Федеральное агентство по образованию

Московский инженерно-физический институт
(государственный университет)

В.О. Тихомиров

ВВЕДЕНИЕ В LINUX

Рекомендовано УМО "Ядерные физика и технологии"
в качестве учебного пособия
для студентов высших учебных заведений

Москва 2007

УДК 004.451.9(075)
ББК 32.973-018.2я7
Т46

Тихомиров В.О. **ВВЕДЕНИЕ В LINUX**: *учебное пособие.* М.: МИФИ, 2007. – 104 с.

Операционная система Linux в последние годы приобретает все большее распространение во всем мире. Помимо традиционного для семейства UNIX применения в различных серверных системах, Linux все чаще используется и на обычных персональных компьютерах. Данное пособие предназначено для начинающих пользователей, которым не требуются специальные знания по установке и настройке Linux, а необходимо быстро научиться пользоваться системой. Основное внимание уделено базовым командам и программам: работе с файлами и каталогами, настройке рабочего окружения пользователя, использованию текстовых редакторов, компиляции программ, работе в сети.

Предназначено для студентов различных факультетов, изучающих Linux в рамах учебной программы, а также для аспирантов и сотрудников, желающих самостоятельно освоить работу в системе Linux.

Пособие подготовлено в рамках Инновационной образовательной программы.

Рецензент канд. физ.-мат. наук, доц. А.А. Богданов

ISBN 978-5-7262-0827-5

Редактор Л.М. Бурлакова

Подписано в печать 29.10.2007. Формат 60x84 1/16
Печ.л. 6,5. Уч.-изд.л. 6,5. Тираж 200 экз.
Изд. № 4/34. Заказ №

Московский инженерно-физический институт
(государственный университет).
115409, Москва, Каширское ш., 31
Типография издательства "Тровант".
г. Троицк Московской обл.

Содержание

Предисловие

Автор этой книги на протяжении многих лет занимается обработкой экспериментальных данных и компьютерным моделированием в области физики элементарных частиц. За это время ему приходилось работать на самых разных компьютерах с различными операционными системами. В последние годы заметно выросла популярность операционной системы Linux – свободно распространяемой системы семейства UNIX. Эту операционную систему можно все чаще встретить установленной как на персональных компьютерах, так и на серверах, входящих в состав крупных вычислительных центров в России и за рубежом. Linux соединяет в себе мощь многопользовательской и многозадачной операционной системы, высокую надежность в работе, гибкость в настройке и относительно невысокую требовательность к ресурсам компьютера. Кроме того, Linux и подавляющее большинство программ для этой системы распространяются бесплатно.

На основании собственного опыта работы в Linux, а также опыта обучения других пользователей, автором был создан лекционно-практический курс, который преподается в Московском инженерно-физическом институте. Материалы этого курса, в свою очередь, послужили основой для написания данной книги. Она предназначена для начинающих ***пользователей*** операционной системы Linux. Здесь не рассматриваются вопросы, относящиеся к инсталляции Linux, к глобальной настройке и администрированию системы. Наша задача – дать обычному пользователю начальные знания для практической работы в Linux. Предполагается, однако, что пользователь достаточно хорошо знаком с компьютером в целом и ему известны такие понятия, как файл, каталог, исполняемая программа и т.п. Подразумевается также, что читатель, получив начальные знания из этой книги, сможет при необходимости легко их расширить, используя справочную систему Linux, соответствующую печатную литературу, а также Интернет. Все сказанное избавляет автора от необходимости освещать многие вопросы, которые обычно занимают значительный объем в толстых руководствах по Linux и дает возможность сосредоточиться на объяснении основных моментов. Итак, цель данной работы отражена в ее названии: это введение в Linux.

Автор будет благодарен за любые замечания и предложения, касающиеся данной книги, которые можно направлять по электронной почте по адресу: `tikhomir@sci.lebedev.ru`.

Сделаем еще несколько вводных замечаний.

Современные версии Linux имеют достаточно развитые графические оболочки (например, KDE или GNOME), позволяющие выполнять многие действия с помощью системы графических панелей, меню, иконок и т.п., аналогично операционной системе Windows. Однако наиболее полно использовать возможности Linux удается с помощью команд, вводимых в командной строке терминала. Командный режим позволяет также лучше разобраться в механизме и логике работы системы. Наконец, графические оболочки типа KDE или GNOME могут быть просто не установлены на вашем компьютере. Поэтому именно работе в режиме командной строки Linux будет посвящена бóльшая часть этой книги.

Работа в Linux возможна в текстовом режиме терминала и в графическом режиме. Везде далее предполагается графический режим, хотя большинство команд, конечно, будут выполняться в этих режимах одинаково. Графическая система в Linux основана на X Window System (или X11). Поэтому далее в тексте будут встречаться обозначения типа "X окно", означающее любое графическое окно в X11, или "X терминал", что означает X окно терминала, в котором вводятся команды.

Наконец, необходимо иметь в виду, что существует множество версий Linux. Версии могут отличаться как ядром (kernel) системы, так и набором программ, которые включены в дистрибутив. Возможна также различная настройка системы. Поэтому необязательно все те команды, файлы и программы, которые упоминаются в этой книге, будут в точности соответствовать тому, что есть на вашем компьютере. Хотя с учетом того, что мы рассматриваем лишь основные команды и программы, различия должны быть минимальными. Материалы данной книги основаны на дистрибутиве Scientific Linux 3, который, в свою очередь, основан на Red Hat Entreprise Linux 3, версия ядра 2.4.

В данном пособии используются следующие способы выделения шрифтом. *Наклонным шрифтом* будут обозначаться команды, которые вводит пользователь с клавиатуры. То, что в ответ выводит на экран операционная система, выделяется `таким шрифтом`. Таким же шрифтом будут обозначаться имена файлов и каталогов, встречающиеся в тексте. **Жирным шрифтом** выделяются имена программ и команд в тексте. ***Жирным наклонным шрифтом*** выделяется текст, на который необходимо обратить внимание. Ключевые слова, такие как `command`, `option`, `parameter`, `filename`, `dirname`, и т.д.

означают, что на их место необходимо поставить конкретную команду, опцию, параметр, имя файла или каталога. Необязательные параметры команд будут заключаться в квадратные скобки: [ ]. Троеточие ... означает возможное повторение однотипных параметров или имен файлов и каталогов в командах. Отдельная клавиша клавиатуры будет обозначаться так: <a>. А такое обозначение: <Ctrl-a> указывает на необходимость нажать клавишу <Ctrl> и, не отпуская ее, клавишу <a>.

1. Начальные понятия

1.1. Синтаксис команд

Общий формат командной строки в Linux выглядит следующим образом:

command [*option...*] [*parameter...*]

Обычно опция представляет собой символ "–", за которым следует цифра или буква латинского алфавита. В этом случае несколько опций можно записывать как несколько букв подряд. В качестве параметров часто выступают имена файлов или каталогов.

