Лабораторная работа №1 «Интерфейс. Файлы. Команды»

Часть 1. CLI — Command-Line Interface

Для его изучения включите терминал (Приложения > Стандартные > Терминал).

Работу ОС LINUX можно представить в виде функционирования множества взаимосвязанных процессов. При загрузке системы сначала запускается ядро (процесс 0), которое в свою очередь запускает командный интерпретатор shell (процесс 1).

Взаимодействие пользователя с системой LINUX происходит в интерактивном режиме посредством командного языка. Оболочка операционной системы – shell – интерпретирует вводимые команды, запускает соответствующие программы (процессы), формирует и выводит ответные сообщения.

Shell - это интерфейс, обеспечивающий взаимодействие между ядром и пользователем. Интерфейс shell очень прост. Обычно он состоит из приглашения, по которому пользователь вводит команды и нажимает клавишу Enter. Строка, в которой вы набираете команду, называется командной строкой. Shell не только интерпретирует команды, но и создает среду, которую вы можете конфигурировать и программировать. У shell есть свой язык программирования, который позволяет писать программы, содержащие достаточно сложные последовательности команд Linux. Язык программирования shell обладает многими свойствами обычного языка программирования, в частности в нем предусмотрено использование циклов и условных переходов. Каждому пользователю системы Linux предоставляется свой собственный пользовательский интерфейс, или shell. Пользователи могут модифицировать свои shell в соответствии с конкретными потребностями. В этом смысле shell пользователя функционирует скорее как операционная среда, которой пользователь может управлять по своему усмотрению.

За последние годы разработано несколько разновидностей shell. Сейчас используются в основном три варианта: Bourne, Korn и C-shell. Bourne-shell был разработан в Bell Labs для System V. C-shell разработан для версии BSD. Kom-shell - это усовершенствованный вариант Bourne-shell. В современных версиях Unix, включая Linux, представлены все три вышеназванных shell, что дает пользователю возможность выбора. В Linux, однако, используются расширенные или общедоступные версии этих shell: Bourne Again, TC-shell и Public Domain Korn. При запуске ОС Linux активизируется Bourne Again Shell, модифицированная версия Bourne. Отсюда можно переключаться в другие shell.

Файловая структура: каталоги и файлы

В операционной системе Linux все файлы организованы в каталоги, которые, в свою очередь, иерархически соединены друг с другом, образуя одну общую файловую структуру. При обращении к файлу необходимо указывать не только его имя, но и место, которое он занимает в этой файловой структуре. Можно создавать любое количество новых каталогов, добавляя их к файловой структуре. Команды работы с файлами ОС Linux могут выполнять сложные операции, например, перемещение и копирование целых каталогов вместе с их подкаталогами. Такие команды, как find, cp, mv и ln, позволяют находить файлы, копировать их и перемещать из одного каталога вдругой, а также создавать ссылки.

Файлы в операционной системе Linux организованы в иерархическую систему каталогов. Каталог может содержать файлы и другие каталоги. В этом смысле каталоги выполняют две важные функции. Во-первых, в каталоге хранятся файлы, подобно папкам в ящике картотеки, а во-вторых, каталог соединяется с другими каталогами, как ветвь дерева соединяется с другими ветвями. По отношению к файлам каталоги выполняют роль ящиков картотеки, в каждом из которых хранится несколько папок. Для того чтобы взять одну из них, нужно открыть ящик. Следует отметить, однако, что, в отличие от ящиков картотеки, каталоги могут содержать не только файлы, но и другие каталоги. Именно таким образом каталог может соединяться с другим каталогом. Из-за сходства с деревом такую структуру часто называют древовидной структурой. Если быть более точным, то эта структура скорее похожа не на дерево, а на перевернутый вверх корнями куст. Ствола здесь нет, и изображается дерево перевернутым, при этом корень находится наверху. Вниз от корня отходят ветви. Каждая ветвь отходит только от одной ветви, а от нее самой может отходить множество ветвей нижнего уровня. В этом смысле данную структуру можно назвать структурой "родители-потомки".

Аналогичным образом любой каталог является подкаталогом другого каталога. Каждый каталог может содержать множество подкаталогов, но сам должен быть потомком только одного родительского каталога. Вверху файловой системы находится корневой каталог (обозначается символом "косая черта"), от которого ответвляются другие каталоги. Каждый каталог может содержать несколько других каталогов или файлов, но родительский каталог у него всегда бывает только один.

В каталоге chris, например, организованы два подкаталога, reports и programs. Сам же каталог chris соединен только с одним родительским каталогом, home. Файловая структура ОС Linux разветвляется на несколько каталогов, начиная с корневого, /. В корневом каталоге имеется несколько системных каталогов, которые содержат файлы и программы, относящиеся к самой ОС Linux. Корневой каталог, кроме того, содержит каталог home, который может содержать начальные каталоги всех пользователей системы. Начальный каталог каждого пользователя, в свою очередь, будет включать в себя каталоги, который пользователь создает для своих нужд. Каждый из этих каталогов тоже может содержать каталоги. Все эти вложенные каталоги ответвляются от начального каталога пользователя.

Получить доступ к каталогу можно либо по имени, либо сделав его каталогом по умолчанию. Каждому каталогу при создании присваивается имя. Этим именем можно пользоваться для доступа к файлам, находящимся в данном каталоге. Если при проведении какой-либо операции над файлами имена каталогов не указываются, то используется каталог по умолчанию, который называют рабочим каталогом. В этом смысле рабочий каталог - это каталог, в котором вы в данный момент работаете. При регистрации в системе в качестве рабочего принимается ваш начальный каталог, имя которого обычно совпадает с вашим регистрационным именем.

Рабочий каталог можно заменить с помощью команды cd. В процессе замены рабочего каталога вы переходите из одного каталога в другой. Каталог можно рассматривать как коридор, в который выходит множество дверей с табличками. Некоторые двери ведут в комнаты, а некоторые - в другие коридоры. Двери, ведущие в комнаты, - это файлы, находящиеся в каталоге, а двери, ведущие в коридоры, - это другие каталоги. Переходя из одного коридора в другой, вы меняете рабочий каталог. Проходя по нескольким коридорам, вы перемещаетесь по нескольким каталогам.

Путевые имена

Имя, которое дается каталогу или файлу при его создании, не является полным. Полным именем каталога является его путевое имя. Иерархические связи, существующие между каталогами, образуют пути, и эти пути можно использовать для однозначного указания каталога или файла и обращения к нему.

Можно сказать, что каждый каталог в файловой структуре имеет собственный уникальный путь. Фактическое имя, которым система обозначает каталог, всегда начинается с корневого каталога и состоит из имен всех каталогов, ведущих к данному каталогу.

В ОС Linux путевое имя каталога состоит из имен всех каталогов, образующих путь. Эти имена отделяются друг от друга символами "косая черта". Косая черта перед первым каталогом пути обозначает корневой каталог(/).

