением файла:

<Esc>:wq! - отредактированная копия замещает исходный файл. Если же требуется отменить все редактирование, то:

<Esc>:q!

В режиме редактирования, например, для ввода новых фрагментов текста файла; используются следующие клавиши

**команды режима ввода текста** (значения клавиш на экране не отображается):

a (append) - набор последующего текста в текущую строку после курсора

i (insert) - вставка текста в текущую строку после курсора

o (open) - вставить пустую строку после текущей

*Замечание*: если в качестве управляющей используется клавиша **нижнего** регистра – операция редактирования реализуется **после** курсора, если **верхнего - перед** курсором.

Операции ввода текста необходимо завершить командой

<Esc> - переводит редактор в режим редактирования последнего варианта текста. В командном режиме могут исполняться и другие группы команд.

**Команды управления курсором в области текста файла**. Два способа - использование специальных клавиш-команд (см справочник) или использование общеизвестных управляющих клавиш клавиатуры.

### Команды удаления текста

х - удаление текущего символа

[#]dw - удаление # текущих слов

[#]dd -удаление текущих строк

dG - удаление всех строк от текущей до последней

d$ - удаление конца строки от текущей позиции

d^ - удаление начала строки до текущей позиции.

*Замечание*: удаленные фрагменты текста не пропадают а временно помещаются в буфер, откуда онимогут быть возвращены в текст или перемещены по файлу (см. ниже).

**Команды отмены** произведенных изменений в текущей строке:

u - отмена (undo) последнего изменения

U - отмена всех изменений.

**Команды копирования** указанного в команде объекта в буфер:

[#]yw - копирование (yank) текущего слова

[#]yy - копирование текущей строки

уG - копирование строк от текущей до конца файла

yS - копирование части строки от курсора до конца строки

y^ - копирование части строки от курсора до начала строки

**Команды вставки буфера в текст:**

P - перед курсором

p - после курсора

**Команды изменения текста:**

r<символ> - заменяет (change) текущий символ на указанный

с<объек> - заменяет указанный объект на текст, который вводится с клавиатуры - <Esc>

* #cw -изменение текущего слова
* #cc - всей текущей строки
* cG - всех строк файла от текущей до последней
* cS -части строки от курсора до конца строки
* с^ - части строки от курсора до начала строки.

**Команды поиска строки файла по фрагменту текста:**

/<текст> - начиная от текущей строки до конца файла с переходом

на его начало

?<текст> - начиная от текущей строки до начала файла и по всему

n - поиск следующей строки в файле, аналогичной найденной

N - поиск предыдущего вхождения в файл заданной строки.

Редактор имеет возможности поиска строк по шаблонам (например по диапазону символов) и пр.

На экране содержимое редактируемого файла представляется как бы через некоторое окно со следующими полями (сверху - вниз):

* область текста файла;
* область пустых строк, которые надо ввести, если они понадобятся;
* тильда (~) - область для ввода дополнительного текста файла, но пустых строк этой области в файле нет;
* область сообщений редактора или область префиксных ex-команд - операций редактора над группой строк файла или над целым файлом (команды вводятся на приглашение ":", отображаются на экране и исполняются по команде управляющей клавиши перевода строки).

Команды используются для многострочных операций с редактируемым файлом и настройки сеанса редактирования:

:w файл - запись текущего буфера в файл;

:m,nw файл - запись строк с m-ой по n-ую в файл;

:e! - отмена всех изменений в буфере с перезагрузкой в него файла с диска;

:e файл - загрузка файла в буфер редактирования, файл замещает старое содержимое самого буфера;

:r файл - добавление содержимого файла после текущего положения курсора;

:set опция - значение опций определяет настройку редактора и пр.

#### Практическая работа №6(1) Изучение интерфейса редактора vi

##### Цель работы:

1. Изучить основные возможности встроенного текстового редактора vi - наиболее распространенного средства для построения текстовых файлов, исходных текстов программ и shell-процедур.

##### Порядок выполнения работы

1. Зарегистрируйтесь в ОС Linux. Запустите на выполнение программу Терминал.
2. Войдите в редактор с созданием нового пустого файла с произвольным именем и расширением - .1.
3. Изучите интерфейс редактора vi. Как перейти в командный режим редактора? В текстовый режим редактора? Какие операции с текстом позволяет выполнить редактор?
4. Поместите в созданный файл текст, включающий не менее четырех строк с несколькими словами в каждой.
5. Вставьте по одной пустой строке до и после одной из строк файла.
6. Заполните пустые строки произвольным текстом.
7. Вставьте еще по одной строке в середину текста файла без предварительного резервирования пустых строк.
8. Перейдите в режим редактирования и выполните произвольное редактирование отдельных слов и строк файла с использованием всех команд из групп "Команды изменения текста" и "Команды отмены произведенных изменений в текущей строке".
9. Выйдите из редактора с сохранением файла. Убедитесь в сохранении созданного файла.

#### Практическая работа №6(2) Создание и редактирование текста с помощью редактора vi

##### Цель работы:

1. Изучить основные возможности встроенного текстового редактора vi.
2. Научиться редактировать текстовые файлы средствами редактора vi.