Например, команда **ls**, введенная без опций и параметров, выведет на экран список файлов в текущем каталоге в кратком формате. Если ввести

ls –l

то формат вывода изменится на подробный. Команда

ls –lt

выдаст листинг в подробном формате, в котором файлы отсортированы по времени их создания или последней модификации, а

ls –lt dirname

выведет на экран листинг каталога `dirname`. Команда **ls –t –l** работает аналогично команде **ls –lt**, т.е. опции можно вводить и по отдельности, а их порядок несущественен.

В некоторых командах используются и многобуквенные опции. Например, команда

xterm –bg black –fg white &

откроет новое окно X терминала с текстом белого цвета на черном фоне.

Ряд команд имеет следующий формат:

command –*option*... [*parameter*...]

Здесь опция представляет собой не один символ, а целое слово, перед которым стоят два знака "–" подряд. Например, если ввести

ls –*help*

то можно получить краткую справку по использованию команды **ls**. Наконец, некоторые команды допускают смешанный синтаксис. Команда

rm –f filename

удалит из текущего каталога файл с именем `filename`, не запрашивая у пользователя подтверждения. То же действие будет выполнено, если набрать

rm –*force filename*

В одной командной строке можно указать сразу несколько команд, разделив их знаком ";". Длинную команду можно продолжить на новой строке, если в предыдущей в качестве последнего символа ввести "\" и нажать <`Enter`>. Можно сгруппировать несколько команд в одну, заключив их в круглые скобки (см. пример в разделе 4.4).

Иногда при работе какой-либо команды или программы на экран выводится такое количество строк текста, которое не умещается в окне терминала. Если возникает необходимость приостановить вывод текста на экран, нажмите клавиши <`Ctrl-s`>. Продолжить выполнение программы и вывод на экран можно, нажав <`Ctrl-q`>. Чтобы совсем прервать выполнение программы, нажмите <`Ctrl-c`>.

Все команды, опции, параметры, а также имена файлов и каталогов в Linux являются регистро-зависимыми, т.е. строчные и заглавные буквы различаются. Сокращение команд, опций или параметров при вводе не допускается.

1.2. Редактирование командной строки

Linux предоставляет широкие возможности для работы с командной строкой. Введенные пользователем в данном окне терминала команды запоминаются в специальном буфере. С помощью клавиш <↑> и <↓> возможна навигация по списку введенных ранее команд. Командную строку можно редактировать, перемещаясь по ней с помощью клавиш <←> и <→>. Клавиша <backspace> удаляет символ, предшествующий курсору, а <`Ctrl-d`> удалит символ непосредственно над

курсором. Передвинуть курсор в начало командной строки можно, нажав <Ctrl-a>, а в конец строки – <Ctrl-e>. Чтобы удалить часть строки от позиции курсора до конца строки, нажмите <Ctrl-k>. Удалить полностью всю строку можно, нажав <Ctrl-u>, а <Ctrl-y> восстановит удаленную строку. Заметим, что большинство приведенных здесь комбинаций клавиш работает не только при редактировании командной строки в окне терминала, но и в текстовых редакторах, при редактировании строк в полях Web браузера типа Mozilla и т.п.

Большое удобство представляет собой возможность автоматического дополнения команд и имен файлов при их вводе с клавиатуры. Так, если ввести в командной строке:

chm

и нажать на клавиатуре клавишу <Tab>, то система дополнит введенные символы до полного имени команды: **chmod**, поскольку набранное буквосочетание chm является среди всех команд уникальным. Если ввести буквосочетание, которое не является уникальным, например, *ch* и нажать <Tab>, то система выведет на экран все возможные варианты продолжения команды: **chmod, chown, chgrp** и др[1].

Возможность автоматического дополнения очень полезна и при вводе имен каталогов и файлов. Например, мы хотим вывести на экран с помощью команды **more** текст файла, путь к которому есть /afs/cern.ch/atlas/user/r/rd6/public/trt02/dsts/README .
Тогда, если набрать в командной строке

more /afs/cern.ch/atl

и нажать клавишу <Tab>, то система сама дополнит путь до /afs/cern.ch/atlas/, поскольку в каталоге /afs/cern.ch/ существует только один файл (каталог), начинающийся с буквосочетания atl – это каталог atlas. Набирая имя дальше и дойдя, допустим, до имени каталога trt02, мы опять можем ускорить набор: набрать только *tr* и нажать <Tab> – система автоматически дополнит имя каталога до trt02, если буквосочетание tr является в данном контексте уникальным. Если же нет – на экран будут выведены имена всех файлов и каталогов, начинающихся с буквосочетания tr. Этим можно воспользоваться, если, например, набрав

[1] Если у вас такая возможность подсказки не реализована, исполните команду ***set autolist*** (семейство C shells) или ***set -9*** (семейство Bourne shells), или вставьте эти команды в свой startup file – сценарий автозагрузки. О shells – командных оболочках и сценариях автозагрузки см. разделы 4.1 и 4.4).

more /afs/cern.ch/atlas/user/r/rd6/public/

вы забыли имя следующего каталога в пути. Нажав <Tab>, вы получите подсказку в виде всех имен файлов и каталогов, содержащихся в каталоге `public`. Отметим также, что такой способ ввода имен каталогов и файлов страхует от возможной опечатки при наборе.

Как уже говорилось, команды, введенные пользователем в командной строке, сохраняются системой в специальном буфере. Вывести на экран содержимое этого буфера можно с помощью команды

history [*n*]

где необязательный параметр `n` – число последних команд в буфере. Команда **history** может быть полезной, если вы хотите вспомнить какую-то сложную и длинную команду, которую вводили уже достаточно давно. Например, команда **history** вывела на экран:

```
565  15:02   paw
566  15:04   ls -l *.f*
567  15:04   ls -al ~/bin/
568  15:04   ls -al ~/hinput/
569  15:05   ln -s /home/tikhomir/hinput/fextr.f77
570  15:06   paw
571  15:07   ls -alt | head
572  15:07   more tomo.kumac
573  15:09   date ; echo "exec tomo 100000" | atlsim -w 0 ; date
574  16:38   ls -alt | head
575  16:39   mv tomo.hbook tomo3x8.33mm.hbook
```

Чтобы просто повторно выполнить одну из команд, наберите, например,

!573

где 573 – это номер команды в списке, выданного командой **history**. Если же вы хотите отредактировать эту команду, то можно сначала скопировать ее в командную строку через буфер обмена, как об этом рассказано ниже. Часто команда **history** используется вместе с фильтром **grep** – в случае, когда желательно вывести на экран не весь список команд, а только некоторые из них, удовлетворяющие определенному критерию отбора (см. пример в разделе 2.8).