Путевые имена могут быть абсолютными и относительными. Абсолютное путевое имя - это полное имя файла или каталога, начинающееся символом корневого каталога. Относительное путевое имя начинается символом рабочего каталога и представляет собой обозначение пути к файлу относительно вашего рабочего каталога.

Системные каталоги

Корневой каталог, являющийся началом файловой структуры ОС Linux, содержит ряд системных каталогов. Системные каталоги содержат файлы и программы, служащие для управления системой и ее сопровождения. Многие из этих каталогов содержат подкаталоги с программами, предназначенными для выполнения конкретных задач.

/bin

bin - это сокращенно от `binaries' (т.е. двоичные или выполняемые файлы). Здесь находится много важных системных программ. Большинство основных команд Unix находятся в этом каталоге.

/dev

"Файлы" в dev известны как драйверы устройств - они используются для доступа к устройствам и ресурсам системы, таким как диски, модемы, память и т.д. Например, вы можете читать данные из файла, точно также вы можете читать входные сигналы от мыши, имея доступ к /dev/mouse. Имена файлов, начинающиеся на fd - это дисководы гибких дисков. fd0 - первый дисковод, fd1 - второй.

Различные /dev/ttys, /dev/cua устройства используются для доступа к последовательным портам. Например, /dev/ttys0 относится к `COM1' под MS-DOS. Устройства /dev/cua относятся к "звонящим" (``callout") устройствам, которые используются совместно с модемами.

Устройства, имена которых начинаются с hd, имеют доступ к жестким дискам. /dev/hda относится ко всему первому жесткому диску, а hda1 только к первому разделу /dev/hda.

Устройства с именами /dev/tty относятся к "виртуальным консолям" вашей системы (доступ путем нажатия alt-F1, alt-F2 и т.д.). /dev/tty1 соответствует первой, /dev/tty2 соответствует второй и т.д.

Устройства, чьи имена начинаются на /dev/pty, это "псевдотерминалы". Они используются для входа с удаленных "терминалов". Например, если ваша машина в сети, вход к вам по telnet будет использовать одно из устройств /dev/pty.

/etc

etc содержит файлы конфигурации системы. Например /etc/passwd(файл паролей), /etc/groups(файл групп), /etc/rc (командный файл инициализации) и т.д.

/sbin

В sbin находятся важные исполняемые системные файлы, используемые системным администратором.

/home

home содержит домашние каталоги пользователей.

/lib

lib содержит образы разделяемых библиотек (shared library images). Эти файлы содержат код, который могут использовать многие программы. Вместо того, чтобы каждая программа имела свою собственную копию этих выполняемых файлов, они хранятся в одном общедоступном месте – в /lib. Это позволяет сделать выполняемые файлы меньше и сэкономит место в системе.

/proc

proc - это "виртуальная файловая система" procfs, в которой файлы хранятся в памяти, а не на диске. Они связаны с различными процессами, происходящими в системе, и позволяют получить информацию о том, что делают программы и процессы в указанное время.

/tmp

Многие программы нуждаются в создании рабочих файлов, которые нужны короткое время. Каноническое место для этих файлов в /tmp (там обычно чаще проводится уборка мусора).

/usr

usr - состоит из ряда подкаталогов, которые в свою очередь содержат наиболее важные и полезные программы и файлы конфигурации, используемые системой. Различные каталоги, описанные выше, необходимы для нормального функционирования системы, но большинство вещей, содержащихся в /usr необязательны для системы. Но это такие необязательные вещи, которые делают систему полезной и интересной.

/usr/X11R6 - содержит The X Window System, если вы ее инсталлировали.

/usr/bin - для различных программ UNIX. Он содержит большинство выполняемых программ, которых нет ни в каких других местах, например, в том же /bin их нет.

/usr/etc – также как и /etc, содержит всевозможные системные программы и конфигурационные файлы.

/usr/include - содержит include-файлы(header - файлы) для компилятора Си.

/usr/lib - содержит библиотеки -"заглушки" и "статические" библиотеки, эквивалентные файлам из /lib. При компиляции программа "связывается" с библиотеками, находящимися в /usr/lib, которые в свою очередь направляют программы обращаться в /lib, если им нужен актуальный код. Кроме того, многие другие программы хранят в /usr/lib свои конфигурационные файлы.

/usr/local - в большой степени похож на /usr - он содержит различные программы и файлы, несущественные для системы /usr/man - содержит страницы Руководства. Здесь два подкаталога для каждого "раздела" Руководства. (С помощью команды "man man" вы можете получить более подробную информацию). Например, /usr/man/man1 содержит исходные тексты (неотформатированный оригинал) страниц Руководства в разделе 1 и usr/man/cat1 содержит отформатированные страницы для раздела 1.

/usr/src - содержит исходные коды (неоткомпилированные программы) для различных программ вашей системы. Наиболее важная вещь здесь это каталог /usr/src/linux, в котором содержатся исходные коды ядра Linux.

/var

var содержит каталоги, которые часто меняются в размере или имеют тенденцию быстро расти. К числу таких каталогов относятся:

/var/adm - содержит различные файлы, интересные системному администратору, специфические системные файлы, фиксирующие ошибки и проблемы, возникающие в системе. Другие файлы фиксируют входы в систему, как и неудачные попытки войти.

/var/spool - содержит файлы, которые предварительно формируются для других программ. Например, если ваша машина подключена к сети, входная почта будет помещаться в /var/spool/mail до тех пор, пока вы не прочитаете ее или не удалите.

Входящие и исходящие новости помещаются в /var/spool/news и т.д. Зарегистрировавшись системе, пользователь попадает в свой начальный каталог. Имя, присвоенное этому каталогу системой, совпадает с регистрационным именем пользователя. Все файлы, создаваемые для нового пользователя, помещаются в начальный каталог. В этом каталоге можно создавать подкаталоги и размещать в них файлы.

Команды

Прежде чем перейти к рассмотрению конкретных команд, дадим определение команде. Пользователям, вышедшим из среды DOS, это понятие знакомо: команда - основа главных функций операционной системы. Из команд DIR, COPY или ATTRIB составляются довольно сложные процедуры, оформляемые в виде bat-файлов (командных файлов).

Однако в DOS, как и в других операционных системах, количество команд ограниченно и статично — пользователь не может вводить собственные команды.

В мире Unix (следовательно, и Linux) понятие команды несколько иное. Здесь команда - это любой выполняемый файл. Командой является любой файл, предназначенный для выполнения, а не для хранения данных или конфигурационных параметров. Любой выполняемый файл, записанный в систему, становится ее командой.

Коротко перечислим средства группирования команд:

- *cmd1 arg ...; cmd2 arg ...; ... cmdN arg ...* - последовательное выполнение команд;
- *cmd1 arg ...& cmd2 arg ...& ... cmdN arg ...* - асинхронное выполнение команд;
- *cmd1 arg ... && cmd2 arg ...* - зависимость последующей команды от предыдущей таким образом, что последующая команда выполняется, если предыдущая выдала нулевое значение;
- *cmd1 arg ... || cmd2 arg ...* - зависимость последующей команды от предыдущей таким образом, что последующая команда выполняется, если предыдущая выдала ненулевое значение.