##### Порядок выполнения работы

Изучаются команды:

* сохранение файлов;
* ввода текста;
* удаления фрагментов текста;
* копирования фрагментов текста в буферную область памяти;
* вставки содержимого буфера в текст файла;
* редактирования (изменения) текста;
* поиска строк файлов по их фрагментам;
* многострочных операций с файлом (префиксные команды)

1. Зарегистрируйтесь в ОС Linux. Запустите на выполнение программу Терминал.
2. Войдите опять в vi для редактирования созданного файла с использованием команд из групп "Команды копирования в буфер" и "Команды вставки буфера в текст".
3. Поменяйте местами несколько слов в строках файла.
4. Поменяйте местами несколько строк.
5. Поменяйте местами последовательно начало строки с её концом и наоборот.
6. Выполните операции поиска строк файла по заданным их фрагментам с различными направлениями поиска ( см. "Команды поиска строки файла по фрагменту её текста").
7. Не выходя из vi, перепишите полученную в результате редактирования версию файла в файл с тем же именем, но с расширением .new.
8. Добавьте первые 3 строки редактируемого файла к файлу .1.
9. Не выходя из vi, загрузите в буфер файл с расширением .1.
10. Последовательно удалите части строк и несколько строк с использованием команд из группы "Команды удаления текста".
11. Выйдите из редактора без сохранения файла.
12. Просмотрите и проанализируйте содержимое редактируемых файлов в текущем каталоге.

### *Контрольные вопросы:*

* 1. В чем особенности и преимущества встроенного редактора vi ОС UNIX?
  2. Какие два основных режима работы использует редактор? Как осуществляется переключение режимов?
  3. Какую структуру имеет экран при редактировании файла? Назначение полей экрана?
  4. Как в vi организовано редактирование открытого в нем файла?
  5. Как организована работа с клавишными командами редактора?
  6. Что такое "префиксные команды" и их назначение? Как организована работа с командами этого вида?
  7. Какие функции редактора Вы использовали при выполнении лабораторной работы?

# Тема 2.5. Изучение сетевых команд shell

## Сетевые команды Shell

Персональный компьютер на вашем рабочем месте может быть подключен к локальной сети. Более того, локальные сети становятся реальностью и в домашних условиях. Рассмотрим работу с одной из сетевых команд ОС Linux – утилитой netstat.

Утилита netstat выводит информацию о локальной сети и средствах TCP/IP. Именно к ней чаще всего обращаются администраторы, чтобы быстро отыскать причину неисправности в сети TCP/IP. Содержание и форма выходной информации зависят от операционной системы, но обычно выводятся следующие данные: список соединений, статистика сетевых интерфейсов, информация по буферам данных, содержание таблицы маршрутизации, статистика работы протокола. Характер выводимой информации можно выбирать с помощью опции командной строки.

*Основные ключи утилиты:*

при выведении параметров утилиты на экран используйте команду | more для постраничного вывода.

|  |  |
| --- | --- |
| **Ключ** | **Функция** |
| -r route | вывод таблицы маршрутизации |
| -i interfaces | вывод статистики сетевых интерфейсов |
| -s statistics | Фвывод статистики передачи данных (по протоколу SNMP) |
| -n numeric | имена портов в числовом виде |
| -N symbolic | имена портов в символическом виде |
| -l listening | вывод состояния портов, находящиеся в режиме ожидания |
| -a all | вывод состояния всех портов |
| Активные соединения через порты: |  |
| -st | TCP |
| -u | UDP |
| -W | RAW |
| -X | UNIX |

### *Список соединений.*

Команда netstat обладает набором ключей для отображения портов, находящихся в активном и/или пассивном состоянии. Таким образом, можно получить список всех серверных приложений, работающих на данном компьютере.

Информация выводится столбцами. В первом из них указан протокол, затем размеры очередей приема и передачи для установленного соединения на данной машине (на другом конце соединения размеры очередей могут быть другими), локальный и удаленный адреса и текущее состояние соединения.