Отметим еще одну возможность, которая применяется при редактировании командной строки. В Linux очень удобно реализован буфер обмена (clipboard). Подведите курсор мыши к началу выделяемого фрагмента текста в окне терминала. Нажмите левую кнопку мыши

и, не отпуская ее, сдвиньте курсор к концу выделяемого фрагмента. Отпустите левую кнопку – и выделенный текст будет скопирован в буфер обмена. Если теперь нажать среднюю кнопку мыши (или правую и левую одновременно, если средней кнопки нет), то содержимое буфера обмена будет скопировано в командную строку терминала. Таким образом в командную строку можно быстро и без ошибок перенести длинные имена файлов или команды с множеством параметров, выведенные, например, на экран командами **ls** или **history**.

1.3. Справочная система Linux

Описания большинства команд, системных функций, специальных файлов Linux хранятся в виде так называемых manual pages (справочные страницы). Справку по использованию команды `command` можно получить, набрав

man [*part*] *command*

Здесь необязательная опция `part` указывает на раздел справочных страниц (цифра от 1 до 9). Справочные страницы делятся на разделы по основному назначению команд: в первый раздел отнесены обычные команды, во второй – системные функции и т.д. Если опция `part` опущена, будет вызвана справочная страница из раздела с наименьшим номером.

Выполнив команду **man**, пользователь попадает в режим просмотра справочной страницы соответствующей команды `command`. Основа навигации по странице такова: нажав клавишу <`space`> на клавиатуре, продвигаемся по справочной странице вперед, нажав клавишу <`b`> – назад. Если справка велика по объему, могут оказаться полезными функции поиска: */pattern* ищет первое появление буквосочетания `pattern` в тексте страницы по направлению вперед, а *?pattern* – по направлению назад. (Последняя возможность будет реализована, если переменная окружения `PAGER` имеет значение `less`[2]). Чтобы выйти из режима просмотра справочной страницы, нажмите клавишу <`q`>.

Команда **man** используется и для поиска команд по ключевому слову. Например, необходимо найти в системе команду, способную конвертировать графический файл из pbm (Portable Bitmap) в PostScript

[2] В некоторых примерах, приведенных в данном разделе, встречаются такие понятия, как переменная окружения (environment), командная оболочка (shell), а также знак перенаправления выходного потока команды “>” и знак конвейера (pipe) “|”. Подробнее значение этих понятий и символов будет разъяснено позже.

формат. Тогда поиск соответствующей команды можно задать, набрав в командной строке:

man –k pbm

или:

man –k postscript

В первом случае на экран будет выведен список всех команд в системе, в описании которых присутствует ключевое слово `pbm`, а во втором случае – ключевое слово `postscript` (здесь мы имеем дело с редким в мире Linux исключением, когда регистр букв в ключевом слове несущественен). Вместе с именем команд будет дано их краткое описание. Используя фильтр **grep** (раздел 2.8), можно задать более сложный алгоритм поиска. Например, командой

man –k pbm | grep –i postscript

задается поиск всех команд, в которых в качестве ключевых слов присутствуют как `pbm`, так и `postscript`.

Вместо **man –k** можно воспользоваться эквивалентной командой **apropos** без опций. Отметим также, что возможность поиска команд по ключевому слову существует только в том случае, если в системе была создана соответствующая база данных. Если это не так, попросите вашего системного администратора сделать это.

Если вы хотите быстро узнать, содержит ли справочная страница команды `command` некое слово (буквосочетание) `pattern`, наберите

man command | grep –i pattern

При этом `pattern` может и не быть ключевым словом из базы данных. Если буквосочетание `pattern` просто встречается в тексте справочной страницы, то на экран будут выведены все строки, содержащие этот образец.

Команда **xman** представляет собой графический интерфейс к **man**. Она удобна тем, что представляет отсортированный по разделам полный список всех команд в системе, для которых существуют справочные страницы.

Для многих команд можно быстро получить краткую справку по их использованию, набрав в командной строке:

command –h

или:

command –-help

Это бывает полезно, например, когда вы знаете, что именно делает данная команда, но забыли, как задается какая-либо ее опция или параметр.

Другим источником помощи по использованию команд и программ Linux является команда **info**. Введенная без параметров, она открывает навигацию по доступным info pages (информационным страницам) различных команд и программ. Выделив клавишами <↓>, <↑> или <Tab> интересующую нас команду и нажав <Enter>, мы войдем в справочную систему info для данной команды. Того же эффекта можно добиться, если сразу ввести в командной строке:

info command

Система организации info-страниц является гипертекстовой. Клавиша <n> переместит нас на следующую страницу документа, клавиша <p> вернет на предыдущую, а <u> – передвинет на один уровень вверх в документации. Выйти из системы info можно, нажав <q>.

Содержимое info-страниц для некоторых команд полностью повторяет содержимое справочных страниц, но иногда информационные страницы дают более подробную и более точную информацию о команде. К тому же система info лучше структурирована. К ней удобнее обращаться, когда вас интересует не отдельная команда Linux, а какой-то более широкий круг вопросов, например – набор команд для работы с файлами.

Если вы работаете в графической оболочке KDE, то можете использовать для просмотра справочных и информационных страниц программу Konqueror – браузер и файловый менеджер KDE. Для этого просто введите `man:command` или `info:command` в адресной строке браузера, где `command` – интересующая вас команда.

Отметим, что для некоторых команд и программ, установленных на компьютере, справочные и/или информационные страницы могут отсутствовать. Заметим также, что часть команд относится к так называемым внутренним (internal) командам командной оболочки (shell), в которой они выполняются. Например, к внутренним относится команда **cd**, с помощью которой можно сменить текущий каталог. Информация о внутренних командах содержится в справочной странице соответствующей оболочки. Таким образом, если вы работаете, скажем, в оболочке `bash`, то справку по использованию команды **cd** можно получить, набрав

man bash

С другой стороны, справочная информация может касаться не только команд, но и системных функций (второй раздел справочных страниц) или некоторых файлов (пятый раздел). Например, команда

man fstab

выдаст информацию о назначении и формате файла `/etc/fstab`. Обращайте также внимание на разделы `FILES` и `SEE ALSO`, которые расположены в конце справочной страницы команды. Там указаны файлы, имеющие отношение к данной команде (если такие существуют), а также другие команды в системе на схожую тему.

Многие справочные страницы весьма велики по своему объему. В таком случае бывает удобно получить соответствующий текст в виде файла для того, чтобы потом просматривать его в каком-либо редакторе или чтобы распечатать текст на принтере. Следующий пример показывает, как сохранить справочную страницу программы-оболочки **bash** в виде текстового файла `bash.txt`:

man bash | col –b > bash.txt

Здесь часть команды **col –b** убирает некоторые элементы форматирования из справочной страницы.