Управление каталогами

Создание и удаление каталогов: mkdir и rmdir

Каталоги создаются и удаляются соответственно командой mkdir и командой rmdir. В том и другом случае можно использовать путевые имена каталогов. В следующем примере пользователь сначала создает каталог reports, а затем, используя абсолютное путевое имя - каталог letters. (Здесь и далее символ $ означает приглашение ко вводу и не относится к командам).

```
$ mkdir reports
$ mkdir /home/chris/letters
```

Для удаления каталога нужно дать команду rmdir с именем этого каталога. В приведенном ниже примере пользователь сперва удаляет командой rmdir каталог reports, а затем - указав абсолютное путевое имя - каталог letters.

```
$ rmdir reports
$ rmdir /home/chris/letters
```

Просмотр содержимого каталогов: ls

Чтобы с помощью команды ls можно получить список файлов и каталогов, находящихся в рабочем каталоге, необходимо выполнить следующую команду.

```
$ ls
```

Для того чтобы имена файлов и имена каталогов различались между собой, эту команду нужно дать с опцией -F. В этом случае после каждого имени каталога в списке ставится косая черта.

```
$ ls
weather reports letters
$ ls -F
weather reports/ letters/
```

В качестве аргумента команда ls может использовать имя или путевое имя каталога. Это позволяет получить список файлов любого каталога, не переходя в него. В следующем примере команда ls использует в качестве аргумента имя каталога reports. Затем она выполняется еще раз, но уже с абсолютным путевым именем этого каталога.

```
$ ls reports
monday tuesday
$ ls /home/chris/reports
monday tuesday
```

Переход в другой каталог: команда cd

Переход из одного каталога в другой осуществляется командой cd. Переход в каталог делает его рабочим. Файловые команды, например Ls, будут манипулировать файлами, находящимися именно в рабочем каталоге, если иного не указано.В качестве аргумента команда cd использует имя каталога, в который вы хотите перейти.

```
$ cd имя_каталога
```

Все создаваемые каталоги будут находиться в рабочем каталоге. Рабочий каталог является для вновь созданного каталога родительским. Для обозначения родительского каталога можно пользоваться двумя точками (..). Этот специальный символ обозначает путевое имя родительского каталога. Его допускается использовать в команде cd для перехода обратно в родительский каталог, таким образом вновь делая этот каталог рабочим.

Если вы хотите вернуться в начальный каталог, нужно ввести команду cd без аргумента. Вы вернетесь прямо в свой начальный каталог, и он вновь станет рабочим.

Путевые имена: команда pwd

В каждом каталоге можно создавать другие каталоги, осуществляя, по сути дела, вложение одного каталога в другой. Команда cd позволяет переходить из одного каталога в другой, однако, никакого указателя на то, в каком каталоге вы в данный момент находитесь, нет. Для того чтобы определить, в какой каталог вы перешли, дайте команду pwd, которая сообщит абсолютное путевое имя рабочего каталога, как показано в следующем примере. Путевое имя состоит из имен рабочего каталога dylan и каталога, частью которого он является, home. Имена каталогов разделены косой чертой. Корневой каталог обозначен первой косой чертой.

```
$ pwd
/home/dylan
```

Обращение к рабочему и родительскому каталогам: . и ..

Каждый каталог обязательно имеет родительский каталог (за исключением, естественно, корневого каталога). При создании каталога в нем сразуже делаются две записи. Одна из них будет представлена точкой (.), а вторая - двумя точками (..). Точка обозначает путевое имя данного каталога, а две точки - путевое имя его родительского каталога. Две точки, используемые как аргумент команды, обозначают родительский каталог. Одна точкаобозначает рабочий каталог.

Точка используется для обозначения рабочего каталога вместо указания его путевого имени. Например, для копирования файла в рабочий каталог с сохранением имени файла можно вместо путевого имени рабочего каталога поставить точку. В этом смысле точка - еще одно имя рабочего каталога.

Символ .. часто используется для обозначения файлов родительского каталога. Используя команду cd с символом .., можно возвращаться из каталога нижнего уровня, последовательно переходя в родительские каталоги по дереву каталогов.

Во многих случаях в команде допускается использование обоих символов. Например, если letters - рабочий каталог и нужно скопировать в него файл weather, то каталог chris можно обозначить двумя точками, а каталог letters - одной:

```
$ cp ../weather .
```

Использование абсолютных и относительных путевых имен: ~

Как упоминалось выше, файлы и каталога можно обозначать абсолютными и относительными путевыми именами. У обоих вариантов, однако, есть свои недостатки. Абсолютное путевое имя пригодно для обозначения любого файла и каталога, но такие имена, как правило, очень длинные и сложные, что затрудняет работу с ними. Относительное путевое имя короче и проще в работе, но число файлов, которые им можно обозначить, ограниченно. Как правило, относительные путевые имена нужно использовать при каждой возможности, а абсолютные - только в случае необходимости. В некоторых shell предусмотрена возможность сокращения абсолютных путевых имен.

Относительные путевые имена применяют для обозначения только файлов, находящихся в подкаталогах рабочего каталога. Этих подкаталогов, вложенных один в другой, может быть сколь угодно много, но их пути должны ответвляться от рабочего каталога. Допустим, вам нужно обратиться к каталогу, расположенному по дереву каталогов выше рабочего или в другой ветви, тогда необходимо использовать абсолютное путевое имя.

Диалоговое руководство

В системе Linux используются различные утилиты, среди которых редакторы, программы-почтальоны и руководства. Эти утилиты представляют собой отдельные программы, имеющие собственные интерфейсы с собственными наборами команд. Примером такой утилиты является диалоговое руководство man, которое позволяет пользователю получить информацию о любой команде и программе ОС Linux. Для обращения к диалоговому руководству введите команду man и имя команды, информация о которой вам нужна. Ниже приведен пример, в котором пользователь вызывает из диалогового руководства информацию о команде ls ,

```
$ man ls
```