#### *Пример:*

st1@pds:~ > netstat -ta

Proto Recv-Q Send-Q Local Address Foreign Address State

tcp 0 2 pds.sut.ru:telnet gerasim.pds.sut.ru:1288 ESTABLISHED

tcp 1 0 pds.sut.ru:4550 pds.sut.ru:3128 CLOSE\_WAIT

tcp 1 0 pds.sut.ru:4548 pds.sut.ru:3128 CLOSE\_WAIT

tcp 0 0 gw.pds.sut.:netbios-ssn marya.pds.sut.ru:1027 ESTABLISHED

tcp 0 0 gw.pds.sut.:netbios-ssn yanko.pds.sut.ru:1104 ESTABLISHED

tcp 0 0 gw.pds.sut.:netbios-ssn mumu.pds.sut.ru:1065 ESTABLISHED

tcp 0 0 \*:6000 \*:\* LISTEN

tcp 0 0 \*:3128 \*:\* LISTEN

tcp 0 0 \*:53333 \*:\* LISTEN

tcp 0 0 \*:389 \*:\* LISTEN

tcp 0 0 localhost:1032 localhost:1033 ESTABLISHED

tcp 0 0 \*:netbios-ssn \*:\* LISTEN

tcp 0 0 \*:smtp \*:\* LISTEN

tcp 0 0 \*:imap2 \*:\* LISTEN

tcp 0 0 \*:pop3 \*:\* LISTEN

tcp 0 0 \*:login \*:\* LISTEN

tcp 0 0 \*:shell \*:\* LISTEN

tcp 0 0 \*:8000 \*:\* LISTEN

tcp 0 0 \*:telnet \*:\* LISTEN

tcp 0 0 \*:ftp \*:\* LISTEN

tcp 0 0 \*:time \*:\* LISTEN

tcp 0 0 \*:www \*:\* LISTEN

tcp 0 0 \*:2049 \*:\* LISTEN

tcp 0 0 \*:832 \*:\* LISTEN

--More—

Как видно из примера, большинство серверов находится в режиме ожидания запроса на соединение (LISTEN). В первой строке отражено соединение (ESTABLISHED) через telnet с машиной gerasim.pds.sut.ru. Состояние CLOSE\_WAIT означает, что соединение разорвано, но переключение в состояние LISTEN еще не произошло; TIME\_WAIT – что соединение ожидает разрыва. Если соединение находится в состоянии SYN\_SENT, то это означает наличие процесса, который пытается установит соединение с несуществующим сервером. Состояние соединения имеет значение только для протокола TCP. Протокол UDP факта установления соединения не проверяет.

### *Содержание таблицы маршрутизации.*

Каждое соединение машины с сетью называется сетевым интерфейсом. Машина, имеющая более одного интерфейса, может принимать данные по одному интерфейсу и передавать их по другому, таким образом осуществляя пересылку данных между сетями. Эта функция называется маршрутизацией, а машина, выполняющая ее – шлюзом.

Данные маршрутизации хранятся в одной из таблиц ядра. Для направления пакета по конкретному адресу ядро подбирает наиболее подходящий маршрут. Если такой маршрут отсутствует и нет маршрута по умолчанию, то отправителю возвращается сообщение об ошибке.

Команда netstat –r позволяет отображать таблицу маршрутизации.

Пункты назначения и шлюзы могут показываться или именами машин, или их IP-адресами. Флаги дают оценку маршрута.

#### *Пример:*

st1@pds:~ > netstat -r

Kernel IP routing table

Destination Gateway Genmask Flags Ifac

pds.sut.ru \* 255.255.255.255 UH eth1

195.19.219.120 \* 255.255.255.248 U eth0

195.19.219.128 \* 255.255.255.192 U eth1

192.168.1.0 \* 255.255.255.0 U eth0

195.19.221.0 lgw.ccs.sut.ru 255.255.255.0 UG eth1

193.125.0.0 lgw.ccs.sut.ru 255.255.0.0 UG eth1

loopback \* 255.0.0.0 U lo

default lgw.ccs.sut.ru 0.0.0.0 UG eth1

|  |  |
| --- | --- |
| **Название столбца** | **Расшифровка** |
| Gatway | Имена используемых шлюзов |
| Genmask | Маска, используемая для отображения общей части адреса, соответствующего данному маршруту |
| Flags | Флаги, описывающие маршрут: G Маршрут использует шлюз |
|  | U Интерфейс активен, может использоваться для передачи данных |
|  | H Данные можно передавать только одному узлу |
|  | D Запись создана перенаправляющим сообщением протокола ICMP |
|  | M Запись модифицирована перенаправляющим сообщением протокола ICMP |
| Iface | Интерфейс, используемый для передачи пакетов |

### *Статистика сетевых интерфейсов.*

При использовании ключа –i команды netstat на экран будут выведены статистические данные всех используемых интерфейсов. Исходя из них, можно выяснить, исправно ли соединение с сетью.

#### *Пример:*

st1@pds:~ > netstat -i

Kernel Interface table

Iface MTU Met RX-OK RX-ERR RX-DRP RX-OVR TX-OK TX-ERR TX-DRP TX-OVR Flg

eth0 1000 0 844904 0 17 0 1454454 5 0 0 BRU

eth0: 1000 0 - no statistics available - BRU

eth1 1500 0 590844 0 7 0 434438 59 0 0 BRU

lo 3924 0 45754 0 0 0 45754 0 0 0 LRU

|  |  |
| --- | --- |
| **Название столбца** | **Расшифровка** |
| Gatway | Имена используемых шлюзов |
| Genmask | Маска, используемая для отображения общей части адреса, соответствующего данному маршруту |

Ошибки являются следствием проблем в кабельной системе. В нормально работающей сети количество конфликтов (RX-OVR, TX-OVR) не должно превышать 3% от числа пакетов, а другие ошибки не должны составлять более 0,5% от общего числа пакетов.

### *Статистика передачи данных.*

Команда netstat –s выдает содержимое счетчиков сетевых программ. В выходной информации есть разделы, относящиеся к различным протоколам: IP, ICMP, TCP, UDP. С ее помощью можно определить место появления ошибки в принятом пакете.

#### *Пример:*

st1@pds:~ > netstat -s

Ip:

179495 total packets received 13 with invalid headers 8753 forwarded

0 incoming packets discarded

168812 incoming packets delivered

325599 requests sent out

544 fragments failed

Icmp: 728 ICMP messages received

3 input ICMP message failed

ICMP input histogram:

destination unreachable: 82

timeout in transit: 55

source quenchs: 9

echo requests: 582

1235 ICMP messages sent

0 ICMP messages failed

ICMP output histogram:

destination unreachable: 646

time exceeded: 6

redirect: 1

echo replies: 582

Tcp:

2428 active connections openings

0 passive connection openings

0 failed connection attempts

0 connection resets received

17 connections established

154840 segments received

318758 segments send out

1480 segments retransmited

99 bad segments received.

499 resets sent

Udp:

13397 packets received

73 packets to unknown port received.