Программы графических оболочек GNOME и KDE не включены в manual pages, а имеют собственную справочную систему.

Если вы пользуетесь дистрибутивом Linux от фирмы RedHat, то основой для установленных на вашем компьютере программ послужили так называемые RPM пакеты. Каждый такой пакет содержит как саму устанавливаемую программу (или набор программ), так и документацию к ней в различных форматах. Можно получить список всех файлов документации, касающихся данной программы и других, относящимся вместе с ней к одному RPM пакету:

rpm –qdf command

Отметим, что здесь в качестве `command` должен фигурировать ***абсолютный путь*** к команде или программе. Например, чтобы получить список файлов документации того RPM пакета, к которому относится программа проверки орфографии **aspell**, наберите

rpm –qdf /usr/bin/aspell

Если вы не знаете, в каком именно каталоге расположена данная программа, воспользуйтесь конструкцией **which** (см. раздел 2.12):

rpm –qdf ‘which aspell‘

Обратите здесь внимание на обратные кавычки.

Многие пакеты программ, а также различные комплексные вопросы освещены в документации, представленной в текстовом или HTML форматах. Среди подобной документации отметим так называемые HOWTO страницы. Например, Emacs HOWTO описывает работу редактора **emacs**, Security HOWTO освещает вопросы, связанные с компьютерной безопасностью и т.д. HOWTO файлы удобнее всего просматривать с помощью браузера типа Mozilla. Файлы с документацией в текстовом или HTML форматах обычно находятся в каталогах `/usr/doc`, `/usr/share/doc`, `/usr/doc/HOWTO` или `/usr/doc/HTML`.

Кроме этого, на вашем компьютере могут быть установлены и другие пакеты с документацией по Linux. Например, "Red Hat Linux Installation Guide", "Red Hat Linux Getting Started Guide", "Red Hat Linux Reference Guide", поставляемые фирмой RedHat вместе с их дистрибутивом Linux, а также "Linux Installation and Getting Started Guide", "Linux System Administrators' Guide" и "The Network Administrators' Guide" от Linux Documentation Project. Обычное место для такой документации – в подкаталогах каталога `/usr/doc` или `/usr/share/doc`.

В последние годы в продаже можно найти довольно много книг по Linux, как переводных, так и российских авторов. Наконец, еще одним важным источником информации для пользователя Linux служит Интернет. Несколько ссылок на литературу и Интернет ресурсы, касающихся операционной системы Linux, можно найти в конце этой книги.

2. Работа с файлами

2.1. Файловая система

Файловая система в Linux имеет иерархическую структуру: файлы вложены в каталоги (то же, что и папка в Windows), которые могут быть вложены в другие каталоги. Однако имена дисков в явном виде отсутствуют, а существует только один корневой каталог, обозначаемый символом "/". В корневом каталоге расположены все другие каталоги. Например, в каталогах `/bin` и `/usr` содержится большинство программ и библиотек системы. В каталоге `/etc` расположены файлы и каталоги, связанные с настройками различных программ. Каталог `/tmp` служит для хранения временных файлов, а в каталоге `/home` содержит в себе домашние каталоги пользователей.

Часть полного имени файла, включающая в себя только имена каталогов, называется path (путь к файлу). Имена каталогов в пути также разделяются символом "/". Таким образом, полное имя файла выглядит так:
`/dirname1/dirname2/.../filename`
где `dirname1, dirname2 ...` – имена каталогов, а `filename` – собственно имя файла. Имена каталогов и файлов могут состоять как из символов латинского алфавита, так и из специальных символов типа ".", ",", "_", "#". Нельзя использовать в именах символы "/", "*", "?", простые и двойные кавычки. Точка обычно используется для разделения имени файла и его расширения, указывающего на тип файла. Напомним, что регистр букв в именах является значимым.

Путь, начинающийся с символа "/", является абсолютным в файловой системе. Если символ "/" в начале пути отсутствует, то подразумевается путь относительно текущего каталога. Таким образом, скажем, имя `/etc/README.txt` относится к файлу `README.txt`, расположенному в каталоге `etc`, который, в свою очередь, находится в корневом каталоге файловой системы. Имя `etc/README.txt` относится к файлу `README.txt` в каталоге `etc`, расположенном в текущем каталоге, а `README.txt` – к файлу в текущем каталоге.

Имя каталога, состоящее из знака ".", означает текущий каталог, ".." – каталог на один уровень выше текущего, а "∼" – корневой каталог пользователя.

В именах файлов и каталогов часто используют шаблон "*", означающий "все, что угодно". Таким образом, `*.f` означает: все файлы (в текущем каталоге) с расширением `.f`; `*dat*` – все файлы и каталоги, в имени которых встречается сочетание символов `dat`; `../*` – все файлы и каталоги в каталоге, расположенном на один уровень выше текущего; `∼/*.f` – все файлы с расширением `.f`, которые расположены в корневом каталоге пользователя.

Другой шаблон – "?" – в именах файлов и каталогов используется в значении "один любой символ в этом месте". Например, обозначение `*.d?t` при обращении к имени файла будет означать все файлы с расширением `.dat`, `.dot`, `.d2t`, но не `.dt` или `.dart`.

Имя текущего каталога можно вывести на экран командой **pwd**.

2.2. Список файлов

Как уже говорилось выше, команда **ls** служит для вывода на экран списка файлов и каталогов.

ls –l

делает список подробным: помимо собственно имен файлов будет выведена информация о его размере, принадлежности тому или иному пользователю, времени последней модификации, правах доступа к файлу.

ls –a

показывает также файлы и каталоги, чьи имена начинаются с символа "." (это так называемые скрытые файлы, в которых обычно содержатся параметры настройки различных программ). Обратите внимание, что в списке будут присутствовать два каталога с именами "." и ".." – они обозначают текущий и родительский каталоги соответственно.

ls –t

сортирует файлы в списке по времени их последней модификации (по умолчанию сортировка проводится в алфавитном порядке).

ls –S

сортирует файлы по их размеру.

ls –r

меняет порядок сортировки на обратный.

ls –R

делает список рекурсивным, т.е. будет также выведен список файлов во всех вложенных каталогах, если такие существуют.

Введенная без параметров, команда **ls** покажет файлы, расположенные в текущем каталоге. В качестве параметра могут служить имена файлов или каталогов, а также шаблоны, содержащие символы "*" и "?". Так,

ls –l /

выведет на экран список каталогов и файлов, расположенных в корневом каталоге файловой системы.

ls –l /data

выдаст список файлов из каталога `/data`, расположенного в корневом каталоге файловой системы Linux.

ls –l data

выведет информацию о файле `data`, находящемся в текущем каталоге или о файлах, находящихся внутри `data`, если это – каталог.

ls –R ~

выдаст список всех файлов, расположенных в корневом каталоге данного пользователя и во всех дочерних каталогах.

