Введение

Операционная система Linux – сетевая операционная система, предназначенная для решения задач вычислительного характера, администрирования серверов, управления специализированными компьютерными системами, управляющими сложным техническим и физическим оборудованием, а также для выполнения задач пользователя.

Операционная система Linux предназначена для персональных компьютеров и рабочих станций. Она представляет собой версию ОС UNIX и может обеспечивать работу в многопользовательском и многозадачном режимах, система является стабильной и настраиваемой, имеет высокую скорость и гибкость в работе, полноценный графический интерфейс, не требовательна к потребляемым ресурсам. Большинство дистрибутивов является бесплатным, но защищено лицензией, его исходные коды доступны в Интернете.

ОС Linux в последние годы интенсивно развивается, обновляются ядра ОС, совершенствуется командный интерпретатор, разработаны многие приложения для пользователей. ОС Linux представляет собой версию ОС UNIX, предназначенную для персональных компьютеров и имеет с ней много общего.

ОС Linux в большинстве дистрибутивов имеет модульное построение, высокую гибкость и стабильность в работе, полноценный графический интерфейс, обладает развитыми сетевыми средствами, предназначенными для работы в сетях, использует все известные стандартизированные протоколы передачи данных. ОС Linux является операционной средой и может настраиваться под конкретный класс задач, существуют дистрибутивы, компилируемые под конкретное аппаратное обеспечение, что оптимизирует работу операционной системы**.**

Первая версия ОС Linux разработана в начале 90-х годах и предназначалась для персональных компьютеров, которые в это время стремительно развивались. Программное обеспечение (ПО) первоначально разрабатывалось для проведения сложных математических расчетов в сфере научных исследований.

ОС Linux имеет большое количество различных дистрибутивов, основные из которых: Red Hat компании RedHat и ее следующая версия Fedora Core; Mandrake - версия, по основным функциям похожая на Red Hat, постоянно обновляющаяся, имеющая удобную инсталляцию и настройки графической среды; версия Debian, являющаяся некоммерческой разработкой и ее последовательница Ubuntu, очень быстро завоевавшая известность. В настоящее время перечисленные версии в основном русифицированы, и некоторые пользователи в нашей стране знают их под другими именами.

ОС Linux работает как в графическом так и в текстовом режимах. Основные команды выполняются в любом из этих режимов. Графическому режиму соответствует графический интерфейс, текстовому режиму - режим интерфейса командной строки, которую обрабатывает командный интерпретатор (КИ). В ОС Linux представлен полный набор КИ, входящих также в ОС UNIX: TCSH, Z-shell, PDKSH. Наиболее распространённый из них - BASH. Режим командной строки используется в основном для выполнения программ расчетов, системного администрирования, управления работой серверов. В нем удобно создавать программы-сценарии для выполнения последовательности действий, групповых выборок, сложных сортировок, сложных команд сетевого администрирования. Широко используются средства разработки в среде Linux. Они имеют качественные компиляторы, большие наборы подключаемых библиотек, удобные утилиты.

ОС Linux - профессиональная операционная система, она удобна для широкого класса пользователей, может настраиваться под конкретный класс задач**.**

Для графических столов в настоящее время существует много приложений. Разработаны офисные программы, растровые и векторные редакторы, браузеры, почтовые программы, которые могут быть различными для разных графических столов.

Изучение ОС Linux стало актуальным для широкого класса разработчиков, программистов и пользователей компьютеров в различных сферах деятельности.

Лабораторная работа №1

Ознакомление с рабочим столом ОС Linux и командным интерпретатором BASH

***Цель работы*:** изучение графического стола Red Hat, приемов работы в командном интерпретаторе и основных команд командного интерпретатора BASH, используемого в ОС Linux и ОС UNIX.
***Продолжительность работы*:** - 4 ч.

Теоретические сведения

Операционная система Linux

ОС Linux, как и ОС UNIX имеет базовую структуру системы: ядро, системная среда, файловая система.

Ядро - основные программы управления аппаратными средствами и программы, обеспечивающие взаимодействие с прикладным программным обеспечением. ***Файловая система*** предназначена для работы с файлами (создание, хранение, копирование, перемещение, удаление, добавление). Она имеет иерархическую структуру и устроена таким образом, что можно работать с файлами большого размера и с большими дисками, при этом скорость обращения к файлам и каталогам высокая, действия выполняются с большой степенью надежности. ***Системная среда*** обслуживает функционирование интерфейса пользователя. Она принимает от пользователя команды и посылает их в ядро ОС для исполнения. *Системную среду* можно назвать интерпретатором, так как она преобразует команды пользователя и направляет их в ядро. В ОС разработано несколько видов системной среды. Это ***рабочие столы, менеджеры окон, интерпретаторы командной строки***, каждый из которых является отдельным модулем ОС Linux. Пользовательский интерфейс настраивается под конкретные нужды определенного пользователя и для каждого пользователя может быть свой.

Командный интерпретатор **(**КИ) имеет ***командную строку***, в которую вводятся команды пользователя. После ввода каждой команды пользователь нажимает клавишу [Enter], после чего команда выполняется. В ОС Unix и ОС Linux может использоваться несколько командных интерпретаторов и их различных версий, устанавливаемых на выбор пользователя. Наиболее известными из них являются: Bourne Again shell, TCSH-shell, Public Domain Korn shell (PDKSH), C, Z-shell. Самый распространенный из них BASH shall, представляющий усовершенствованную версию командного интерпретатора Bourne. Его полное название Bourne Again shell. Для версий ОС Unix BSD первоначально была разработана расширенная версия интерпретатора C, названная TCSH-shell, которая позднее стала использоваться в ОС Linux. По умолчанию в ОС Linux применяется интерпретатор Bourne Again shell, сокращенно BASH, но можно использовать по умолчанию и любой другой КИ, имеющийся в системе. В ОС UNIX более предпочтительными являются другие командные интерпретаторы, например, Z - расширенная версия интерпретатора Korn. Командный интерпретатор обеспечен текстовым режимом работы ОС и потребляет незначительные ресурсы оперативной памяти. В процессе работы в одном рабочем окне, называемом окном терминала, можно применять один командный интерпретатор.

Менеджеры окон - это усечённая версия рабочего стола, поддерживающая операции с окнами, управляющая видом самого окна, его рамок, меню. Менеджер окон состоит из трех частей: *системы X Windows*, обеспечивающей открытие окон и вывод изображения на экран; *менеджера файлов,* выполняющего действия с файлами, использующего пиктограммы и меню; *менеджера программ* - запускающего программы на выполнение, располагающего их на панели задач.

Рабочие столы. В отличие от текстового режима работы КИ можно использовать графический режим, который обеспечивается оконным интерфейсом и представлен рабочим столом с его атрибутами. Рабочий стол дает полноценный графический интерфейс пользователя, он похож на графический интерфейс ОС Windows или OS/2. Как и другие рабочие столы, рабочие столы ОС Linux и ОС Unix имеют аналогичные основные объекты: собственно рабочий стол, на нем располагаются окна, пиктограммы, меню, панель задач, среди пиктограмм есть **Home** (или **Home directory**) и **Trash** аналогично **«Мой компьютер»** в ОС Windows и **«Корзина»**. Управлять этими объектами можно также с помощью мыши или клавиатуры.