После нажатия клавиши Enter вы попадаете в утилиту man, которая выдает первую страницу документа о команде ls. В утилите man используется собственный набор команд, для задания которых, как правило, достаточно нажатия одной клавиши. Нажатие клавиши пробела или клавиши f выводит следующую страницу. Нажатие клавиши b возвращает вас на предыдущую страницу. Закончив работу, выйдите из утилиты и вернитесь в командную строку (нажатием клавиши q). Описание команд в руководстве состоит из нескольких частей. Чаще всего их пять: синопсис, описание, опции, файлы и перекрестные ссылки. Синопсис содержит синтаксис команды с указанием ее опций и аргументов. В описании команды рассказывается, для чего конкретно она применяется в системе. Затем перечисляются и поясняются опции. В следующей части перечисляются системные файлы, которые использует команда, а в списке перекрестных ссылок указываются родственные команды и пункты руководства. Ниже приведен сокращенный вариант страницы руководства, посвященной команде ls.

```
LS(1L)
LS(1L)
NAME
     ls, dir, vdir - list contents of directories
SYNOPSIS
     Is [-abcdfgiklmnpqrstuxABCFGLNQRSUXl] t-w cols]  [-T cols]
     [-1 pattern] [--all] [--directory] [--inode] [--kilobytes]
     [--no-group] [--hide-control-chars] [--reverse] [--size]
     [--width=cols] [--sort^fnone,time,size,extension}]
DESCRIPTION
     This manual page documents the GNU version of Is. dir and
     vdir are versions of Is with different default output for-
     mats. These programs list each given file or directory
     name. Directory contents are sorted alphabetically. For
     Is, files are by default listed in columns, sorted verti-
     cally, if the standard output is a terminal; otherwise
     they are listed one per line. For dir, files are by
     default listed in columns, sorted vertically. For vdir,
     files are by default listed in long format.
OPTIONS
     -a, --all
           List all files in directories, including all files
           that start with '.'.
     -b, --escape
           Quote nongraphic characters in file names using
           alphabetic and octal backslash sequences like those
           used in C.
     -c, --time=ctime, --time=status
           Sort directory contents according to the files'
           status change time instead of the modification
           time. If the long listing format is being   used,
           print the status change time.instead of the modifi-
           cation time.
     -d, --directory
           List directories like other files, rather than
           listing their contents.
     -f    Do not sort directory contents; list them in what-
           ever order they are stored on the disk. The   same
           as enabling -a and -U and disabling -1, -s, and -t.
     --full-time
           List times in full, rather than using the standard
FSF                  GNU File Utilities                      1
```

Утилита man имеет несколько полезных особенностей, в частности она позволяет проводить поиск. Эта функция активизируется нажатием либо клавиши /, либо клавиши ?. Первый вариант предусматривает поиск вперед, а второй - поиск назад. После нажатия клавиши / в нижней части экрана появляется строка, в которую нужно ввести искомое слово. Затем нужно нажать Enter. Поиск осуществляется по образцу, поэтому можно ввести часть слова или практически любой набор символов. Повторение поиска осуществляется нажатием клавиши п. Повторно вводить образец не нужно.

Команды whatis и apropos

Команды whatis и apropos обеспечивают поиск в базе данных заголовков man-страниц и выдают все найденные заголовки с кратким описанием каждого. Команда whatis позволяет искать заголовки по целым словам. Например, если вы хотите увидеть все пункты руководства, в которых есть отдельно стоящая буква x, необходимо дать следующую команду (она выдаст вам все пункты руководства, в названиях которых упоминается X Window):

```
$ whatis x
X (3)                - a portable, network-transparent window
system
X Consortium (3)   - X Consortium information
X Standards (3)    - X Consortium Standards
X security (3)     - x display access control
x (3)                  - a Portable, network-transparent window
system
X Consortium (3)   - X Consortium information
X Standards (3)    - x Consortium Standards
X security (3)     - X display access control
(END)
$
```

С помощью этих команд осуществляется постраничный вывод результатов поиска. Если выводимые данные занимают несколько страниц можно перемещаться по ним с помощью клавиш f и Ь. Допускается и выполнение поиска по образцу (клавишами / и ?). Для выхода нажмите клавишу q Лишь после этого вы вернетесь в командную строку.Команда apropos выполняет ту же задачу, что и команда whatis но поиск выполняется по образцу, а не по целым словам. Скажем, команда apropos x выдаст несколько страниц с данными, в которых будут перечислены все пункты руководства, начинающиеся с буквы x, например xwpe и xloadimage. В следующем примере выдается перечень пунктов руководства начинающихся с комбинации ls. Сюда входит команда ls и другие команды, например lseek и lsearch.

```
$ apropos ls
ls, dir, vdir (1)    - list contents of directories
lsattr (1)           - list file attributes on a Linux second
                       extended file system
lsearch (n)             - see if a list contains a particular
element
lseek (2)            - reposition read/write file offset
lsort (n)            - sort the elements of a list
lsattr (1)           - list file attributes on a Linux second
                       extended file system
lsearch (n)             - See if a list contains a particular
element
lseek (2)            - reposition read/write file offset
lsort (n)            - Sort the elements of a list
(END)
$
```

Отображение файлов: cat и more

Во многих случаях бывает необходимо просматривать содержимое файла. Команды cat и more выводят содержимое файла на экран. Название команды cat образовано путем сокращения слова concatenate. Это очень сложная и универсальная

команда. В нижеследующем примере употребляется в очень узких рамках, только для вывода текста файла на экран:

```
$ cat mydata
computers
```

Команда cat выводит на экран сразу весь текст файла. Если файл имеет большой размер, то текст очень быстро мелькает на экране. Для устранения этого недостатка служит команда more, с помощью которой текст на экран можно выводить порциями. Эта команда вызывается с именем файла, который вы хотите просмотреть:

```
$ more mydata
```

Когда more вызывает файл, отображается его первый фрагмент, умещающийся на экране. Для отображения следующего фрагмента нажимается клавиша f или клавиша пробела. Для возврата к предыдущему тексту используется клавиша b. Нажав клавишу q, можно в любой момент выйти из данной программы.

Печать файлов: команды lpr, lpq и lprm

Если нужно напечатать файл, перешлите его на принтер, подключенный к вашей системе. Это делается при помощи команды lpr. В следующем примере пользователь дает команду печатать файл mydata.

```
$ lpr mydata
```

Если вы хотите одновременно напечатать несколько файлов, укажите их имена в командной строке. В следующем примере пользователю необходимо печатать файлы mydata и preface:

```
$ lpr mydata preface
```

Задания на печать ставятся в очередь и выполняются в фоновом режиме. Пока они выполняются, вы делаете другую работу. Команда lpq позволяет в любой момент проверить ход выполнения заданий на печать. С ее помощью на экран выводятся имя владельца задания (регистрационное имя пользователя, который послал это задание), идентификатор задания, его размер в байтах и имя временного файла, в котором оно в данный момент находится. В нашем примере владелец - chris, а идентификатор задания ~ 00015:

```
$ lpq
Owner       ID          Chars       Filename
Chris       00015       360          /usr/lpd/cfa00015
```

Операции с файлами и каталогами: find, cp, mv, rm, ln

По мере создания файлов возникает необходимость снятия с них резервных копий, изменения их имен, удаления некоторых из них и даже присваивания им дополнительных имен. В ОС Linux предусмотрен набор команд, которые обеспечивают поиск, копирование, переименование и удаление файлов. Команды представляют собой сокращенную форму слов, состоящую из двух символов. Команда cp обозначает "copy" и позволяет копировать файл, mv обозначает "move" и дает возможность перемещать либо переименовывать файл, rm обозначает "remove" и приводит к удалению файла и, наконец. Ln обозначает "link" и позволяет дать файлу еще одно имя. Исключение из этого правила - команда find, с помощью которой осуществляется поиск файла в списке имен.