*ls –Sl *.data*

покажет в порядке убывания размера все файлы текущего каталога, имеющего расширение `.data`.

*ls –tl ../*data**

выведет отсортированный по времени последней модификации список всех файлов, в именах которых встречается буквосочетание `data` и расположенных в родительском по отношению к текущему каталоге.

Полезным может оказаться использование опции **ls ––color**, которая выдаст на экран листинг, выделяя в нем файлы разных типов разными цветами. Пример использования данной опции приведен в разделе 4.4.

Отметим, что вообще говоря, с помощью команды **ls** каждый пользователь безусловно может получить информацию только о файлах, принадлежащих лично ему. Информация о файлах других пользователей и о системных файлах может быть закрыта. Если пользователь попытается обратиться к таким закрытым для него файлам с помощью команды **ls**, он получит сообщение: `Permission denied` (доступ запрещен). Подробнее о правах доступа к файлам говорится в разделе 2.3.

Команду **ls** приходится применять очень часто, поэтому рекомендуется завести себе короткие псевдонимы (aliases, см. раздел 4.3) для наиболее часто употребляемых вариантов команды. Несколько примеров такого рода приведены в разделе 4.4.

2.3. Права доступа к файлам

Рассмотрим более подробно пример листинга, выданного командой **ls** с опцией –l. Здесь часть верхней строки до символа "$" включительно представляет собой prompt – приглашение, выдаваемое системой, когда она ожидает ввода команд.

[tikhomir@lxplus050] ~/public/strtmc $ *ls –lt*

```
-rw-r--r--   1 tikhomir zp     247 Oct  9  2006 last.kumac
lrwxr-xr-x   1 tikhomir zp      23 Oct  9  2006 sdst.001 -> ../trt0
```

```
6/sim02/sdst.444
-rw-r--r--   1 tikhomir zp    4860 Oct  6  2006 zsctrec.age
-rw-r--r--   1 tikhomir zp    4785 Oct  6  2006 zsctrec.age~
-rw-r--r--   1 tikhomir zp  139411 Oct  5  2006 mctrdali.f
lrwxr-xr-x   1 tikhomir zp      24 Oct  5  2006 resid -> ../trt06/p
roc/newf/resid
lrwxr-xr-x   1 tikhomir zp      32 Oct  5  2006 alistr2.kumac -> ..
/trt99/proc/newf/alistr2.kumac
lrwxr-xr-x   1 tikhomir zp      12 Oct  1  2006 sdst.020 -> dst/sds
t.020
drwxr-xr-x   2 tikhomir zp    2048 Oct  1  2006 dst
-rw-r--r--   1 tikhomir zp   47434 Dec 25  2005 gfiles.tar.gz
-rw-r--r--   1 tikhomir zp     590 Nov 24  2005 outres.dat
```

В подробном листинге для каждого файла или каталога указаны (справа налево): имя файла или каталога, время создания или последней модификации, размер файла в байтах, group-имя (группа, к которой относится пользователь, здесь: `zp`), user-имя (имя пользователя – владельца файла, здесь: `tikhomir`), а также число ссылок на имя данного каталога в файловой системе.

Особо обратим внимание на первые десять символов в строке. Первый слева из них указывает на тип файла: символ "–" означает обычный файл, символ "`d`" – каталог (directory), символ "`l`" – ссылка (link, раздел 2.6). Существуют и другие специальные типы файлов и каталогов.

Последующие девять символов относятся к одному из важнейших атрибутов файла – правам доступа. Право доступа означают право для определенных пользователей на совершение определенных действий с этим файлом. Права назначаются для трех категорий пользователей: 1) user – владелец файла; 2) group – пользователь, относящийся к той же группе, что и владелец; 3) other – все остальные пользователи. Первая тройка символов в листинге относится к user, вторая – к group, третья – к other. Буквы "`r`", "`w`" и "`x`" означают соответственно: право на чтение (read) файла, право на запись (write) или изменение файла, и право на исполнение (eXecute) файла. Символ "–" означает отсутствие соответствующего права. Таким образом, в приведенном выше примере для всех обычных файлов (не ссылки и не каталоги) их владельцу разрешен доступ на чтение и изменение, а всем остальным – только на чтение.

Для каталогов права read, write и execute имеют несколько другой смысл, чем для файлов. Read означает только возможность с помощью команды **ls** увидеть имена файлов в каталоге, execute – возможность просматривать или копировать файлы из него, write – возможность добавлять или удалять файлы из каталога.

Обычно при работе с файлами и каталогами – создании, копировании и т.п. – система автоматически присваивает файлам разумные права доступа: владелец может делать со своими файлами все, что угодно, пользователь из одной с владельцем группы – читать и исполнять, а все остальные – либо только читать, либо лишены всех прав доступа. Такой подход защищает как файлы обычных пользователей, так и системные файлы от несанкционированного доступа. Однако иногда пользователю необходимо вносить коррективы в правила, принятые в системе по умолчанию. Например, если вы хотите полностью защитить какие-то свои файлы или каталоги от посторонних глаз, необходимо убрать атрибут read для всех категорий пользователей, кроме владельца. Или, напротив, необходимо создать в системе некий каталог для хранения больших файлов с какими-то данными, и разрешить доступ к нему всем пользователям. Необходимость изменить права доступа может возникнуть и при переносе файлов через сеть, дискету, флэш-память или компакт-диск.

Владелец файла `filename` может изменить права доступа к нему с помощью команды **chmod**:

chmod mode filename

Ключ `mode` состоит из трех частей. Первая часть представляет собой одну или несколько букв “u”, “g”, “o”, “a”, которые соответствуют категориям пользователей: **u**ser, **g**roup, **o**ther и **a**ll. Затем идет один из знаков “+”, “–” или “=”, которые означают: “добавить”, “убрать” и “назначить” определенные права. Затем указываются символы “r”, “w” и “x”, означающие соответствующие права доступа. Так,

chmod u–w filename

лишает владельца файла `filename` права на изменение этого файла (например, для защиты от случайного удаления);

chmod go–rwx filename

лишает всех прав доступа к файлу `filename` как group, так и other категории пользователей, т.е. всех пользователей, кроме владельца;

chmod a=xr dirname

делает каталог `dirname` для всех пользователей доступным для чтения (но не для записи).