Рабочий стол представлен четырьмя виртуальными рабочими столами, установленными по умолчанию, но настроить ОС можно и на большее количество, равное двум в натуральной степени. На каждом рабочем столе могут быть расположены свои пиктограммы, обычно всегда присутствуют **Home directory** и **Trash**. Наиболее распространенными для различных дистрибутивов и версий ОС Linux являются рабочие столы **GNOME** и **KDE** (рис.1).

По нажатию левой кнопки мыши объекты рабочего стола выделяются, двойному нажатию запускаются на выполнение, по нажатию правой открывается дополнительное меню, в нем можно установить свойства объекта. Предусмотрены настройки рабочих столов, дающие возможность выделить объект подведением курсора, а запустить задание на выполнение одиночным нажатием левой кнопки мыши.

Панелей задач может быть несколько, размещаются произвольно на рабочем столе согласно вкусам пользователя. Рекомендуется использование двух или нескольких панелей в случаях, когда приложения разбиваются на группы по смысловому содержанию.

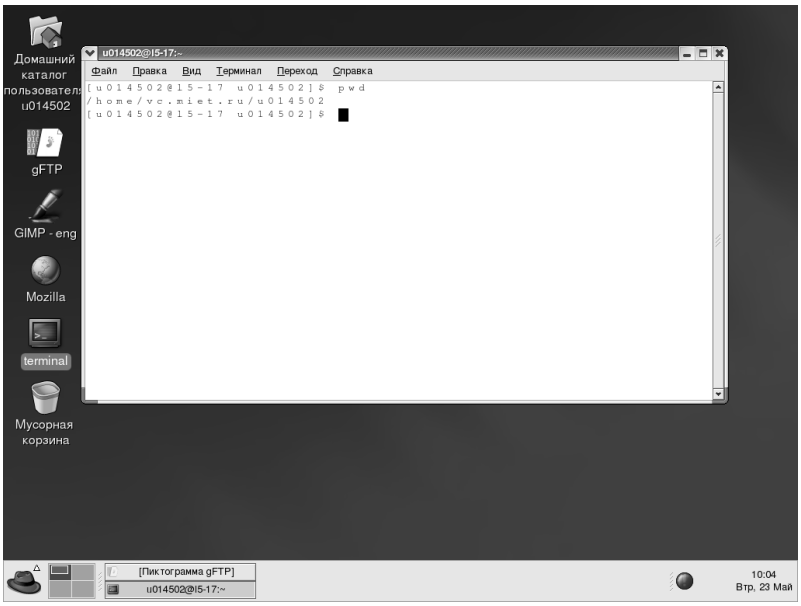

Рис.1. Графический стол Red Hat и окно командного интерпретатора

Рабочие столы Gnome и KDE

Состав рабочих столов **Gnome** (GNU Network Object Model Invironment) и **KDE** (K Desktop) следующий: интерфейс состоит из панели задач, собственно рабочего стола и расположенных на нем пикторамм. Панель - длинная полоса внизу экрана, которая, однако, может находиться в любом удобном месте рабочего стола. На ней размещаются меню, программы и **апплеты** - небольшие программы, в основном стандартные и служебные, предназначенные для запуска с панели.

На рабочем столе располагаются пикторамы начального каталога, Web-браузера, корзины, а также прикладных программ и файлов на усмотрение пользователя. В верхнем левом углу на рабочем столе находится пиктограмма начального каталога, называемая **Home Directory**. При щелчке левой кнопкой мыши по этой пиктограмме откроется окно менеджера файлов. В левой его части расположено дерево каталогов, в правой - содержимое выбранного подкаталога. При входе открывается личный каталог пользователя в соответствии с его регистрационным именем.

Рабочие столы поддерживают функции перемещения. Если перемещать файл или другой объект и нажимать на среднюю кнопку мыши или, если ее нет, на левую и правую кнопки одновременно, то на рабочем столе будет создаваться ссылка на первоначальный объект. Окна содержат кнопки увеличения/уменьшения, закрытия, свертывания на панель задач, иногда кнопку «закрепить», позволяющую закрепить окно в определённом месте экрана. Чтобы переместить окно, нужно щелкнуть мышью на строке заголовка и, не отпуская кнопки мыши, переместить окно в новую позицию. Двойной щелчок в строке заголовка уменьшает окно до размера строки заголовка, следующий двойной щелчок восстанавливает окно.

Рабочие столы можно настраивать и оформлять в соответствии с выбранной темой. Размер рабочего стола превышает размер экрана монитора, в ОС Linux создаются виртуальные рабочие столы, на которые можно переключаться посредством специальной пиктограммы на панели задач. Рабочий стол может состоять из нескольких виртуальных столов, на которых можно расположить группы пиктограмм по темам.

Рабочий стол **Gnome** является мощной и простой в использовании системной средой. Это основной графический интерфейс, поддерживаемый дистрибутивом Red Hat, но он также входит в большинство других распространенных дистрибутивов.

Основные компоненты рабочего стола **Gnome**: функциональные средства, представленные пиктограммами, панель запуска программ, средства настроек. Обычно на рабочем столе присутствуют приложения, совместимые с Gnome: менеджер файлов Nautilus, Web-браузер Mozilla, менеджер окон, использующий базовуюсистему X Window. Слева на панели задач имеется кнопка с изображением ступни гнома, открывающая главное меню рабочего стола Gnome со списком задач, которые можно запустить, что реализовано подобно кнопке **Пуск** в ОС Windows. На панели можно установить ее автоматическое свертывание и развертывание. Возможно использование нескольких панелей на рабочем столе. Количество виртуальных рабочих столов по умолчанию устанавливается равным четырем. Пользователь на свое усмотрение может поместить на эти столы пиктограммы наиболее часто используемых приложений, файлов и каталогов.

Если войти в основное меню на панели задач, изображаемое ступней гнома или красной шляпой, выбрать строку **Programs**, в следующем открывшемся меню - **Office**, далее - **Офисные программы**, то в этом меню можно найти текстовый редактор, который по функциям аналогичен программе Word под Windows, табличный редактор и программу презентаций. Наибольшее распространение получил набор офисных программ, называемый OpenOffice.org, являющийся бесплатным и доступным для копирования из Интернета.

Рабочий стол **KDE** является универсальным рабочим столом и содержит менеджер окон, менеджер файлов, корзину, набор приложений. Рабочий стол KDE хорошо приспособлен для работы в Internet и имеет полный набор встроенных сетевых приложений: почтовую программу, программу чтения новостей, **Web-браузер**. Менеджер файлов устроен таким образом, что является клиентом **FTP** и **Web**, это дает возможность заходить на узлы Internet с рабочего стола.

Интерфейс **KDE** схож с интерфейсом **GNOME**, но для выделения объекта нужно нажать на левую кнопку мыши одновременно с нажатием на клавишу **Ctrl**. А для запуска программы на выполнение нужно один раз нажать на левую кнопку мыши на требуемом файле, что соответствует в ОС Windows двойному нажатию левой кнопки мыши. Рабочий стол может состоять из 4 - 16 частей, на каждую из которых можно переключаться на специальной пиктограмме, расположенной на панели задач.