Поиск в каталогах: команда find

Если вы используете много файлов, разбросанных по разным каталогам, то для выявления одного из них или нескольких файлов определенного типа можно провести

поиск. Эта функция осуществляется с помощью команды find. В качестве аргументов в ней используются имена каталогов, за которыми следуют несколько опций, задающих тип и критерии поиска. Команда find позволяет производить поиск в перечисленных каталогах и их подкаталогах, отыскивая файлы, соответствующие указанным критериям. Команда find дает возможность искать файлы по имени, типу, владельцу и даже по времени послед него изменения.

```
$ find список каталогов -опция критерии
```

Копирование файлов

Для того чтобы создать копию файла, нужно указать команде cp два имени файла. Первое из них - имя копируемого файла, который уже существует. Этот файл часто называют исходным. Второе - имя, которое вы хотите присвоить копии. Это будет новый файл, содержащий копию всех данных исходного файла. Его часто называют выходным файлом. Команда cp имеет следующий синтаксис:

```
$ cp исходный файл выходной файл
```

В следующем примере пользователь копирует файл proposal в новый файл, oldprop.

```
$ cp proposal oldprop
```

Когда пользователь запросит перечень файлов, содержащихся в каталоге, среди них будет новая копия.

```
$ ls proposal oldprop
```

Может случиться так, что при копировании файла с помощью команды cp вы непреднамеренно разрушите другой файл. При создании копии посредством этой команды сначала создается файл, а затем в него копируются данные. Если какой-нибудь файл уже имеет то имя, которое вы указали для выходного файла, первый из них разрушается и создается новый файл с этим именем. В некотором смысле можно сказать, что файл-оригинал перезаписывается новой копией. В следующем примере файл proposal перезаписывается новой копией (потому что файл с таким именем уже существовал).

```
$ cp newprop proposal
```

Чтобы выявить подобные случаи лучше пользоваться командой cp с опцией -i. Такая команда сначала проверяет, существует ли файл под указанным именем. Если да, то программа спросит у вас, хотите ли вы перезаписать этот файл. Если вы ответите y, то существующий файл будет разрушен, и программа создаст новый файл в качестве его копии. Если вы дадите другой ответ, он будет считаться отрицательным и выполнение команды cp будет прервано, а файл-оригинал сохранен.

```
$ cp -i newprop proposal
Overwrite proposal? n
$
```

Копирование файлов в каталоги

Для того чтобы скопировать файл из рабочего каталога в другой каталог, нужно указать имя этого каталога команде cp в качестве второго аргумента. Имя новой копии будет таким же, как у оригинала, но находиться она будет в другом каталоге. Файлы в разных каталогах могут иметь одинаковые имена.

```
$ cp имена_файлов имя_каталога
```

Для того чтобы скопировать файл из начального каталога в подкаталог, просто укажите имя этого каталога. В следующем примере файл newprop копируется из рабочего каталога в каталог props.

```
$ cp newprop props
```

Команда cp может использовать в качестве аргументов имена многих файлов, заданные в виде списка, поэтому можно одновременно копировать в каталог несколько файлов. Введите имена этих файлов в командной строке, причем имя каталога должно быть последним аргументом. Все эти файлы копируются в указанный каталог. В следующем примере пользователь копирует файлы preface и doc1 в каталог props. Обратите внимание: props - последний аргумент.

```
$ cp preface doc1 props
```

При создании списка имен файлов для команды cp или команды mv можно использовать любые специальные символы. Пусть, например, вам нужно скопировать в заданный каталог все файлы с исходными текстами программ, написанными на языке С. Вместо того чтобы указывать в командной строке все эти файлы, можно ввести специальный символ * с расширением.с, обозначая тем самым все файлы с расширением.с (т.е. все файлы исходных текстов С-программ) и формируя их список. В следующем примере пользователь копирует все файлы исходных текстов программ из текущего каталога в каталог sourcebks.

```
$ cp *. c sourcebks
```

Если вы хотите скопировать все файлы из одного каталога в другой, можно при помощи обозначения *. * получить список всех файлов (имеющих расширение или в имени которых есть точка). В следующем примере пользователь копирует все файлы из каталога props в каталог oldprop. Обратите внимание на использование путевого имени props перед специальными символами *.*. В данном контексте props - это путевое имя, которое будет вставлено перед каждым именем файла в списке, создаваемом за счет использования спецсимволов *.

```
*.
$ cp props/*.* oldprop
```

Допускается использование и других специальных символов, например ? и [ ]. В приведенном ниже примере пользователь копирует файлы исходного кода и файлы объектного кода (.с и.о) в каталог projbk.

```
$ cp *. [oc] projbk
```

При копировании файла можно дать копии имя, отличное от имени оригинала. Для этого нужно поместить новое имя файла после косой черты, следующей вслед за именем каталога.

```
$ cp имя_файла имя_каталога/новое_имя файла
```

В следующем примере файл newprop копируется в каталог props и копии присваивается имя version1. Затем пользователь переходит в каталог props и получает список файлов. В нем имеется только один файл, который называется version1.

```
$ cd newprop props/version1
$ cd props
$ ls version1
```

Если нужно скопировать файл из дочернего каталога, например, из props, в родительский каталог, нужно указать имя этого дочернего каталога. Первый аргумент команды cp - имя копируемого файла. Перед ним должно через косую черту стоять имя

дочернего каталога. Второй аргумент - имя, которое файл будет иметь в родительском каталоге.

```
$ cp имя_дочернего_каталога/имя_файла новое_имя_файла
```

В следующем примере файл version1 копируется из каталога props в начальный каталог:

```
$ cp props/version1 version1
```

Предположим теперь, что вы хотите скопировать файл из дочернего каталога в родительский. Вам нужно как-то указать на этот родительский каталог. Это можно сделать двумя точками, которые обозначают путевое имя родительского каталога:

```
$ cp имя файла ..
$ cp имя_файла .. /новое__имя_файла
```

Например, если props - ваш текущий рабочий каталог и вы хотите скопировать файл version1 из props в его родительский каталог (в данном случае в начальный каталог пользователя), нужно вместо второго аргумента команды cp использовать двойную точку.

```
$ cp version1 ..
```

Если вы хотите дать копии файла version1 новое имя, добавьте его ко второму аргументу через косую черту:

```
$ cp version1 ../newversion
```

Перемещение файлов

С помощью команды mv можно либо изменить имя файла, либо переместить файл из одного каталога в другой. Используя mv для переименования файла, в качестве второго аргумента нужно указать новое имя файла. Первый аргумент - текущее имя файла.

```
$ mv текущее имя_файла новое_имя_файла
```

В следующем примере имя файла proposal меняется на version1.

```
$ mv proposal version1
```

Как и при использовании команды cp, здесь можно очень просто совершить ошибку, удалив нужный файл. Переименовывая файл, вы можете выбрать имя, которое уже носит другой файл, и этот файл будет удален. Команда mv тоже имеет опцию -i, которая сначала проверяет, существует ли файл с указанным именем. Если да, программа спросит, хотите ли вы перезаписать его. В следующем примере файл с именем version1 уже существует. Программа обнаруживает, что будет осуществлена перезапись, и спрашивает, хотите вы это сделать или нет.

```
$ ls
proposal version1
$ mv -i version1 proposal
Overwrite proposal? n
$
```

Файл можно перенести из одного каталога в другой. Для этого нужно в качестве второго аргумента в команде mv поставить имя каталога. В данном случае можно считать, что команда mv не переименовывает файл, а просто перемещает его из одного каталога в другой.