12 packet receive errors

5608 packets sent

TcpExt:

15 resets received for embryonic SYN\_RECV sockets

### *Утилита netconf*

Настройки локальной сети можно производить и с помощью специальных утилит **netconf** или **netcfg**, которые являются просто составной частью пакета **linuxconf**.Первая из них работает в графическом режиме, а вторая — в текстовом.

Надо иметь в виду, что многие опытные пользователи Linux критически относятся к возможностям пакета **linuxconf** и предпочитают прямое редактирование конфигурационных файлов. Но для новичка эти утилиты могут оказаться удобнее.

#### Практическая работа №7(1) Утилита netstat: список соединений, открытых на сервере

##### Цель работы:

1. С помощью утилиты netstat исследовать состояние локальной IP-сети.
2. Научиться получать список соединений, открытых на сервере.

##### Порядок выполнения работы

1. Зарегистрируйтесь в ОС Linux. Запустите на выполнение программу Терминал.
2. С помощью утилиты netstat получить список соединений, открытых на сервере. Прокомментировать соединения, находящиеся в режиме ESTABLISHED (внутренний или внешний интерфейс, с каким узлом, по какому протоколу).

#### Практическая работа №7(2) Утилита netstat: таблица маршрутизации

##### Цель работы:

1. С помощью утилиты netstat исследовать состояние локальной IP-сети.
2. Научиться анализировать таблицу маршрутизации.

##### Порядок выполнения работы

1. Зарегистрируйтесь в ОС Linux. Запустите на выполнение программу Терминал.
2. Получить таблицу маршрутизации. Указать, через какие интерфейсы с какими сетями происходит связь, имя шлюза, маски локальных сетей.

#### Практическая работа №7(3) Утилита netstat статистика сетевых интерфейсов, статистика передачи данных

##### Цель работы:

1. С помощью утилиты netstat исследовать состояние локальной IP-сети.
2. Научиться анализировать статистику сетевых интерфейсов и статистику передачи данных.

##### Порядок выполнения работы

1. Зарегистрируйтесь в ОС Linux. Запустите на выполнение программу Терминал.
2. Получить статистику сетевых интерфейсов. Построить графики статистической информации для внутренней (eth0) и внешней (eth1) локальных сетей, объяснить значение установленных флагов. Сравнить количество ошибок с требованиями, предъявляемыми к работе сетей; сделать вывод о работе локальной сети.
3. Получить статистику сетевых интерфейсов. Проанализировать работу каждого из протоколов. Для протокола ICMP построить графики входной и выходной гистограмм.

### *Контрольные вопросы:*

* 1. Каково назначение утилиты netstat?
  2. Какие ключи утилиты netstat позволяют отображать порты, находящиеся в активно и/или пассивном состоянии?
  3. В каких состояниях может находиться сервер в системе?
  4. Какой ключ утилиты netstat позволяет отображать таблицу маршрутизации?
  5. Какую информацию отражает таблица маршрутизации?
  6. Как с использованием утилиты netstat вывести статические данные всех используемых интерфейсов? Какую информацию позволяют выяснить эти данные?
  7. Как вывести на экран содержимое счетчиков сетевых протоколов? Что позволяет выявить данная информация?
  8. Какие утилиты, позволяют производить настройку локальной сети?

# Раздел 3. Сервисные и прикладные программы ASP Linux

# 

# Тема 3.1. Архивация данных

## 

## 3.1.1. Архиваторы ASP Linux

*Архив* - это файл, заключающий в себе содержимое многих файлов. Кроме того, архив идентифицирует имена файлов, их хозяев и т.д. (в архивах также записываются разрешение доступа, пользователь и группа, размер в байтах и время последней модификации. В некоторых архивах еще указываются имена файлов в каждом заархивированном каталоге и информация о каталогах).

Файлы, содержащиеся в архиве, называются *членами архива*.

Основным средством архивирования в UNIX (а, следовательно, и в Linux) является комплекс из двух программ — tar и gzip. Хотя никто не запрещает пользоваться arj, pkzip, lha, rar и т. д. — версии этих программ для Linux общедоступны. Просто уж исторически сложилось, что пользователи Unix чаще применяют именно  tar и gzip, и именно в таком формате распространяется большая часть программного обеспечения для Unix.

"Зачем использовать две программы архивирования?” Все дело в том, что tar расшифровывается как Tape ARchiver, он не сжимает данные, а лишь объединяет их в единый файл с последовательным доступом для последующей записи на ленту. По умолчанию этот архивный файл создается на ленточном накопителе, точнее на устройстве /dev/rmt0. Если вы хотите создать архивный файл на диске, то необходимо использовать команду tar с опцией f, после которой указывается имя архивного файла.

Хотя программа tar создает архивы, она, как было сказано, не сжимает архивы, а просто соединяет отдельные файлы в единый архивный файл. Для сжатия этого файла часто применяют команду gzip.

Поскольку программа gzip не умеет сохранять в одном архиве несколько файлов, то обычно ее применяют для сжатия архивов, созданных программой tar.

В последнее время все чаще вместо программы gzip используется архиватор bzip2, который обеспечивает более высокую степень сжатия и работает несколько быстрее. Команда bzip2 обычно не устанавливается автоматически при инсталляции Linux. Но она имеется на дистрибутивном диске в виде rpm-пакета и ее легко установить.

Работает bzip2 примерно так же, как команда gzip, т. е. замещает каждый файл, имя которого задано в командной строке, сжатой версией, добавляя к имени файла суффикс .bz2.