Файловая структура ОС Linux

Файловая структура ОС Linux – иерархическая структура каталогов, в которой размещена информация о файлах всей операционной системы. Дерево каталогов ОС Linux имеет один общийкорневой каталог для всей системы, обозначаемый знаком "слеш" (/) и включающий основные системные каталоги bin, boot, dev, etc, lib, lost+found, mnt, proc, sbin, pub, opt, tmp, var, usr. В версиях ядра ОС Linux могут быть небольшие различия в составе входящих каталогов.

Каталог **bin** содержит пользовательские команды и утилиты (примеры: ***cat, ls , fgrep, mkdir, zcat***); **dev** используется для специальных файлов, представляющих устройства; **etc** служит для хранения команд администратора системы; **lib** включает важные совместно используемые библиотеки и модули ядра; **lost+found** необходим дя быстрого восстановления системы после сбоев; **media** – для монтирования компакт-дисков и flash-накопителей; **mnt** - содержит каталоги для монтирования новых файловых систем и временных накопителей; **proc** - каталог процессов; **opt** - дополнительные приложения (например, **GNOME,** если установлено **KDE**); **sbin** содержит системные команды, включая команды привилегированного пользователя; **tmp** используется для хранения временных файлов; **usr** включает исходные коды, файлы и команды системы, документацию, содержащую пользовательскую информацию, новые программы, устанавливаемые по умолчанию; **var** - каталог, предназначенный для быстро изменяющихся файлов, в частности, файлов почтовых ящиков, системных журналов, протоколов приложений, очереей печати.

В свою очередь эти каталоги могут содержать каталоги следующего уровня. Каталог **usr** включает подкаталоги: **bin** - хранение дополнительных команд; **games** - игры; **include** - хранение фрагментов системных программ; **lib** - хранение дополнительных библиотек.

Полные имена файлов, как и в других ОС включают имена каталогов на пути к ним: **/usr/bin**, **/usr/games**, **/usr/include, /usr/lib**. Формальным признаком полного имени в ОС Linux и Unix является то, что оно начинается со знака "слэш", обозначаемого знаком (/).

В качестве имени файла, как правило, может использоваться любая последовательность из букв, цифр и знаков подчеркиваний длиной до 255 символов. Могут применяться и другие символы, однако использование этих символов в имени требует специального экранирования. Старые файловые системы поддерживают длину имени файла четырнадцать символов (этого ограничения желательно придерживаться для переносимости файлов), однако в большинстве систем допускаются более длинные имена - до 256 символов. В общем случае расширения в именах файлов не являются обязательными. В ОС UNIX и ОС Linux прописные и строчные буквы воспринимаются как различные, например, Ivan, IVAN и ivan - это три различных имени.

ОС UNIX и ОС Linux работают с файловой системой, а не с устройствами хранения информации, как это реализовано в ОС Widows и многих других ОС. Прежде чем считать информацию с внешнего устройства, его нужно смонтировать специальной командой **mount**, после чего подсоединенные файлы и каталоги становятся элементами файловой системы, и пользователь может обращаться к любым доступным файлам и каталогам, при этом в имени никак не отражается устройство, на котором файл (каталог) находится или создается.

Отдельные части файловой системы могут находиться на различных физических устройствах, например, на нескольких жестких и гибких дисках (или в различных частях одного диска). Соответствующие фрагменты (поддеревья файловой системы) монтируются в единую файловую систему также командой mount.

В ОС UNIX и ОС Linux имеется примерно 200 базовых команд - инструментальных средств, позволяющих пользователю решать многие проблемы, не прибегая к программированию на языках типа Си или использованию специальных пакетов.

Основные типы файловых систем, поддерживаемые **ОС Linux: ext2, ext3, swap, ReiserFS, JFS.**

Командный интерпретатор

Командные интерпретаторы используют ***интерфейс командной строки***. В начале строки появляется ***приглашение к работе*****,** после которого можно вводить команду (рис.1).

Большинство команд **ОС UNIX** и **ОС Linux**, выполняемых в командном интерпретаторе, можно выполнить из специальных утилит графического интерфейса, но не все, некоторые гораздо быстрее выполнить из КИ с заданием опций и параметров.

Структура команды командного интерпретатора следующая:

$ имя_команды опции параметры

Появление приглашения к работе со знаком **$** в конце приглашенияили знаком # зависит от того, с какими правами доступа зарегистрирован работающий в системе. Знак **$** является приглашением к работе для пользователя, # - приглашение для системного администратора. **Опция** - однобуквенный код, перед которым стоит дефис, уточняет действия команды. После дефиса может быть указано несколько опций. Все последующие указываются без пробелов. Дефис обозначает, что после него и до знака пробела указаны опции. Для каждой конкретной команды опции и параметры могут быть обязательными или нет. **Параметр** - дополнительные данные для выполнения команды, часто это имя файла или каталога. Указанный порядок расположения компонентов структуры команды **КИ** обязателен для любых командных интерпретаторов.

Пример 1. Команда **ls** выводит на экран протокол, состоящий из файлов текущего каталога. Опция **l** команды **ls** дает подробную информацию о файле. Чтобы вывести на экран подробную информацию обо всех файлах текущего каталога, следует набрать в командной строке:

$ ls -l

На экране появится протокол, фрагмент которого приведен ниже:

drwxrwxr-x 2 root 2048 nov 3 12: 11 bin
drwxrwxr-x 2 root 1024 jan 9 11:55 dev
drw-r--r-- 3 root 4096 nov 17 12: 01 include
drwxr-xr-x 7 root 480 nov 17 12:30 lib

Пример 2. Вывести подробную информацию об одном файле, например, о файле **file1**, находящемся в текущем каталоге:

$ ls -l file1

На экране появится протокол, состоящий из одной строки из одной строки, с подробной информацией о свойствах файла:

drwxrwxr-x 2 ivan 8192 jan 9 11:55 file1

Пример 3. Вывести подробную информацию о файлах каталога **usr**, пользователь при этом находится в корневом каталоге (рис.2):

$ ls -l /usr

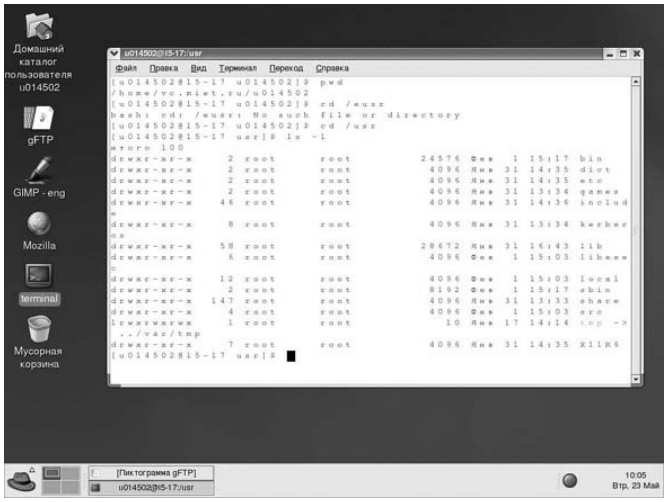

Рис.2. Просмотр содержимого текущего каталога **usr** командой **ls** с опцией **l**