После перемещения файла у него останется то имя, которое он носил в исходном каталоге (если вы не укажете иного).

```
$ mv имя_файла имя_каталога
```

В следующем примере файл newprop перемещается из начального каталога в каталог props.

```
$ mv newprop props
```

Если при перемещении файла вы хотите переименовать его, укажите новое имя файла после имени каталога. Имя каталога отделяется от нового имени файла косой чертой. В следующем примере файл newprop перемещается в каталог props и получает имя version1.

```
$ mv newprops props/version1
$ cd props
$ ls version1
```

Указав имя дочернего каталога перед именем файла, его можно переместить из этого каталога обратно в родительский.

```
$ mv props/version1 version1
```

Предположим теперь, что вы сделали рабочим каталогом дочерний и хотите переместить файл из дочернего каталога в родительский. Как и в случае использования команды cp, можно двумя точками обозначить родительский каталог.

```
$ mv имя файла ..
$ mv имя файла ../новое имя файла
```

Например, если props - ваш текущий рабочий каталог и вы хотите переместить файл version1 из props в его родительский каталог (в данном случае в начальный каталог пользователя), нужно вместо второго аргумента команды mv использовать две точки.

```
$ mv version1 ..
```

Если вы хотите дать файлу version1 в родительском каталоге новое имя, добавьте его ко второму аргументу через косую черту:

```
$ mv version1 ../oldprop
```

Фактически имя файла - это имя, предваренное путевым именем его каталога. При перемещении файла tuesday в каталог reports путевое имя изменилось. Полное имя файла tuesday изменилось с /home/chris/tuesday на /home/chris/reports/tuesday. Теперь его путевое имя включает название каталога reports.

Столь же свободно можно использовать абсолютное путевое имя. В следующем примере файл today перемещается в каталог reports и получает новое имя, tuesday. Обратите внимание: абсолютное путевое имя используется в качестве аргумента и в команде mv, и в команде ls.

```
$ mv today /home/chris/reports/tuesday
$ ls /home/chris/reports
monday tuesday
$
```

Как и команда cp, команда mv позволяет одновременно переместить из одного каталога в другой несколько файлов. Нужно только ввести имена этих файлов в командной строке. Последним всегда должно стоять имя нового каталога. В следующем примере пользователь перемещает файлы Wednesday и friday в каталог lastweek.

```
$ cp
wednesday friday lastweek
```

При создании списка имен файлов для команды mv можно использовать любые специальные символы. В следующем примере пользователь перемещает все файлы исходных текстов программ из текущего каталога в каталог newproj.

```
$ mv *.c newproj
```

Если вы хотите переместить все файлы из данного каталога в другой каталог, можно использовать обозначение *.* и получить список всех этих файлов. В следующем примере пользователь перемещает все файлы из каталога reports в каталог repbks.

```
$ mv reports/*.* repbks
```

Перемещение и копирование каталогов

Система Linux позволяет копировать и перемещать целые каталоги. В качестве первого аргумента команды cp и mv могут использовать имя каталога, позволяя копировать и перемещать подкаталоги из одного каталога в другой. Первый аргумент - имя перемещаемого или копируемого каталога, а второй - имя каталога, в который он будет помещен. При перемещении и копировании каталогов действует та же структура путевых имен, что и при соответствующих операциях с файлами.

Подкаталоги можно так же легко, как и файлы, копировать из одного каталога в другой. Для копирования каталога команду cp необходимо использовать с опцией -r (сокращение от recursive, т.е. "рекурсивный"). Эта опция дает команде cp указание копировать каталог вместе со всеми его подкаталогами. Другими словами, копируется все поддерево каталогов, начиная с указанного. В следующем примере каталог thankyou копируется в каталог oldletters. После завершения этой операции начинают равноправно сосуществовать два подкаталога thankyou: один в каталоге letters, другой в oldletters.

```
$ cp -r letters/thankyou oldletters
$ ls -F letters
thankyou/
$ ls -F oldletters
thankyou/
```

Предположим, вы хотите скопировать не каталог, делая его тем самым подкаталогом другого каталога, а только все его файлы. Для копирования всех файлов из одного каталога в другой нужно указать имена этих файлов. Специальный символ * обозначает имена всех файлов и каталогов в данном каталоге. Для того чтобы скопировать все файлы из каталога letters в каталог oldletters, нужно в качестве первого аргумента поставить звездочку, и программа создаст список всех имен файлов, имеющихся в каталоге letters. Если нужно указать путевое имя первого аргумента, сделайте это, а звездочку поставьте в конце. В следующем примере все файлы из каталога letters копируются в каталог oldletters. Для letters указано путевое имя, а звездочка в конце этого имени обозначает все файлы в данном каталоге.

```
$ cp letters/* oldletters
```

Если вы хотите, чтобы операция копирования осуществлялась и над подкаталогами, нужно указать опцию -r.

```
$ cp -r letters/* oldletters
```

Специальный символ ~

Вы уже знаете, как обозначать тильдой абсолютное путевое имя начального каталога. Например, при копировании файла из нижестоящего каталога в начальный каталог тильдой можно обозначить абсолютное путевое имя начального каталога. В приведенном ниже примере пользователь переходит в каталог reports, а затем копирует из него файл monday в начальный каталог.

```
$ cd reports
$ cp monday ~
```

Для того чтобы при копировании файла в начальный каталог дать ему новое имя, поставьте новое имя после пары символов ~/. В следующем примере файл monday копируется в начальный каталог, и копия получает имя today.

```
$ cp monday ~/today
```

В аргументах команды mv тильда используется точно так же. Ниже показано, как файл monday перемещается из каталога reports в начальный каталог.

```
$ mv monday ~
```

Если при перемещении файла из нижестоящего каталога в начальный вы меняете его имя, то перед новым именем файла ставится тильда с косой чертой, ~/. В следующем примере пользователь переходит в каталог reports, а затем перемещает файл monday в начальный каталог и дает ему имя today.

```
$ cd reports
$ mv monday ~/today
```

Тильду можно использовать во всех случаях, когда речь идет о путевом имени начального каталога. В приведенном ниже примере описанные ранее команды mv и ls выполняются с тильдой.

```
$ mv weather ~/reports/monday
$ ls ~/reports
monday
$
```