Сжатый файл имеет то же самое время модификации, права доступа и, по возможности, того же владельца, что и исходный файл, что дает возможность восстановить эти атрибуты при извлечении файлов из архива.

В некоторых случаях сжатый файл может оказаться даже больше по размеру, чем исходный. Это происходит, например, для файлов длиной менее 100 байт, потому что механизм сжатия использует заголовок длиной около 50 байт. Для файлов, представляющих собой случайную последовательность символов (в том числе для выходных файлов большинства файловых архиваторов) длина файла увеличивается примерно на 0.5%.

### *Утилита ark*

Архиваторы tar, zip, gzip, bzip2, lha изначально были предназначены для работы через интерфейс командной строки. В настоящее время существуют программы, представляющее собой графические оболочки для упаковщиков и архиваторов. Они не подменяют их собой, а лишь обеспечивают более удобный (интуитивно понятный) интерфейс работы с архивами.

В KDE имеется утилита ark, позволяющая работать с архивами tar, zip, gzip, bzip2, lha. Она входит в состав пакета kdeutils-2.2.1-1.asp, ее внешний вид представлен на рис. 1.

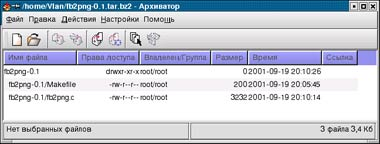


Рис.3.1.1.

Ark прост в использовании, имеет русифицированное меню. Кроме них можно пользоваться программами LinZip, Karchiveur и RAR для Linux.

#### Практическая работа №8(1). Создании архивов с использованием программ архивации tar, gzip, bzip, bzip2

##### Цель работы:

* + 1. Изучить основные возможности архиваторов Linux
    2. Научиться работать с архиваторами tar, gzip, bzip, bzip2.

##### Порядок выполнения работы

1. Зарегистрируйтесь в ОС Linux. Запустите на выполнение программу Терминал.
2. Создайте в своем домашнем каталоге папку «Архиваторы»
3. В графической среде Linux осуществите поиск файлов в корневом каталоге с расширением txt. Найденные файлы поместите в созданную ранее папку
4. Упакуйте папку «Архиваторы» в архив tar. Для этого в программе Konqueror выделите нужную папку, вызовите контактное меню и выберите в нем пункт «Упаковать» и выберите соответствующий пункт упаковки. Просмотрите свойства созданного архива. Определите его объем.
5. Созданный архив упакуйте, используя методы упаковки: gzip и bzip2. Просмотрите свойства созданных архивов. Определите их объем.
6. Упакуйте папку «Архиваторы» в zip архив. Просмотрите свойства созданного архива. Определите его объем.
7. Упакуйте папку «Архиваторы» методами tar-gzip и tar-bzip2. Просмотрите свойства созданных архивов. Определите их объем.
8. Проанализируйте различные методы упаковки архивов на основании полученных данных. Какой метод упаковки плотнее?
9. Завершить сеанс работы с системой.

#### Практическая работа №8(2) Менеджер архивов ark

##### Цель работы:

* 1. Познакомиться с работой менеджера архивов ark
  2. Научится работать с менеджером архивов: просматривать архивы, добавлять информацию в архивы, удалять файлы из архива.

##### Порядок выполнения работы

1. Зарегистрируйтесь в ОС Linux. Запустите на выполнение программу Терминал.
2. Запустите менеджер архивов KDE (Стандартные/Ark). Для каких целей предназначена данная программа? С какими форматами архивов она работает?
3. Откройте один из созданных ранее в практической работе 8(1) архивов. Какие файлы содержит архив? Какие операции можно осуществлять с архивом, используя программу atk?
4. Просмотрите содержимое любого файла из архива. Каковы результаты просмотра?
5. Удалите любой из файлов архива. Какие способы удаления файлов из архива Вы знаете?
6. Осуществите распаковку файлов из архива в любую созданную Вами в личном каталоге директорию.
7. Откройте любой другой архив. Осуществите выборочную распаковку файлов из архива в директорию Вашего личного каталога.
8. Добавьте в архив созданный Вами файл.
9. Распакуйте весь архив в новую папку в домашнем каталоге.
10. Удалите файлы с которыми Вы работали по завершении оформления отчета.
11. Завершите сеанс работы с ОС Linux.

### *Контрольные вопросы:*

1. Объясните понятие архива. Какие форматы архивов Linux Вы знаете?
2. Что означает понятие член архива?
3. Есть ли различие во времени модификации и правах доступа сжатого и исходного файлов?
4. Существуют ли случаи, когда сжатый файл оказывается больше по размеру, чем исходный? Если да, то почему такое возможно?
5. Для каких целей предназначена программа tar?
6. Для каких целей предназначена gzip?
7. Как работает программа bzip2? Какова степень сжатия архивов данной программой?
8. Для каких целей предназначен менеджер архивов ark? Какие форматы архивов он поддерживает?

# Тема 3.2. Знакомство с текстовым редактором ASP Linux пакта

# OpenOffice

## 

## 3.2.1. Работа с текстовым редактором OpenOffice.org Writer:

## основные принципы работы

### *Текстовые процессоры Linux*

Текстовых процессоров для Linux существует множество. Наиболее известны из них StarWriter из пакета StarOffice фирмы Sun и процессор Word Perfect 8. Аналог последнего, разработанный для ОС Windows, долгое время на равных конкурировал с пакетом MS Word, что, конечно, говорит о его высоком качестве.