Права доступа к файлам и каталогам

Каждому файлу или каталогу соответствует строка информации при выводе команды **ls –l**, как показано в примерах 1 и 2. Первый символ строки означает тип файла (**d** - каталог, знак "дефис"(-) - обычный файл). В каждой строке символы со второго по десятый описывают **права доступа** к файлам и каталогам, опеделенные для трех категорий пользователей: **владельцев, членов группы, прочих пользователей**. Каждая категория пользователей может иметь право на **чтение**, **запись**, **выполнение** файлов. В примере 2 дан потокол, в котором первая триада - **права владельца**, которая разрешает: **r** - чтение каталога, **w** - запись в него и **x** - выполнение, для файлов типа d (каталогов) w означает разрешение создавать в каталоге файлы и удалять их из него; вторая триада - **rwx** , предназначена **членам группы**, в которую входит владелец файла, для нее разрешены все три операции. Последняя триада **r-x** отражает права доступа **прочих пользователей**, которым разрешено только читать и выполнять файл, запрещено записывать в файл. Далее следует цифра 2, указывающая число сссылок на файл, что означает наличие в системе еще одного имени-ссылки, связанного с этим файлом; ivan - имя владельца, 8192 - число символов в файле, дата и время создания или последней модификации файла (9 января в 11 часов 55 минут); file1 - имя файла.

Команды командного интерпретатора BASH и специальные символы

Язык команд КИ ОС UNIX и ОС Linux часто называют **shell**, что в переводе с английского означает "скорлупа". Ниже приведены часто используемые команды.

Справочник команд терминала man и общие команды. Команда **man** является справочником и содержит описание команд с их опциями и параметрами, так же приведены примеры использования команд. Если набрать в командной строке командного интерпретатора:

$ man

то откроется весь справочник, повключающий команды интерпретатора.

Если указать имя команды, то можно получить справочную информацию об указанной команде:

$ man <имя команды>

Например, информацию о командном интерпретаторе **BASH** можно посмотреть следующим образом: **$ man bash** Информацию о команде даты и времени можно посмотреть так: **$ man date** Информацию о **дате** и времени можно посмотреть так: **$ date** Результатом будет строка с указанием дня недели, месяца, года, даты и времени.

При задании команды в **КИ** можно указывать одну опцию или несколько. Например, чтобы показать подробную информацию обо всех файлах текущего каталога, включая скрытые файлы (опция **a**), в командной строке вводится: **$ ls -alF**

Прописные и строчные буквы различаются и при указании опций. Таким образом, F и f являются разными опциями. Если опций несколько, то они пишутся подряд без пробелов, начиная со знака "дефис". Выполнение команды начинается после нажатия на клавишу [Enter]. Если команда запущена на выполнение ошибочно, отменить ее действие можно, используя нажатие клавиши [Ctrl+c] или [Del]. Если команда с ее опциями и параметрами оказалась длиннее строки, то для продолжения команды в следующей строке нужно ввести символ "обратный слеш" (\), потом [Enter], далее продолжать набор команды в следующей строке. Вместо символа "обратный слеш" может использоваться другой символ, если его задать в переменной командного интерпретатора, что описано ниже.

Команда history. Командная строка представляет собой текстовый буфер ввода с возможностью редактирования. Если набрать команду с необходимыми опциями и параметрами до нажатия клавиши [Enter], текст можно отредактировать. Используя клавиши [Backspace] или [Del] можно удалить ненужные символы, а, включив режим вставки (клавиша [Insert]), можно вставить нужные символы, набрав их на клавиатуре. Перемещаться по тексту можно, используя клавиши стрелок влево и вправо (назад и вперед соответственно).

Существует архив команд текущего сеанса работы, находящийся в файле **history_list**, благодаря которому можно повторять предыдущие команды. Используя клавишу ↑, следует найти нужную команду из ранее выполненных в данном сеансе командного интерпретатора, отредактировать ее, если нужно, и, нажав клавишу [Enter], выполнить.

Нажимая несколько раз клавишу ↑, находим нужную команду. Если нужно вернуться по этому списку команд, нажимать клавишу↓.

Список последних выполненных команд с их номерами можно просмотреть так же, воспользовавшись командой-утилитой:

$ history

Выполнить команду из списка историй можно так: **$!№**

где № - номер команды, полученный из списка **history_list**. Если № не указан, то имеется в виду последняя выполненная команда. Вместо № можно набрать несколько первых символов команды из списка команд, однозначно ее идентифицирующую. Попробуйте повторить одну из ранее выполненных команд, обратившись к ней по номеру из списка событий.

Снятие блокировки (!). Эта команда используется, чтобы разрешить запись в уже существующий файл. Обычно по умолчанию установлена проверка на наличие существующих файлов и выдача на экран предупреждающего сообщения, но иногда файлы нужно перезаписывать в уже существующие. Тогда используется снятие блокировки. Например, файл **f10** предположительно уже существует, но в него нужно записать содержиое файла **file10**, тогда удобна следующая запись:

$ cat file10 >! f10

Удобство более ощутимо, когда перезаписывается много файлов, поскольку просматривать, есть ли предыдущие версии, и потом удалять их дольше, чем поставить знак ! в вышеуказанной команде.

Путевое имя - это полное имя файла с учетом каталогов, начиная с домашней директории пользователя, включает последовательно имена всех каталогов, ведущих к текущему. Иначе имя называется относительным. При входе в систему пользователь оказывается в определенной заранее вершине дерева файловой системы. Обычно это /**home** или /**home/имя_пользователя**, иногда используется каталог **/usr**. Путевое имя для суперпользователя **root** - путь от корневого каталога в текущий каталог. Команда **pwd** сообщает местоположение пользователя в файловой системе. С ее помощью выводится полное имя текущего каталога.

$ pwd

Команды создания и удаления каталогов. Команда предназначена для создания каталогов в любой вершине файловой структуры при наличии прав доступа и имеет следующий синтаксис:

$ mkdir <имена создаваемых каталогов>

Например, для создания в текущем каталоге двух новых каталогов с относительными именами **k1** и **k2**, нужно выполнить команду:

$ mkdir k1 k2

Для удаления пустых (не содержащих файлов и каталогов) каталогов используется команда **rmdir**, имеющая структуру:

$ rmdir <имена удаляемых каталогов>

Удалить обычный файл можно командой:

$ rm <имена удаляемых файлов>

Чтобы просмотреть содержимое рабочего (текущего) каталога со всеми его подкаталогами (рис.3): **$ ls –R**