Используя другую форму команды **chmod**, можно задать сразу все права доступа к файлу для всех категорий пользователей:

chmod LMN filename

где `L`, `M` и `N` – три цифры в интервале от 0 до 7, представляющие права доступа для user (`L`), group (`M`) и other (`N`). Три бита этих цифр (от младшего к старшему) соответствуют `x`, `w` и `r` атрибутам доступа, т.е., скажем L=1 соответствует наличию `x` атрибута для владельца файла, L=2 – `w` атрибута, L=3 – `xw`, L=4 – `r` и т.д. Таким образом, команда

chmod 754 filename

установит следующие права доступа к файлу `filename`: `xwr` для user, `xr` для group и `r` для other.

Если пользователь попытается совершить с файлом операцию, на которую он не имеет соответствующих прав, система выдаст сообщение: `Permission denied`.

2.4. Навигация по файловой системе

Напомним, что команда **pwd** выводит на экран имя текущего каталога. Команда **cd** используется для того, чтобы сменить текущий каталог:

cd [*dirname*]

сделает `dirname` текущим каталогом. Команда **cd**, выполненная без параметров, сделает текущим корневой каталог пользователя.

cd ..

передвинет текущий каталог на один уровень вверх, а

cd –

позволит вернуться в тот каталог, который являлся текущим ранее, до перехода в текущий каталог.

Имя текущего каталога (или часть этого имени) удобно включить в текст prompt – приглашения командной строки, которое появляется в окне терминала, когда система готова к вводу команд. См., например, верхнюю строку в примере листинга в предыдущем разделе. Здесь часть приглашения `~public/strtmc` представляет собой имя текущего каталога. Каждый раз, когда пользователь меняет текущий каталог,

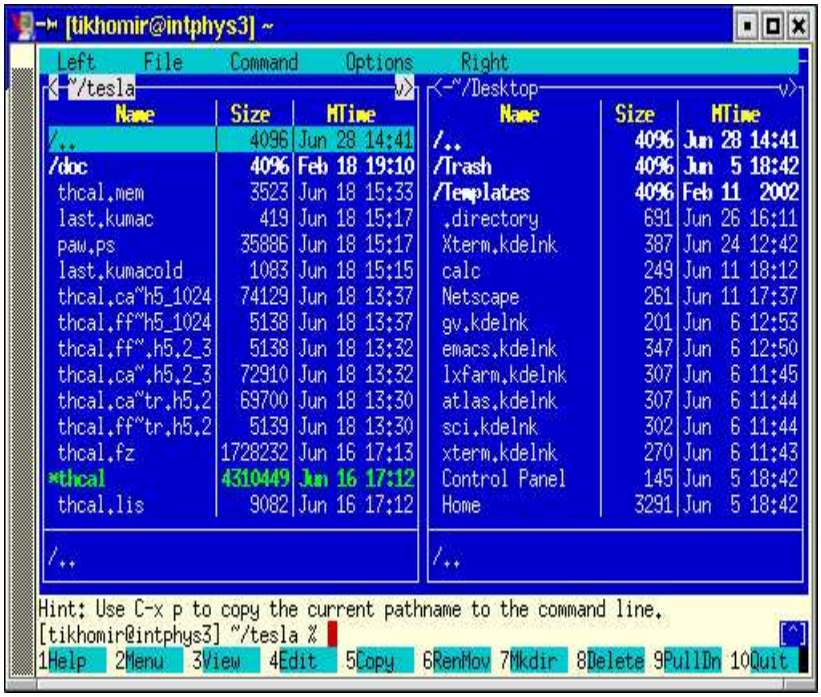

Рис. 1. *Панель программы* **mc** *(Midnight Commander)*

будет автоматически меняться и вид приглашения. О том, как включить имя каталога в строку приглашения, будет рассказано ниже, в разделе о переменных окружения.

Навигация по файловой системе возможна и с помощью различных специальных программ – файл-менеджеров. Например, если на вашем компьютере установлен Midnight Commander (команда **mc**), то его удобно использовать для навигации по файловой системе и работы с файлами. Как видно из рис. 1, панель **mc** очень напоминает широко известный Norton Commander. Создатели **mc** сохранили также назначение функциональных клавиш: <F5> – копирование, <F4> – редактирование и т.д. Программа **mc** обладает сетевыми возможностями: в разделе 8.2 будет рассказано, как ими пользоваться.

В графических оболочках KDE и GNOME существуют собственные программы для навигации по файловой системе – Konqueror и

Nautilus соответственно – похожие на Проводник в операционных системах Windows.

2.5. Создание, копирование, удаление файлов и каталогов

Простейший способ создать новый файл – это воспользоваться командой **touch**:

touch filename

создаст пустой файл в текущем каталоге. Разумеется, обычно новые файлы создаются другими способами: с помощью редакторов или различных программ. Команда **touch** приведена здесь для того, чтобы вы могли уже сейчас создать несколько новых файлов и изучить на них действие приведенных здесь команд. Чтобы удалить файл, наберите:

rm filename

Создать новый каталог можно так:

mkdir dirname

А удалить – так:

rmdir dirname

Таким образом можно удалять только пустой каталог, в котором нет ни одного файла. Чтобы удалить непустой каталог, а также все каталоги, вложенные в него, выполните:

rm –r dirname

Для того, чтобы скопировать существующий файл `oldfile` в новый `newfile`, наберите:

cp oldfile newfile

Так можно рекурсивно скопировать содержимое существующего каталога `oldadir` в новый `newdir`:

cp –r olddir newdir

Переименовать или переместить в другое место файловой системы файл или каталог:

mv oldfile newfile

При копировании или переименовании файлов особое значение имеет символ ".", применяемый в качестве имени нового файла или каталога. Этот символ означает, что новый файл должен иметь то же имя,

что и старый. Так, команда

cp /home/tikhomir/scitrd/.dat .*

скопирует все файлы с расширением `.dat` из каталога `/home/tikhomir/scitrd` в текущий каталог, не изменяя имен файлов.

Команды **cp** и **mv** могут иметь более двух параметров, если последним из них является имя каталога. Например,

*cp geant.ffr *.dat /home/tikhomir/scitrd*

скопирует файл `geant.ffr`, а также все файлы с расширением `.dat` из текущего каталога в каталог `/home/tikhomir/scitrd`.

Заметим, что в Linux крайне сложно восстановить даже только что удаленный файл. Поэтому командами **rm, mv** и **cp** следует пользоваться с осторожностью. Можно рекомендовать переобозначить эти команды (см. раздел 4.3 о псевдонимах) так, чтобы случайно не удалить нужный файл. Так, команда

rm –i filename

будет требовать от пользователя подтверждение на удаление, также как и команды **mv –i** и **cp –i** в том случае, когда создаваемый `newfile` уже существует. С другой стороны, если вы переобозначили эти команды, а вам необходимо удалить, скажем, все файлы с расширением `.dat` в текущем каталоге, то давать подтверждение на удаление каждого файла может оказаться довольно утомительной процедурой. В этом случае можно форсировать операцию удаления: команда

*rm –f *.dat*

удалит все файлы с расширением `.dat`, не спрашивая подтверждения. Еще одна команда, которой следует пользоваться с крайней осторожностью:

rm –fr dirname

удалит, не спрашивая подтверждения, все файлы и каталоги, расположенные в каталоге `dirname`, а также сам этот каталог.