Удаление файла: команда rm

В процессе работы с ОС Linux число используемых файлов будет стремительно возрастать. Появляются новые файлы в этой системе очень часто. Многие из них создаются при работе различных приложений, скажем, редакторов, и с помощью команд, например ср. Постепенно некоторые из этих файлов устаревают. Их можно удалить посредством команды rm. В следующем примере пользователь удаляет файл oldprop.

```
$ rm oldprop
```

Команда rm может быть использована с любым числом аргументов, что позволяет одновременно удалять несколько файлов. Имена этих файлов указываются в командной строке после имени команды.

```
$ rm proposal version1 version2
```

Командой rm следует пользоваться осторожно, так как отменить ее действие нельзя. Если файл удален, восстановить его не удастся. Предположим, что вы случайно ввели эту команду вместо какой-то другой, например, ср или mv. Когда вы опомнитесь, будет слишком поздно - файлы пропали. Для того чтобы избежать таких ошибок, используйте команду rm с опцией -i, которая инициирует выдачу запроса на подтверждение удаления. Теперь перед удалением каждого файла система будет спрашивать, действительно ли вы хотите удалить его. Если вы введете у, файл будет удален. При любом ином ответе файл не удаляется. В следующем примере посредством команды rm система получает указание удалить файлы proposal и oldprop, а затем запрашивает подтверждение по каждому из них. Пользователь решает удалить oldprop, а proposal оставить.

```
$ rm -i proposal oldprop
Remove proposal? n
```

```
Remove oldprop? y
$
```

Управление файлами

В операционной системе Linux используются разнообразные средства управления файлами и каталогами. Пользователь имеет возможность получить подробную информацию о файлах. Он может, например, узнать, когда они в последний раз корректировались и сколько на них имеется ссылок. Пользователь может управлять доступом к своим файлам. С каждым файлом в ОС Linux связаны права доступа, которые определяют круг лиц, имеющих к нему доступ, и вид доступа. Вы можете разрешить доступ к файлам другим пользователям либо не разрешить такового.

Файлы располагаются на физических устройствах - жестких дисках, CD-ROM, дискетах - и на каждом устройстве организуются в файловую систему. Для того чтобы получить доступ к файлам, находящимся на каком-либо устройстве, необходимо присоединить его файловую систему к определенному каталогу. Эта операция называется монтированием файловой системы. Например, для работы с файлами, расположенными на дискете, нужно сначала смонтировать ее файловую систему в определенном каталоге. В данной главе рассказывается о том, как работать с компакт-дисками, дискетами и разделами жестких дисков. Можно даже обращаться к разделу жесткого диска MS-DOS, дискете MS-DOS, а также к файловым системам, находящимся на удаленном сервере. В системе предусмотрена возможность создания резервных архивов файлов и передачи архивов по сети в другие системы. Файлы можно сжимать, что позволяет повысить эффективность передачи и обеспечивает экономию дискового пространства.

Архивирование и сжатие широко применяются при получении программных пакетов из удаленных источников. Сжатый заархивированный пакет программ переписывается на жесткий диск, а затем распаковывается и разархивируется. После этого его можно инсталлировать в систему. Именно таким образом пользователи получают большинство новых программных средств ОС Linux.

Вывод информации о файлах: команда ls -l

Команда ls -l позволяет получить подробную информацию о файле. Сначала указываются права доступа, затем количество ссылок, имя владельца файла, имя группы, к которой он относится, размер файла в байтах, дата и время последнего изменения и, наконец, имя файла. Имя группы обозначает группу, которой предоставляется доступ по категории "группа". Например (см. ниже), тип файла mydata - обычный файл. У него всего одна ссылка; это говорит о том, что у файла нет других имен. Имя владельца - chris, оно совпадает с регистрационным именем данного пользователя. Имя группы - weather. Вероятно, есть и другие пользователи, входящие в эту группу. Размер файла - 207 байт. Последний раз его корректировали 20 февраля в 11:55. Имя файла - mydata.

```
   Права доступа                                    Дата и время
      |                                              Последнего
      |                                              изменения
 Тип  | Количество    Имя        Имя    Размер       |        Имя
файла | ссылок     владельца   группы  файла         |       Файла
 |    |      |         |          |    в байтах     |         |
 |  -----    |         |          |       |    ----------     |Ц
 v |     |  v         v          v       v   |          |    v
 -rw-r--r-- 1      chris       weather   207 Feb  20  11:55 mydata
Если вы хотите получить подобную информацию обо всех файлах,  на-
ходящихся в данном каталоге, дайте команду ls -l без аргумента:
$ ls -l
-rw-r--r-- chris weather 207 Feb 20 11:55 mydata
-rw-rw-r-- chris weather 568 Feb 14 10:30 today
-rw-rw-r-- chris weather 308 Feb 17 12:40 monday
```

Часть 2. GUI — Graphic User Interface

Linux чрезвычайно гибок и поддерживает использование множества различных рабочих сред. KDE — самая популярная рабочая среда в мире Linux и занимает это место по достоинству. Она изящно выглядит, весьма удобна и проворна, солидна и обладает богатыми возможностями. Вы можете положиться на нее в деле отправки и получения электронной почты, с помощью ее браузеров можете путешествовать по Всемирной Сети, ее приложения помогут вам записывать компакт-диски и просматривать кинофильмы, работать с текстовыми документами и крупноформатными таблицами. Но при всем этом, KDE — это интерфейс, построенный на мощном основании графического движка Linux-систем, который называют X Window system, XFree86, или просто X. То, что делает рабочая среда KDE в глубине операционной системы — управление окнами, границами, фоном, изображениями, полосами прокрутки и многим другим — обеспечивается работой X.

Если вы когда-либо самостоятельно устанавливали систему Linux, то обязательно проходили этап конфигурирования графического вида рабочей среды. Однако то, что вы настраивали тогда, не было средой KDE или средой GNOME. В тот момент ваши действия касались именно X. X — это та основа, на поддержку которой опирается любая графическая программа, в том числе и рабочая среда, типа KDE или GNOME.

Таким образом, у пользователя появляется дополнительная степень свободы. Коль скоро пользовательский интерфейс представляет собой обычную служебную программу, то ее также можно выбирать исходя из особенностей выполняемой задачи. Свободная ОС не привязывает потребителя к конкретному графическому окружению. Эта особенность позволяет добиться высокой степени эргономичности, учитывая не только объективно существующие цели, но и субъективные предпочтения самого потребителя.

Пользовательские оболочки Linux принято делить на две категории: оконные менеджеры и интегрированные графические среды. Первые предоставляют потребителю только механизм управления визуальными объектами, тогда как вторые включают в себя дополнительное ПО.

Разумеется, эта классификация не является строгой — для некоторых оконных менеджеров существуют графические инструменты их настройки. Тем не менее ради такой мелочи терминологию менять никто не стал. Особой путаницы это не вносит, поскольку тип интерфейса интересует потребителя в последнюю очередь. Главное для него — практическая функциональность.

Вряд ли найдется дистрибутив, включающий в себя все существующие оконные менеджеры и интегрированные графические среды. На сегодняшний день их насчитывается около двухсот. Хотя для решения обычных пользовательских задач вполне можно ограничиться значительно меньшим количеством интерфейсов.