Особое место среди текстовых процессоров занимает издательская система TEX и основанные на ней продукты типа Lyx. TEX - это скорее язык программирования, чем текстовый редактор? простому пользователю, которому редактор нужен для написания деловых писем или диссертации, вряд ли стоит браться за его изучение.

13 октября 2000 года Sun открыла исходные коды пакета StarOffice. Тем самым появился проект OpenOffice.org.

Исходные коды OpenOffice.org основаны на технологии, которая первоначально была разработана Sun Microsystems для будущих версий пакета StarOffice(TM). В отличие от StarOffice проект OpenOffice.org не содержит интегрирующей оболочки - архитектура пакета предполагает отдельное использование входящих в пакет приложений. Пакет написан на языке C++, и включает в себя все основные офисные приложения, такие как текстовый процессор, электронную таблицу, программу управления презентациями, программу для работы с графикой, средства создания диаграмм и редактирования формул. Но OpenOffice.org не включает в себя клиента электронной почты, календаря и браузера.

Среди новшеств можно отметить использование формата файлов, основанного на стандарте XML, что облегчает взаимодействие с другими продуктами (и, следовательно, с другими пользователями компьютеров), а также обеспечивает ясные перспективы развития продукта в будущем. В состав продукта включены конверторы для других распространенных файловых форматов, включая форматы всех версий Microsoft Office, и улучшена поддержка азиатских языков (китайского, японского, корейского).

В настоящее время локализация пакета OpenOffice.org выполнена для 27 языков. Пакет стабильно работает в операционных системах Solaris, Linux (включая PPC Linux) и Windows. Перенос его на платформы FreeBSD, IRIX и Mac OS X находится в стадии завершения.

Впервые появилась возможность под Linux работать с файлами, созданными в Microsoft Office, то есть нет необходимости запускать эмуляторы для того, чтобы просмотреть файл формата MS Word , полученный как вложение по электронной почте от любого из приверженцев Microsoft.

### *Окно OpenOffice.org Writer*

Внешний вид окна программы несколько непривычен для пользователя MS Office. Окно OpenOffice.org Writer содержит следующие элементы:

* *Заголовок окна*, содержащий три кнопки управления приложением, названия приложения и активного документа;
* *Строку меню,* позволяющую выбрать и выполнить любую команду редактора;
* Горизонтальные панели инструментов, содержащие кнопки для выполнения основных команд и настроек;
* Вертикальная панель инструментов (слева);
* *Горизонтальную линейку* для установки отступов от границ страницы;
* *Линии прокрутки* (горизонтальная полоса прокрутки содержит четыре кнопки выбора режима отображения);
* *Строку состояния*, отображающую координаты курсора и номера страницы и раздела).

### *Основные элементы документа:*

К основным **объектам текста** относятся следующие элементы:

* **Символ** ‑ минимальный элемент текста. Внешне воспринимается как отдельный знак.
* **Абзац** – оформленный определенным образом фрагмент текста. Выделение абзаца диктуется содержанием текста. Абзац заканчивается нажатием клавиши Enter. Конец абзаца отмечается символом ¶.
* **Колонтитул** – область страницы, в которой размещается справочный текст. Обычно ‑ это номер страницы, название документа, раздела или главы. Различают **Верхний** и **Нижний** колонтитулы.
* **Сноска** – пояснение к отдельному слову или предложению, размещается вне основного текста внизу страницы или в конце документа.
* **Примечание** – комментарий к слову или фрагменту текста, появляющийся в виде всплывающих подсказок.
* **Страница** – фрагмент документа, размещаемый при печати на отдельной странице. Содержит текст (возможно с таблицами и рисунками) и колонтитулы.
* **Список** – фрагмент текста, содержащий нумерованные или отмеченные маркерами строки.

**Стиль –** набор способов оформления абзацев для различных элементов текста (заголовков, названий, колонтитулов и др.) и символов.

#### Практическая работа №9(1) Работа с текстовым редактором OpenOffice.org Writer: создание итогового отчета

##### Цель работы:

* 1. Ознакомиться с возможностями и основными объектами приложения OpenOffice.org Writer
  2. Освоить приемы форматирования символов и абзацев.
  3. Оформить титульный лист отчета по практике.

##### Порядок выполнения работы

* + 1. Зарегистрируйтесь в ОС Linux. Запустите на выполнение OpenOffice.org Writer и найдите основные элементы окна, ответьте на контрольные вопросы.
    2. Создайте новый документ. Установите параметры страницы:
* Верхнее поле — 2 см,
* Нижнее поле — 1,5 см
* Правое поле — 1,5 см,
* Левое поле — 2,5 см
  + 1. Наберите текст своего титульного листа в виде 8 строк текста (выравнивание по правому краю, шрифт Helvetica, кегль 12) в соответствии с образцом:

ГОУ СПО АСТРАХАНСКИЙ КОЛЛЕДЖ ВЫЧМСЛИТЕЛЬНОЙ ТЕХНИКИ

Специальность 2206

ОТЧЕТ

о выполнении практических работ по учебной практике

«Операционные системы и среды»

Выполнил: студент гр. ИК-21 н.д. Иванов В.Г.

Проверил: Прошунина Л.М.

2005

* + 1. Сделайте видимыми границы текста.