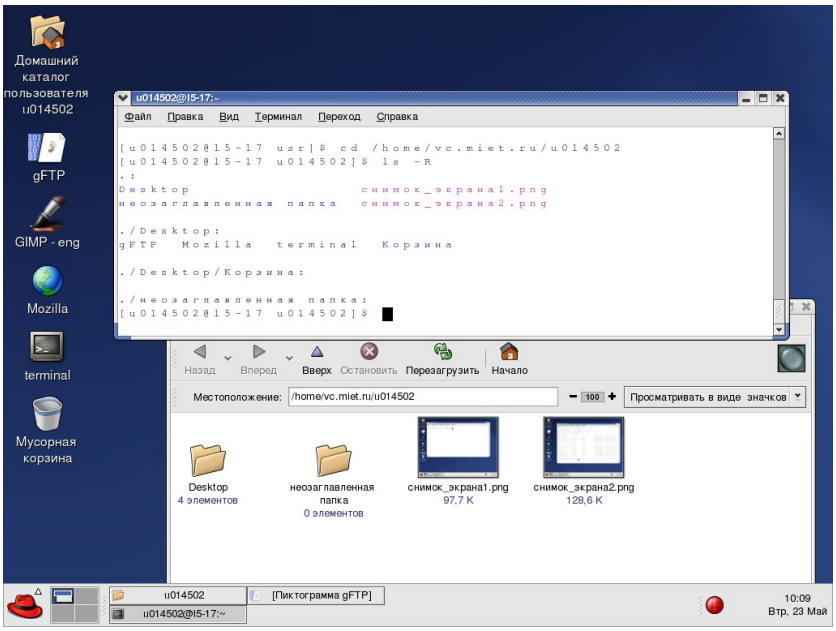

Рис.3. Просмотр содержимого текущего каталога. Рабочий стол **GNOME**

Выводится писок файлов и подкаталогов в текущем каталоге с указанием имен подкаталогов, после их имени ставится знак "слеш" (/).

Перемещение по каталогам. Чтобы перейти в другой каталог, используется команда **cd**:

$ cd <каталог>

Этот символ можно также использовать при копировании или перемещении, указав каталог на один уровень выше, чем текущий.

Пример 1. Находясь в корневом каталоге, можно перейти в каталог **/usr/lib**, набрав команду:

$ cd /usr/lib

Указано полное имя каталога **/usr/lib**. Если пользователь находится в каталоге **usr**, то указывается относительное имя: **$ cd lib**

Пример 2. Команда **cd** без параметров осуществит переход в начальный каталог пользователя, т.е. в каталог, в котором пользователь оказывается при входе в систему: **$ cd**

Копирование файлов и каталогов выполняется командой **cp**, причем последнее указанное в строке КИ имя считается именем каталога, в который нужно скопировать файлы. Общая структура команды:

$ cp <полное_имя_файла/имя_каталога,_которые_копируются>
<полное_имя_файла/имя_каталога,_куда_копируется>

Команда **cp** имеет следующие основные опции:

-r - используется при копировании подкаталога из одного каталога в другой. Копирование выполняется со всеми подкаталогами, входящими в копируемый подкаталог.

-i - проверка, существует ли уже файл (каталог) с таким именем, в месте, указанном для копирования.

Полное имя можно не указывать, если действия выполняются в текущем каталоге, что в общем виде представлено структурой:

$ cp <имя_файла> имя_каталога/новое_имя_файла>

Пример 3. Копирование файла file1 из каталога **dx** в каталог user в файл file2, каталоги **dx** и **user** лежат в текущем каталоге:

$ cp dx/file1 user/file2 ,

Пример 4. $ cp file1_katal/file11

Здесь **katal** - каталог, находящийся в текущем каталоге. Можно использовать и такую форму записи, но тогда имя нового файла в каталоге **katal** будет как и исходное, **file1**:

$ cp file1 katal

Пример 5. Можно копировать несколько файлов в каталог:

$ cp <имена_файлов> <имя_каталога>,

$ cp file1 file2 katal

Чтобы скопировать все файлы текущего каталога в указанный каталог **katal2** (сохранив также старые файлы) для указания нескольких символов используется знак “*”, следует выполнить:

$ cp *.* katal2

Пример 6. Чтобы скопировать все файлы с заданным расширением:

$ cp *.cpp katal3

Перемещение файлов и каталогов выполняется командой **mv**. Синтаксис команды аналогичен синтаксису команды **cp**. Общая структура команды:

$ mv <имя_файла/каталога> <имя_файла/каталога>

Пример 7. Чтобы файлу **file1** присвоить имя **file5**, следует выполнить команду:

$ mv file1 file5

Опции **-r-** и **-i** команды **mv** имеют назначение, как и в команде **cp**.

Знак "тильда". При выполнении команд, использующих файлы и каталоги, иногда требуется указывать полный путь к файлу или каталогу. Для сокращенной записи используется знак "тильда**" (~)**.

Пример 8. Использование знака "тильда"**.**

$ cp file1 ~/katal1

$ mv file1 ~/katal1

$ mv ../file5

Знак "точка". Каталог, в котором работает пользователь, называется рабочим каталогом. Текущий рабочий каталог можно обозначать знаком **"**точка**" (.)**.

Пример 9. Пользователь находится в каталоге **katal**, нужно посмотреть его содержимое. Используется любая из следующих записей: **$ ls**

или запись: **$ ls .** Точка здесь означает текущий каталог.

Пример 10. Для копирования из какого-либо каталога в текущий каталог можно использовать знак "точка":

$ cp /home/mykatal/file7 .

Знак "две точки" (..).Переход в родительский каталог обозначается двумя точками **".."**.

Пример 10. Чтобы не подниматься еще на один уровень вверх, а посмотреть содержимое каталога, в который входит рабочий каталог **katal**, т.е. содержимое родительского каталога, можно записать: **$ ls ..**

Пример 11. Из **/usr/include/sys** произойдет переход в **/usr/include**, если набрать в командной строке: **$ cd ..**

Пример 11. Команды просмотра файлов cat и more. Общая структура команд:

$ cat <имя_файла>,

$ more <имя_файла>

Команда **more** используется для просмотра больших файлов, т.к. дает постраничный просмотр и имеет несколько опций для движения по файлу.

Пример 12. Команду **cat** можно использовать для просмотра небольших файлов следующим образом:

$ cat имя_файла

Команду **cat** можно использовать для создания текстовых файлов:

$ cat > имя_файла

Например: **$ cat > file1.** Далее ввести с клавиатуры текст файла. Набрать **Ctrl ^ D** одновременно. Файл создан.

***Команда* rm.** Для удаления файлов используется команда **rm**:

$ rm <список_файлов>

Можно удалять один или несколько файлов, при записи разделенных пробелами. Количество имен файлов можно указать любое.

При использовании опции **-i** появляется сообщение о подтверждении удаления, нужно указать "**да**" или "**нет**" (обычно **yes** или **no**).

Для удаления подкаталогов используется опция **-r** , а чтобы стереть все файлы в текущем каталоге вместо имен файлов можно указать знак ***:

$ rm *

Проверить команду **Что за команда?** (это календарь **call**).

$ whatis call

Найти подкаталог **katal** в указанном каталоге **/home/student**:

$ find /home/student type -d katal

Найти файл **file1** в указанном каталоге **/home/student**:

$ find /home/Student type -f file1

Знак вопроса заменяет один произвольный символ, командой **find** ведется поиск файлов, названия которых начинаются на file, а далее следует ещё один символ названия:

$ find . -name file?

Программные каналы, перенаправление входных и выходных потоков данных

Многие команды ОС **Linux** принимают данные со стандартного ввода или передают на стандартный вывод. По умолчанию в качестве устройства для стандартного ввода используется клавиатура, для стандартного вывода - терминал.