Напомним, что все операции, связанные с созданием, копированием, переименованием и удалением файлов и каталогов, требуют наличия у пользователя соответствующих прав доступа.

2.6. Ссылки

Одним из видов файла является ссылка (link). Ссылки бывают двух типов: hardlink и symbolic link. Hardlink (жесткая ссылка) по существу

является копией файла с другим именем. При изменении самого файла автоматически изменяется и содержимое hardlink. Жесткие ссылки используются довольно редко и далее рассматриваться не будут.

Symbolic link (символическая ссылка, или далее – просто ссылка) представляет собой поименованный указатель на существующий файл или каталог. В некотором смысле такая ссылка схожа с ярлыком в Windows. Символическая ссылка создается командой

ln –s target [*linkname*]

где `target` – имя существующего файла или каталога, а `linkname` – имя создаваемой ссылки. Если `linkname` опустить, то имя ссылки будет совпадать с именем файла или каталога. Таким образом, команда

ln –s ../scitrd.ffr

создаст в текущем каталоге ссылку с именем `scitrd.ffr`, которая будет указывать на файл с таким же именем, расположенный в родительском каталоге. А команда

ln –s /afs/cern.ch/atlas/user/r/rd6/public/trt02/dsts/ mydsts

создаст в текущем каталоге ссылку `mydsts`, указывающую на каталог

`/afs/cern.ch/atlas/user/r/rd6/public/trt02/dsts/` .

Если при попытке создать ссылку с именем `linkname` в текущем каталоге уже существует файл или ссылка с таким же именем, то будет выдано сообщение об ошибке. Форсировать создание новой ссылки можно с помощью опции **–f**:

ln –sf filename linkname

Как мы видели в примере, приведенном в разделе 2.3, символическая ссылка отмечается буквой `l` в первой позиции подробного листинга, а после имени ссылки и символов `->` следует имя того файла или каталога, на который она указывает. Отметим, что символическая ссылка практически не занимает место на диске – ее размер составляет несколько байт.

После создания ссылки с ней можно проводить те же операции, что и с обычным файлом или каталогом: выводить содержимое на экран, редактировать, запускать на выполнение, если ссылка указывает на программу и т.д. При этом все действия реально будут совершаться с файлом или с каталогом, на которые указывает ссылка. Однако если удалить ссылку:

rm linkname

то файл или каталог, на которые она указывает, останутся нетронутыми.

Символическая ссылка очень часто применяется в файловой системе Linux. Представим, например, что мы много раз запускаем программу, входные данные для которой хранятся в файле с длинным именем:

`/afs/cern.ch/atlas/user/r/rd6/public/trt02/dsts/dst01553.ntup`

Каждый раз набирать такое имя очень неудобно, к тому же можно легко ошибиться. Если же выполнить команду:

ln –s /afs/cern.ch/atlas/user/r/rd6/public/trt02/dsts/dst01553.ntup

то в текущем каталоге создастся ссылка к данному файлу с коротким именем `dst01553.ntup`, к которой можно обращаться так же, как и к самому файлу.

Другое применение ссылок можно проиллюстрировать на следующем примере. Допустим, что некоторое приложение использует большое количество файлов с библиотеками, расположенных в каталоге `/usr/lib`. Представим, что мы устанавливаем на компьютере другое приложение, которому также нужны эти библиотеки. Но для этого второго приложения по каким-то причинам требуется, чтобы путь к библиотекам был, например, такой: `/usr/local/lib`. Можно было бы просто скопировать нужные файлы библиотек из каталога `/usr/lib` в `/usr/local/lib`. Но такое решение имеет два существенных недостатка. Во-первых, скопированные файлы займут место на диске, а во-вторых, при внесении каких-то изменений в библиотеки необходимо все время помнить о второй копии. Проблема решается просто:

cd /usr/local/lib
*ln –s /usr/lib/** .

Итак, символическая ссылка позволяют практически без затрат дискового пространства совместно разным пользователям и приложениям использовать ресурсы в виде файлов и каталогов.

2.7. Просмотр текстовых файлов

Большие текстовые файлы обычно просматривают с помощью программ-редакторов. Для быстрого просмотра зачастую удобнее пользоваться специальными командами.

Команда

cat filename

выведет на экран терминала содержимое текстового файла `filename`. Если добавить опцию **–n**, то строки файла будут пронумерованы. Если текст файла не умещается на экране, то вместо **cat** удобнее пользоваться командами **more** или **less** – они выводят на экран содержимое файла постранично. Для продвижения по тексту файла по направлению вперед в программах **more** и **less** используется клавиша <`space`>, по направлению назад – клавиша <`b`>. Прервать просмотр можно, нажав <`q`>. Команда **less** обладает широкими возможностями поиска в просматриваемом документе. В частности, */pattern* ищет первое появление буквосочетания `pattern` в тексте файла по направлению вперед, а *?pattern* – по направлению назад.

Команда

head `[`*–n*`]` *filename*

выведет на экран первые `n` (по умолчанию – 10) строк файла `filename`. А команда

tail `[`*–n*`]` *filename*

выведет последние `n` строк файла.

Приведенные в этом разделе команды часто используются в составных командах – конвейерах (раздел 2.10) для просмотра текста, выводимого на экран другими командами и программами.

2.8. Поиск текста в файлах

Команда **grep** используется для поиска заданного текста в файлах или выходном потоке какой-либо команды или программы.

grep pattern filename

ищет в файле `filename` образец текста `pattern` и выводит на экран строку с найденным образцом. Например,

grep Dimension `*`*.f*

выдаст на экран все строки в файлах текущего каталога с расширением `.f`, в которых содержится слово `Dimension`.

По умолчанию образец искомого текста является регистро-зависимым. Таким образом, в предыдущем примере строки, содержащие слово `dimension`, найдены не будут. Чтобы сделать поиск регистронезависимым, используют опцию **–i**. Команда

grep –i dimension `*`*.f*

найдет оба образца: и `Dimension` и `dimension`.

Если в образце `pattern` кроме букв и цифр встречаются другие символы, например символ пробела или "." и т.п., то искомый текст заключают в кавычки:

*grep –i 'dimension(4,' *.f*

Если вместо имени файла `filename` указать имя каталога, то опция **–r** сделает поиск рекурсивным: будут просмотрены все файлы и каталоги, расположенные внутри указанного. Опция **–l** укажет на то, что требуется вывести на экран только имена файлов, а не сами строки, содержащие заданный образец. Так, команда

grep –rl pattern dirname

просмотрит все файлы в каталоге `dirname` и во вложенных в него каталогах и выведет на экран имена тех файлов, в тексте которых встречается последовательность символов `pattern`.