Отформатируйте текст правильно, не вставляя лишних пустых строк, задавая параметры:

- для первой, третей, четвертой, пятой строк выравнивание по центру;

- для первой и второй строки: начертание Полужирный, кегль 12;

- для второй строки: интервал перед абзацем 18; выравнивание по правому краю

- для третей строки: кегль 18, интервал перед абзацем 250, начертание Полужирный

- для четвертой и пятой строки: кегль 16,

- для шестой строки: кегль 14, интервал перед абзацем 90,

- для седьмой строки: кегль 14, интервал перед абзацем Авто,

- выделив шестую и седьмую строки, сдвиньте их с помощью клавиши табуляции [Таб] вправо, с помощью этой же клавиши выровняйте начало текстов после двоеточия;

- для последней строки: кегль 14, интервал перед абзацем 250, выравнивание по центру, начертание Полужирный.

Пример отформатированного титульного листа приведен в Приложении 1.

* + 1. Сохраните текст.

#### Практическая работа №9(2) Работа с текстовым редактором OpenOffice.org Writer: создание итогового отчета

##### Цель работы:

* + - 1. Ознакомиться с возможностями и основными объектами приложения OpenOffice.org Writer
      2. Освоить приемы форматирования символов и абзацев.
      3. Оформить титульный лист отчета по практике.

##### Порядок выполнения работы

* + - * 1. Зарегистрируйтесь в ОС Linux. Запустите на выполнение OpenOffice.org Writer и найдите основные элементы окна, ответьте на контрольные вопросы.
        2. Откройте созданный на практической работе 9(1) титульный лист итогового отчета.
        3. Продолжив документ на следующей странице оформите отчет по практике. В отчете должны содержаться следующие разделы:

1. Цели и задачи практики;

2. База практики;

3. Содержание практики;

4. Рабочий этап практики;

5. Отзыв руководителя практики;

6. Приложения.

### *Контрольные вопросы:*

1. Какие виды работ позволяет выполнить текстовый редактор OpenOffice.org Writer?
2. Найдите в тексте описания практической работы абзацы, заголовки, списки.
3. В каких единицах представлен размер шрифта в OpenOffice.org Writer?
4. Что такое колонтитул, сноска, примечание?
5. Что представляют собой Панели инструментов?
6. Как выделить фрагмент текста в OpenOffice.org Writer?
7. Как выделить строку и несколько строк в OpenOffice.org Writer?
8. Как следует набирать абзац?
9. Перечислите основные параметры символа.
10. Перечислите основные параметры абзаца.
11. Перечислите основные параметры страницы.
12. Как выделяются при наборе знаки препинания?
13. Как настроить автоматическую расстановку переносов?
14. Какое расширение имеют файлы документов, подготовленных с помощью OpenOffice.org Writer?
15. Как вводятся в текст знаки, отсутствующие на клавиатуре?
16. В каких случаях удобно пользоваться автозаменой?
17. Как настроить Word на выполнение автозамены?
18. Перечислите виды работ по созданию и форматированию текста, которые Вы освоили в данной лабораторной работе.

# Тема 3.3. Итоговое зачетное занятие

Завершением практики является сдача отчета преподавателю и выполнение индивидуального задания.

Описание рабочего этапа практики должно содержать следующие подразделы:

1. Задание

2. Выполнение задания

## Итоговое задание по практике

**Вариант 1**

1. Дайте ответ на поставленный вопрос:

Какие системные имена каталогов Вам известны? Объясните назначение и содержание каждого поля каталога. Какую информацию содержит «пустой» вновь созданный каталог?

1. Напишите команду shell, выполняющую указанные действия:

А) поиск файлов в корневом каталоге типа директория

Б) переименование файла temp в tmp

В) удаление в каталоге /home/linux/user06/I-21 всех файлов, имена которых заканчиваются на однозначную цифру

Г) создание на файл Alpha символьной ссылки с именем SALP в корневом каталоге

Д) просмотр содержимого текущего каталога с выводом скрытых файлов

Е) просмотр всех активных процессов в системе

Ж) запуск процесса yes в фоновом режиме

З) установить права доступа для файла liter: для владельца – все права, для группы владельца и остальных пользователей – только чтение

1. Расшифруйте выданное в результате выполнения команды сообщение:

ls –l grt

-rwxr-xr-- 2 root root 105 10 Ноября 2005 grt

**Вариант 2**

1. Дайте ответ на поставленный вопрос:

Как создать несколько ссылок с совпадающими именами на несколько файлов в другом каталоге? Как отражается на содержимом связанных файлов изменение содержания одного из них и почему? На какой атрибут и как влияет удаление одного из связанных файлов?

1. Напишите команду shell, выполняющую указанные действия:

А) поиск в текущем каталоге всех файлов, размер которых более 10 байт

Б) переход в вышележащий каталог

В) копирование из каталога /home/linux/user06/I-21 всех файлов, с расширением txt в каталог /home/linux/user06/ tmp

Г) создание в текущем каталоге жесткой ссылки с именем ssl на файл /home/linux/user06/tmp/jaf

Д) просмотр содержимого текущего каталога с выводом инодов файлов

Е) просмотр информации обо всех запущенных процессах в системе в длинном формате

Ж) убить процесс с идентификатором 1090

З) запретить следующие права доступа к файлу: liter: выполнение - для группы владельца и все права - для остальных пользователей

1. Расшифруйте выданное в результате выполнения команды сообщение:

ls –l grt

drwxr-x--- 1 user users 10 5 Ноября 2005 grt

# Вариант 3

1. Дайте ответ на поставленный вопрос:

Объясните понятие архива. Какие форматы архивов Linux Вы знаете? Для каких целей предназначена программа tar, gzip, bzip2?