Перенаправление стандартного вывода в файл или на устройство обозначается знаком "больше" (**>**). Стандартный ввод может быть получен не с клавиатуры, а из файла. Оператором перенаправления стандартного ввода является знак "меньше" (<).

Пример 1. Вывести на экран содержимое файла **file** и записать в новый файл **file1** можно, используя перенаправление стандартного ввода-вывода, представляющего байтовый поток. Чтение данных из файла **file** и запись в **fi1le1** в командной строке осуществляются следующим образом:

$ cat < file > file1

То же самое, но только с опцией **-i** - проверки наличия файла **filef**:

$ cat < file > -i file1

Поскольку файл уже создан, появится сообщение о наличии файла **file1**, нужно ответить **n**, чтобы не изменять уже существующий файл, или **y**, чтобы записать в него.

Если требуется добавить информацию в файл, т.е. осуществлять запись в него не с начала файла, а в конец уже имеющейся записи в файле, то используется оператор добавления. Допишем в **fi1e1** информацию из файла **file2**:

$ cat file2 >> file1

ОС Linux поддерживает стандартный вывод сообщений об ошибках (**2>**), отличающийся от стандартного вывода, результат выводится в файл и в зависимости от настроек на экран.

Пример 2. Чтобы сообщения об ошибках выполнения команды **cp** записать в файл **error**, нужно выполнить команду:

$ cp fileprog /k1 2> error

Чтобы добавить сообщения об ошибках в файл **error,** вместо знака (>) предыдущей команды нужно использовать знак добавления (>>):

$ cat fileinfo 2>> **error**

Ошибка могла возникнуть, если указанные файлы не существуют.

Для передачи результат выполнения команды в другую команду используется символ вертикальной черты **|** - так называемый программный канал. Другими словами, он передаёт выходные данные команды на вход следующей команды.

Пример 3. Передать в файл **filelist** список имен файлов текущего каталога, выданный командой **ls**. Передача данных в данном примере осуществляется по программному каналу: **$ ls | cat > filelist**.

Программа-оболочка Midnight Commander

Программа **Midnight Commander** является аналогом операционной оболочки **Norton Commander** и имеет с ней большое сходство в интерфейсе и функциях. Она выполняет команды менеджера файлов в текстовом режиме и имеет дополнительно удобные средства просмотра файлов и файловых структур и а также удобные свойства. Запуск **Midnight Commander** осуществляется из окна терминала:

$ mc

Текстовые редакторы

Пользователь может создавать файлы с использованием текстового редактора **vi** и его расширенной версии **vim**, а также графических редакторов **gvim**, **gedit**, **gnotepad+** и других. Все они являются небольшими редакторами, некоторые из них имеют встроенные функции.

При отладке программ, работе с серверными приложениями, когда нет необходимости устанавливать графический режим, широко используются текстовый редактор **vi** и его расширенный вариант - текстовый редактор **vim**. Они являются командами и имеются в большинстве командных интерпретаторов, включая **BASH**. Их можно запустить из командной строки следующим образом:

$ vi и **$ vim**

Текстовый редактор **vi** дает возможность выполнить основные команды редактирования и ввода текста, **vim** - более мощный текстовый редактор, имеющий большое количество команд редактирования.

Эти редакторы используют в основном клавиатуру и имеют два режима работы: командный и режим ввода текста. После запуска редактора появляется рабочее окно, внизу которого находится командная строка. Набрав в ней команду **i** (**input** - ввод) или **a** (**add** - добавление), можно приступать к вводу текста. Выход из режима ввода осуществляется нажатием клавиши **Esc**, переход в командный режим можно осуществить, используя знак "**:**" . Чтобы запомнить промежуточные результаты ввода, в командном режиме необходимо использовать команду **w**. Выход из редактора осуществляется нажатием **ZZ** при нажатой клавише **Shift**, выход с запоминанием результатов редактирования - **wq**. Чтобы выйти из редактора без запоминания, в случае, если не было редактирования файла перед этим, в командном режиме выполняется команда **q**, если выполнялось редактирование - команда **q!**. Возможно использование команды **exit**.

Продолжением разработки редактора vim является редактор **gvim**, он работает в графическом режиме и имеет встроенный интерфейс, как у редактора **vi**, но дополнен строкой с несколькими кнопками меню, расположенными вверху окна, позволяющими выбирать команды редактора с помощью мыши.

Для рабочего стола **GNOME** основным является графический редактор **gedit**, обеспечивающий полную поддержку мыши и реализующий основные функции по редактированию файлов; его функциональные возможности шире, чем у предыдущих.

Для редактирования небольших текстовых файлов и создания web-страниц предназначен графический редактор **gnotepad+**. Он имеет панель инструментов, дающую возможность вставлять некоторые часто используемые элементы языка **HTML**.

Компилятор gcc

Для отладки и выполнения программ на языке С используется компилятор gcc. Практически все дистрибутивы Linux поставляются вместе с компиляторами языка С и языка С++. На начальном этапе обучения версия компилятора не имеет значения. Если в дистрибутиве есть компилятор gcc, то обычно есть и линковщик.

Создадим отдельным текстовым файлом простейшую программу, пусть ее имя будет **prog1.c** (расширение писать обязательно), а исходный текст следующий:

```
#include <stdio.h>

int main (void)
{
        printf ("Hello World\n");
}
```

Чтобы откомпилировать программу, достаточно набрать в командной строке:

```
$ gcc -o prog1 prog1.c
$
```

Команда **gcc**, вводимая в командной строке командного интерпретатора, выполнит компиляцию, и если в программе нет **синтаксических** ошибок, то компилятор завершит свою работу без каких-либо сообщений и создаст исполняемый модуль **prog1.exe** в текущем директории: Набрав команду **ls**, вы обнаружите новый файл с именем **prog1**. Этот файл содержит исполняемый код программы. Такие файлы называют исполняемыми файлами ('executable files') или *бинарниками* ('binary files').

Опция -o компилятора gcc указывает на то, каким должно быть имя **выходного** файла. Выходным файлом может быть не только бинарник. Если не указать опцию -o, то бинарнику, в нашем случае, будет присвоено имя a.out.

Осталось только запустить полученный бинарный файл на выполнение. Для этого набираем в командной строке следующую команду:

```
$ ./prog1
Hello World
$
```

Когда мы набираем в командной строке путь к бинарнику, мы сообщаем КИ, что надо выполнить программу. КИ передает бинарник ядру операционной системы, а ядро системы особым способом отдает программу на выполнение процессору. Затем, если программа не была запущена в фоновом режиме, то КИ ждет от ядра сообщения о том, что программа выполнилась. Получив такое сообщение, КИ выдает приглашение на ввод новой команды. Вы можете еще раз набрать **./prog1** и процедура повторится. В нашем случае программа выполняется очень быстро, и новое приглашение командной строки появляется практически сразу.