KDE

KDE — мощная графическая оболочка, включающая в себя набор прикладных программ, достаточный для удовлетворения потребностей обычного пользователя. Имеется даже полнофункциональный офисный пакет KOffice, по своим возможностям на уступающий более знаменитому OpenOffice.org. Эта оболочка входит в стандартную поставку дистрибутивов Kubuntu, OpenSuSE, ASPLinux, ALTLinux, MOPS, Mandriva, Debian.

KDE чрезвычайно популярен благодаря разнообразию входящего в его состав программного обеспечения. Так, набор приложений дистрибутива MOPS фактически состоит только из ПО, встроенного в KDE. И при этом получаемая система достаточно функциональна, чтобы удовлетворить потребности большинства пользователей.

Помимо офисного пакета в KDE входят браузер, почтовый клиент и даже программа для работы в сети BitTorrent. Стремление создателей этой оболочки сделать самодостаточную рабочую среду привело к тому, что фирменный центр управления

позволяет определять ряд общесистемных параметров. А конфигурации шрифтов и раскладок, используемых этим интерфейсом, имеют приоритет над задаваемыми обычным для Linux способом.

С другой стороны, приложения KDE практически не знакомы пользователям Windows, поскольку не являются кроссплатформенными. Из-за этого могут возникнуть определенные сложности при переходе между ними.

А как же OpenOffice, Firefox, Thunderbird? Неужели они не могут быть запущены из KDE? Конечно, могут, и будут работать вполне исправно. Однако пользователь может столкнуться с некоторыми тяжело решаемыми проблемами. Например, если ему потребуется уменьшить шрифт на личной панели закладок браузера Firefox, то средствами KDE этого сделать не получится — придется вручную править конфигурационный файл браузера.

Центр управления — особая гордость KDE. С его помощью можно настроить всё, что в принципе настраивается: от сетевых интерфейсов до фона рабочего стола. Его концепция особенно привлекательна для начинающего пользователя, поскольку все необходимые инструменты собраны в одной оболочке.

Однако ряд современных дистрибутивов (Mandriva, OpenSuSE, ALTLinux, Linux XP) имеют свои центры настройки, что снижает потребительскую ценность аналогичной программы, входящей в состав KDE. Фактически она дублирует основные функции штатного настройщика.

Главным недостатком KDE принято считать высокие требования к аппаратной конфигурации. Действительно, на старых и маломощных машинах этот интерфейс лучше не использовать.

GNOME

GMOME — давний конкурент KDE. Споры между сторонниками и противниками использования этих рабочих сред уже переросли в очередную Holy War. Идут они и по сей день, поэтому можно смело сказать, что решающего преимущества нет ни у одного интерфейса. GMOME используется в дистрибутивах Ubuntu, Linux XP, OpenSuSE, ASPLinux, Fedora, Mandriva, Debian.

Главное достоинство этой рабочей среды заключается в том, что она построена на библиотеке GTK. Этот инструментарий распространяется по гибкой лицензии, допускающей его использование коммерческими программами. Поэтому теоретически потенциал GNOME весьма высок. Если Linux займет солидную нишу в корпоративном секторе, то скорее всего ряд разработчиков востребованных прикладных программ не смогут принять GPL и будут ориентироваться на модели, позволяющие извлекать серьезную прибыль. Кстати, отечественный дистрибутив Linux XP, создатели которого сразу позиционировали свой продукт как коммерческий, использует именно этот рабочий стол. Надо сказать, что именно GTK применялась при создании весьма популярных программ Firefox и Thunderbird. Поэтому уже упоминавшаяся выше задача об уменьшении размера шрифтов личной панели закладок в GNOME решается средствами самой оболочки, что значительно проще.

Настраивается GNOME при помощи централизованной системы GConf. Правда, ее нельзя считать до конца отработанной и адаптированной для начинающего пользователя. Меню несколько запутанно, и не всегда нужный пункт можно найти интуитивно, документацию все-таки придется полистать.

Требования у GNOME и KDE к ресурсам машины примерно одинаковы. Компьютер, на котором работает одна оболочка, пригоден и для использования второй.

Задание для выполнения

Часть 1.

1. Определить путевое имя рабочего каталога. Как обозначается корневой каталог? Какое путевое имя получили (относительное или абсолютное)?
2. Создать в начальном каталоге два подкаталога. Просмотреть содержимое рабочего каталога. Просмотреть содержимое родительского каталога, не переходя в него.
3. Перейти в системный каталог. Просмотреть его содержимое. Просмотреть содержимое начального каталога. Вернуться в начальный каталог.
4. Удалить созданные ранее подкаталоги.
5. Получить информацию по командам ls и cd с помощью утилиты man. Изучить структуру man-документа.
6. Получить краткую информацию по командам ls и cd с помощью команды whatis и apropos. В чем различие?
7. То же, что и в п.5, только с помощью команды info.

Часть 2.

1. Изучите информацию о панелях, рабочем столе, меню используемой вами системы в Справочной системе вашего дистрибутива.
2. Проверьте, установлены ли у вас пакеты build-essential и manpages-dev при помощи утилиты Synaptic (Система > Администрирование > Менеджер пакетов Synaptic). Если не установлены — установите, они вам пригодятся.
3. Настройте под себя внешний вид системы, опишите весь процесс настройки в отчете.
4. Определите, какие приложения установлены в системе, подумайте, какие из них вам могут пригодиться для выполнения лабораторных работ, а какие — нет.

Часть 3.

1. Создайте в домашнем каталоге следующую структуру подкаталогов (существующие каталоги не удаляйте!)
   a. домашний каталог ----|
      1. |-ВашаФамилия-|
         a. |-1-|
         b. | |-2
         c. | |-3
         d. |
         e. |-4
2. Скопируйте файл /etc/group в каталог 1 используя абсолютные имена копируемого файла и каталога назначения.
3. Скопируйте файл /etc/group в каталог 2 используя абсолютное имя копируемого файла и относительное имя каталога назначения.
4. Скопируйте файл /etc/group в каталог 3 используя относительные имена копируемого файла и каталога назначения.
5. Скопируйте файл /etc/group в каталог 4 используя абсолютные имена копируемого файла и относительное имя каталога назначения с использованием специального символа ~.
6. При помощи одной команды зайдите в каталог 3.
7. Удалите файл group из каталога 4 при помощи одной команды.
8. Перейдите в свой домашний каталог. Удалите каталоги 1 и 4.
9. Выведите первые и последние 13 строк файла /etc/group.

Отчет

В этой и последующих работах отчет по проделанной работе представляется преподавателю в стандартной форме: на листах формата А4, с титульным листом (включающим тему, фио, номер зачетки и пр.), целью, ходом работы и выводами по выполненной работе. Каждое задание должно быть отражено в отчете следующим образом: 1) что надо было сделать, 2) как это сделали, 3) что получилось, и, в зависимости от задания – 4) почему получилось именно так, а не иначе.