1. Напишите команду shell, выполняющую указанные действия:

А) поиск в текущем каталоге всех файлов, на которые имеется более одной ссылки

Б) создать текстовый файл ret в текущем каталоге

В) создать каталог yY в текущем каталоге

Г) создание в текущем каталоге жесткой ссылки с именем yE на файл yY

Д) просмотр содержимого текущего каталога с выводом специальных символов в конце файла

Е) перевести работу с номером 2 с переднего плана работы в фоновой режим

Ж) запустить процесс поиска всех директорий в текущем каталоге с приоритетом, численное значение которого увеличено на 5 пунктов

З) добавить следующие права доступа к файлу: liter: чтение и выполнение - для группы владельца и чтение - для остальных пользователей

1. Расшифруйте выданное в результате выполнения команды сообщение:

ls –l grt

lrwxr-x--- 3 user users 5 5 Ноября 2005 grt ->jj

**Вариант 4**

1. Дайте ответ на поставленный вопрос:

Кто может пользоваться и изменять права доступа к файлам? Как кодируются в атрибутах файла и каталога права доступа?

1. Напишите команду shell, выполняющую указанные действия:

*А)* поиск в корневом каталоге всех файлов, размер которых менее 50 байт

Б) справку по команде rm перенаправить в текстовый файл spr, расположенный в текущем каталоге

В) определить текущий каталог

Г) создание в текущем каталоге символьную ссылку с именем DDD на файл spr

Д) просмотреть содержимого текущего каталога с выводом информации о файлах в длинном формате

Е) запустить процесс просмотра содержимого файла spr с приоритетом, увеличенным на 10 пунктов

Ж) просмотреть список запущенных процессов

З) для файла liter: установить права доступа для владельца – все права, для группы владельца – только чтение, убрать право записи для остальных пользователей

1. Расшифруйте выданное в результате выполнения команды сообщение:

$ps -flu lev

PID PPID TTY PRI TIME CMD

927 1 tty5 0:04 sh

1001 927 tty5 0:02 ps

# Вариант 5

1. Дайте ответ на поставленный вопрос:

В чем особенности и преимущества встроенного редактора vi ОС UNIX? Какие два основных режима работы использует редактор? Как осуществляется переключение режимов?

1. Напишите команду shell, выполняющую указанные действия:

*А)* запустить процесс просмотра содержимого файла ter в фоновом режиме

Б) объединить два текстовых файла text1 и text2 в текстовом файле ter

В) обновить временные характеристики файла text1

Г) создание в текущем каталоге жесткой ссылки с именем DDD на файл ter

Д) просмотреть содержимого текущего каталога с выводом информации об уникальных именах файлов

Е) поиск в домашнем каталоге всех файлов с расширением txt

Ж) приостановить работу процесса

З) для файла liter установить права доступа для владельца – чтение и запись, для группы владельца и остальных пользователей отменить все права

1. Расшифруйте выданное в результате выполнения команды сообщение:

$ps

PID TTY TIME CMD

194 tty5 0:02 grep

200 tty5 0:01 ps

Приложение 1

**ГОУ СПО АСТРАХАНСКИЙ КОЛЛЕДЖ ВЫЧМСЛИТЕЛЬНОЙ ТЕХНИКИ**

**Специальность 090108**

**ОТЧЕТ**

о выполнении практических работ по учебной практике

«Операционные системы и среды»

Выполнил: студент гр. ИК-21 н.д. Иванов В.Г.

Проверил: Прошунина Л.М.

2005

# Литература

1. Болл Билл “Линукс за 24 часа”, изд. Дом “Вильямс”, Ю, С-Пб, 1999
2. Браун С. Операционная система Unix – М.: Мир, 1986
3. Карлинг М., Деглер С., Деннис Дж. Системное администрирование Linux.: Пер. с англ.: Уч. пос. – М.: Издательский дом "Вильямс", 2000.
4. Кирх О. Linux для профессионалов. Руководство администратора сети. – СПб: Издательство "Питер", 2000.
5. Шевель А. “LINUX Обработка текстов. Специальный справочник”. – СПб. : Питер, 2001
6. Зубков С.В. «Linux» – М.: ДМК Пресс,2000 г.
7. В.Костромин, "Linux для пользователя" – СПб: БХВ-Петербург, 2002 г.

# Технические средства обучения

Практика проводится в учебном вычислительном центре (УВЦ) Астраханского колледжа вычислительной техники в компьютерном классе № 232.

**Аппаратные средства:**

* Процессор Celeron 1700МГц
* OЗУ DIMM 256 Mbt SDRAM
* Дисковод 3,5 “ 1,44 Mbt
* Жесткий диск Futjitsu 10 Gbt
* Сеть Fast Ethernet PCI 100 Mbt/с
* Hub: Gompex ReadyLink RE 100 TX
* Клавиатура 105 – кнопочная
* Мышь PS/2 2-х кнопочная
* Монитор Samsung 15“ 0,28, Sync Master 550S, MPR 11

**Программные средства:**

* Операционная система ASP Linux 9.0
* Офисный пакет OpenOffice.org