Мы рассмотрели идеальный случай, когда программа написана без **синтаксических** ошибок. Попробуем намеренно испортить программу таким образом, чтобы была ошибка. Для этого достаточно убрать точку с запятой в конце строки функции **printf()**:

```
printf ("Hello World\n")
```

Теперь, если попытаться откомпилировать программу, то компилятор укажет на то, что он считает неправильным:

```
$ gcc -o prog1 prog1.c
hello.c: In function 'main':
hello.c:7: error: syntax error before '}' token
$
```

В первой строке говорится, что в файле prog1.c в теле функции main() что-то произошло. Вторая строка сообщает, что именно произошло: седьмая строка файла hello.c вызвала ошибку (error). Далее идет расшифровка: синтаксическая ошибка перед закрывающейся фигурной скобкой.

Заглянув в файл prog1.c мы обнаружим, что ошибка не в седьмой, а в шестой строке. Дело в том, что компилятор обнаружил неполадки только в седьмой строке, но написал before (до), что означает "смотри перед".

Естественно, пока мы не исправим ошибку, ни о каком бинарнике не может идти и речи. Если мы удалим старый бинарник prog1, оставшийся от прошлой компиляции, то увидим, что компиляция испорченного файла не даст никакого результата. Однако иногда компилятор может лишь "заподозрить" что-то неладное, потенциально опасное для нормального существования программы. Тогда вместо 'error' пишется 'warning' (предупреждение), и бинарник все-таки появляется на свет (если в другом месте нет явных ошибок). Не следует игнорировать предупреждения, за исключением тех случаев, когда вы на 100% знаете, что делаете.

Мультифайловое программирование

Если исходный код сколько-нибудь серьезной программы уместить в одном файле, то такой код станет просто нечитаемым. К тому же если программа компилируется достаточно долго (особенно это относится к языку C++), то после исправления одной ошибки, нужно перекомпилировать весь код.

Поэтому лучше разбросать исходный код по нескольким файлам (осмысленно, по какому-нибудь критерию), и компилировать каждый такой файл отдельно. Как вы вскоре узнаете, это очень даже просто.

Давайте сначала разберемся, как из исходного файла получается бинарник. Исходный файл не сразу превращается в бинарник. После компиляции создается *объектный код*. Это исполняемый код с некоторыми "вкраплениями", из-за которых объектный код еще не способен к выполнению. Сразу в голову приходит стиральная машина: вы ее только что купили и она стоит у вас дома в коробке. В таком состоянии она стирать не будет, но вы все равно рады, потому что осталось только вытащить из коробки и подключить.

Вернемся к объектному коду. Эти самые "вкрапления" (самое главное среди них - таблица символов) позволяют объектному коду "пристыковываться" к другому объектному коду. Такой фокус делает *компоновщик* (*линковщик*) - программа, которая объединяет объектный код, полученный из "разных мест", удаляет все лишнее и создает полноценный бинарник. Этот процесс называется *компоновкой* или *линковкой*.

Итак, чтобы откомпилировать мультифайловую программу, надо сначала добыть объектный код из каждого исходного файла в отдельности. Каждый такой код будет представлять собой *объектный модуль*. Каждый объектный модуль записывается в отдельный *объектный файл*. Затем объектные модули надо скомпоновать в один бинарник.

В Linux в качестве линковщика используется программа **ld**, обладающая приличным арсеналом опций. К счастью gcc самостоятельно вызывает компоновщик с нужными опциями, избавляя нас от "ручной" линковки.

Попробуем теперь, вооружившись запасом знаний, написать мультифайловый Hello World. Создадим первый файл с именем **main.c**:

```
/* main.c */

int main (void)
{
        print_hello ();
}
```

Теперь создадим еще один файл **hello.c** со следующим содержимым:

```
/* hello.c */
#include <stdio.h>

void print_hello (void)
{
        printf ("Hello World\n");
}
```

Здесь функция main() вызывает функцию print_hello(), находящуюся в другом файле. Функция print_hello() выводит на экран заветное приветствие. Теперь нужно получить два объектных файла. Опция -c компилятора gcc заставляет его отказаться от линковки после компиляции. Если не указывать опцию -o, то в имени объектного файла расширение .c будет заменено на .o (обычные объектные файлы имеют расширение .o):

```
$ gcc -c main.c
$ gcc -c hello.c
$ ls
hello.c  hello.o  main.c  main.o
$
```

Итак, мы получили два объектных файла. Теперь их надо объединить в один бинарник:

```
$ gcc -o hello main.o hello.o
$ ls
hello*  hello.c  hello.o  main.c  main.o
$ ./hello
Hello World
$
```

Компилятор увидел, что вместо исходных файлов (с расширением .c) ему подбросили объектные файлы (с расширением .o) и отреагировал согласно ситуации: вызвал линковщик с нужными опциями.

Чтобы понять, что произошло, воспользуемся утилитой **nm**. Объектные файлы содержат *таблицу символов*. Утилита nm как раз позволяет посмотреть эту таблицу в читаемом виде. Те, кто пробовал программировать на ассемблере знают, что в исполняемом файле буквально все (функции, переменные) стоит на своей позиции: стоит только вставить или убрать из программы один байт, как программа тут же превратиться в груду мусора из-за смещенных позиций (адресов). У объектных файлов особая роль: они хранят в таблице символов имена некоторых позиций (глобально объявленных функций, например). В процессе линковки происходит стыковка имен и пересчет позиций, что позволяет нескольким объектным файлам объединиться в один бинарник. Если вызвать nm для файла hello.o, то увидим следующую картину:

```
$ nm hello.o
         U printf
00000000 T print_hello
$
```

О смысловой нагрузке нулей и литер U,T мы будем говорить при изучении библиотек. Сейчас же важным является то, что в объектном файле сохранилась информация об использованных именах. Своя информация есть и в файле main.o:

```
$ nm main.o
00000000 T main
         U print_hello
$
```

Таблицы символов объектных файлов содержат общее имя print_hello. В процессе линковки высчитываются и подставляются в нужные места адреса, соответствующие именам из таблицы. Вот и весь секрет.

```
$ gcc progr1
gcc: no input files
$
```

Настройка командного интерпретатора с помощью специальных переменных

В **ОС Linux** есть специальные переменные интерпретатора, с помощью которых осуществляется настройка интерпретатора пользователя. Часть из них определяет система при регистрации пользователя, а остальные пользователь может определить сам, изменив установленные системой по умолчанию.

Автоматическое определение специальных переменных осуществляется специальными сценариями командного интерпретатора, которые называются **файлами инициализации**. При запуске КИ выполняются соответствующие файлы инициализации. При каждой регистрации пользователя в системе, включающей КИ **BASH**, выполняется файл инициализации **.bash_profile** командного интерпретатора **BASH**. В этом файле содержатся определения специальных переменных и значения для них. Их можно изменить, редактируя файл или выполняя специальные команды из КИ **BASH**. Редактировать файлы инициализации имеет право системный администратор, а изменить специальные пользовательские переменные может пользователь из командной строки. К таким переменным относятся: **HOME**, **SHELL**, **PATH**, **PS1**, **PS2**, **MAIL**.

HOME - содержит путевое имя начального каталога пользователя;

SHELL - содержит путевое имя программы для того типа интерпретатора, в котором регистрируется пользователь;

PATH - содержит перечень каталогов, в которых выполняется поиск команд Linux;

PS1 - содержит основные символы приглашения к работе, показываемые в командной строке;

PS2 - содержит дополнительный символ приглашения, который используется для команд, состоящих из нескольких строк;

MAIL - содержит путевое имя файла почтового ящика.

Значения указанных переменных можно просмотреть, набрав в командной строке: **$ echo $имя_переменной**, например: **$ echo $MAIL**

Значения указанных переменных можно изменить, присвоив им новые значения. При этом можно использовать заранее определенный набор кодирующих символов: **\w** - показать текущий рабочий каталог, **\u** - показать имя пользователя, **\t** - показать время или **\!** - номер события в хронологическом списке, например:

$ PS1="ваш текст\u\w $",
$ PS1="\t\u ->",
$ PS2="@"

Команда сортировки sort

Команда **sort** предназначена для сортировки строк текстовых файлов по алфавиту. Для изучения команды **sort** выполнить:

Задание 1. 1. Войти в свой рабочий каталог (если Вы начали работу, то в нем окажетесь автоматически) и создать новый файл spisok, состоящий из 10 - 15 непронумерованных строк, т.е. имитирующий список каких-либо объектов, например фамилий.

Задание 2. Отсортировать список в алфавитном порядке и вывести на экран (| - обозначение так называемого программного канала - передача выходных данных команды на вход следующей команды):

$ sort spisok | cat

что аналогично выполнению последовательности двух команд:

$ sort spisok

$ cat spisok

При этом содержимое файла **spisok** не изменяется, а на экране воспроизводится отсортированный файл.

Задание 3. Отсортировать файл в обратном порядке (опция **-f** или –**r** команды **sort**), пронумеровать строки и вывести на стандартное устройство вывода - экран (опция **-n** команды **cat**), результат записать в новый файл **spisok1** (> - перенаправление стандартного вывода в файл или на устройство).

$ sort spisok | cat -n > spisok1

Задание 4. Вывести на экран содержимое файла **spisok1** и записать в файл **spisok2** с использованием перенаправления стандартного ввода-вывода, следующим образом:

$ cat < spisok1 > spisok2

Контрольное задание

Изучите элементы рабочего стола, который Вы увидели после входа в систему. Просмотрите файловую структуру ОС Linux в графическом режиме. Удобно войти в нее через ярлык **home directory**, расположенный на рабочем столе или запустить из главного меню программу **Nautilus** (рабочий стол **GNOME**) или **Konquerror** (рабочий стол **KDE**).

Просмотрите содержимое основных каталогов, доступных пользователю для чтения: **bin**, **dev**, **etc**, **lib**, **mnt**, **tmp**, **var**. Из каталога **usr** скопируйте 2 файла в Ваш рабочий каталог.

Запустите окно терминала и выполните команды определения путевого каталога, просмотра списка файлов и каталогов текущего каталога и выбранных Вами из файловой структуры. Используйте разные опции команды **ls**.

Используя текстовые редакторы **vi**, **vim**, **gvim**, **gedit** и команду **cat**, создайте по одному файлу в каждом из перечисленных текстовых редакторов, установленных в Вашей системе. Созданный в одном редакторе файл отредактируйте в другом (можно выбрать вариант: текстовый и графический редакторы).

Составьте файловую структуру, имеющую три уровня вложения и содержащую 5 - 6 созданных Вами файлов, расположив их в каталогах разных уровней. Выполните это задание в режиме консоли, в качестве примера используйте пример, представленный в конце работы. Создавая структуру, проверяйте результаты командой **ls** с различными опциями.

Используя справочник **man**, уточните опции и параметры нескольких описанных в работе команд. Выборочно проверьте некоторые из них.

Используя архив команд **history_list**, повторите несколько команд на выбор. Проверьте возможность редактирования этих команд.

Настройте командный интерпретатор с помощью специальных переменных по-другому, используя описанные выше кодирующие символы.

Изучите программу **Midnight Commander**. Ее запуск осуществляется из окна терминала: **$ mc**. Создайте подкаталог в Вашем рабочем каталоге и запишите в него 2 - 3 файла.

Выполните примеры и задания, представленные в работе.

Выполните следующие команды, используя свои файлы и каталоги (перемещения по каталогам отслеживать самостоятельно):

1. Сравнение двух файлов посимвольно: **$ cmp f1 f2**

где **f1** и **f2** - любые файлы текущего каталога. Команда **cmp** проверяет файлы до первого различия и выводит позицию отличающегося символа и строки второго файла.

2. Проверка: **файл** или **директорий**? **$ file f1 k2**

В результате выводится информация:

f1 : text

k2 : directory

3. Просмотр содержимое текущего каталога другим способом: **$ ls -F**

Результат выполнения команды: f1 k2/

4. Просмотр файла по байтам

$ od - опция

Возможные опции:

- **-c** - в символьном формате
- **-d** - в десятичном формате
- **-o** - в шестнадцатеричном формате
- **-x** - значения байтов в восьмеричном формате

5. Запись в файл и на стандартный вывод (терминал или экран):

$ tee file1

6. Использование различных опций в команде **more**, удобной для просмотра больших файлов. Просмотреть содержимое указанных файлов (без опций): **$ more f1 f2 f3 f4**, здесь **f1**, **f2**, **f3**, **f4**, - имена файлов.

Использование команды **more** с различными опциями:

Показать содержимое файла **file1**, начиная с четвертой страницы:

$ more +4 file1

Пропуск пяти страниц вперёд: **$ more 5 file1**

Пропуск двух страниц назад: **$ more - 2 file1**

Прокрутка содержимого файла по полстраницы: **$ more d file1**

Просмотр опций команды **more**: **$ more h**

q - выход из команды-утилиты **more**.

7. Копирование файла **f1** в другой файл **f5** с проверкой: существует ли уже файл **f5**?

$ cp -i f1 f5

Если файл **f5** уже существует, то на экране появляется сообщение: **overwrite f5?** (перезаписать файл **f5**?) требуется ответить: **n** - если не перезаписывать; **y** - если перезаписывать вместо предыдущего файла **f5**.

8. Аналогично, опция **i** используется для команды перемещения и удаления:

$ mv -i f3 f4

(здесь будет выведено подтверждение на перемещение файлов **f3** и **f4**)

9. **$ rm -i f5 (**будет выведено подтверждение на удаление файла **f5**).

10. Изучить команды **head** и **tail**, дающие возможность просмотреть первые или последние несколько строк файла (по умолчанию выводятся первые 10 строк файла), используя соответственно:

$ head имя_файла и **$ tail имя_файла**

Лабораторное задание и порядок выполнения работы

Изучить материал, выполняя примеры и задания, представленные в работе, и используя личные файлы и каталоги.

Выполнить контрольное задание, описанное в конце работы, используя свои файлы и каталоги.

(По решению преподавателя! Кратко законспектировать материал по новым командам.

Оформить отчет и защитить работу).

Требования к отчету

Отчет должен содержать по требованию преподавателя:

1. Описание последовательности команд создания своей файловой структуры (с Вашими именами файлов и каталогов).
2. Краткие сведения о работе и перечень опробованных в этой работе команд командного интерпретатора **BASH**.