Опция **–n** пронумерует выводимые строки. Наконец, опция **-v** инвертирует поиск: будут выведены те строки, в которых ***не*** встречается заданный образец `pattern`.

О том, как использовать команду **grep** в выходном потоке команд или программ, рассказывается в разделе 2.10.

2.9. Перенаправление ввода-вывода

Вывод таких команд, как **cat** или **ls** происходит на выходное устройство (standard output), которым по умолчанию является экран терминала. Входным устройством (standard input) для выполняемых команд по умолчанию является клавиатура. Изменить эти правила можно перенаправлением ввода и/или вывода с помощью знаков "<" и ">". Так, команда

cat filename1 > filename2

вместо того чтобы выводить текст файла `filename1` на экран, запишет этот текст во вновь созданный файл `filename2`. Если файл `filename2` уже существует, его старое содержимое будет перезаписано. Команда

cat filename1 >> filename2

присоединит содержимое файла `filename1` к концу файла `filename2`, не удаляя его старого содержания. Поскольку в команде **cat** можно перечислять несколько файлов, то самый простой способ объединить несколько файлов в один – это набрать команду:

cat filename1 filename2 ... > filename

А, например, команда

ls –alR ~ > ls.list

запишет в файл `ls.list` полный листинг всех каталогов и файлов пользователя.

Иногда возникает необходимость направить выходной поток в какой-либо дисковый файл, но при этом одновременно и видеть его на экране. Для этого можно использовать команду **tee**:

ls –alR | tee filename

Такая конструкция выведет на экран подробный рекурсивный листинг текущего каталога и одновременно запишет его в файл `filename`. (Подробнее об использовании знака "|" см. раздел 2.10).

Теперь предположим, что пользователь запускает некую программу **scitrd**, которая требует ввода данных с клавиатуры. Количество данных может быть достаточно большим и при их вводе легко ошибиться. К тому же введенные с клавиатуры данные нигде не сохраняются и спустя какое-то время пользователь может и не вспомнить – какие именно входные данные использовала программа. Удобным решение в этом случае будет предварительная запись данных в какой-либо файл, скажем `scitrd.inp`. Если теперь выполнить команду

scitrd < scitrd.inp

то запущенная программа **scitrd** вместо того, чтобы ожидать ввода необходимых данных с клавиатуры, будет построчно читать их из файла `scitrd.inp`. Таким образом, знак "<" служит для перенаправления стандартного ввода.

Кроме потоков стандартного ввода и вывода, существует поток ошибок (standard error), в который направляются сообщения системы об ошибках, если таковые возникают при выполнении программы или команды. По умолчанию поток ошибок направляется на экран терминала. Если вы хотите перенаправить поток ошибок в файл, используйте для перенаправления символы `>2`:

scitrd >2 scitrd.err

Можно также объединить поток ошибок с выходным потоком standard output. Знаком объединения двух потоков служит "`>&`". Так, команда

scitrd < scitrd.inp >& scitrd.out

запустит программу **scitrd**, которая будет читать входные данные из

файла `scitrd.inp` и направлять свой вывод, а также поток ошибок в файл `scitrd.out`.

Иногда может возникнуть необходимость избавиться от вывода на экран потоков стандартного вывода и/или ошибок. Например, вы запускаете некую программу **thcal**, которая в процессе своей работы выдает на экран массу ненужных вам сообщений. Вы можете перенаправить их в специальный файл `/dev/null`, который является своего рода "черной дырой" в том смысле, что все, что направлено в этот файл, безвозвратно пропадает:

thcal `>&` */dev/null*

Программа **thcal** отработает "молча", не выдавая ни на экран, ни в дисковый файл никаких сообщений.

Подчеркнем, что перенаправление ввода-вывода затрагивает только ввод с клавиатуры и вывод на экран. Если какая-то программа должна читать данные из дискового файла или записывать что-то в файл, то эти операции перенаправлением ввода-вывода никак не затрагивается.

2.10. Конвейер

Содержание этого раздела не связано напрямую с понятием "работа с файлами". Однако применение конвейера в командной строке нам уже встречалось в примерах в предыдущих разделах и будет часто встречаться далее в этой книге. Поэтому для удобства дальнейшего изложения материала нам представляется полезным рассмотреть эти вопросы здесь.

Конвейер (pipe) является еще одним инструментом, позволяющим управлять потоками ввода-вывода. Конвейер направляет выходной поток одной команды на вход другой. Знаком конвейера служит символ "|". Так, команда

ls –alR ~ | grep dat

сначала создаст рекурсивный листинг каталогов пользователя, но не выведет этот листинг на экран, а подаст на вход другой команды, стоящей за знаком "|", в данном случае – команды **grep**. Команда **grep** отберет из этого потока только те строки, в которых содержатся символы `dat` и уже их выведет на экран. Таким образом пользователь может найти у себя все файлы и каталоги, в имени которых содержатся символы `dat`. Если такой листинг окажется слишком длинным,

можно применить более сложную конструкцию:

ls –alR ~ | grep dat | more

Следующий пример показывает, как можно использовать конвейер для вывода на экран листинга девяти самых "новых" по времени модификации файлов из текущего каталога:

ls –lt | head

Еще один пример. Допустим, нужно запустить некую программу **root4star**, которую пользователь уже запускал ранее, но забыл полный путь к ней. Список выполненных ранее команд, как уже говорилось, можно получить с помощью команды **history**. Тогда, набрав

history | grep root4star

мы увидим на экране:

```
36 15:32 /afs/rhic/star/packages/SL00m/.i386_redhat61/bin/root4star
```

Здесь 36 – порядковый номер выполненной команды, за которым следует время, когда она была выполнена и сама команда (в данном случае – это имя выполняемого файла с полным путем к нему). Теперь можно либо просто повторить эту команду, набрав *!36*, либо скопировать ее мышкой через буфер обмена в командную строку и редактировать – ввести, например, дополнительные входные параметры для программы **root4star**.

2.11. Архивирование файлов и каталогов

Команда **gzip** применяется для компрессии файла, если необходимо, чтобы он занимал меньше места на диске. Команда

gzip filename

"сожмет" файл `filename` и запишет результат компрессии в новый файл `filename`.gz. Оригинальный файл `filename` при этом не сохраняется. С помощью одной команды **gzip** можно сжать (по отдельности) сразу несколько файлов, если использовать шаблон "*". Например, команда

*gzip *.f*

сожмет все файлы с расширением `.f` в текущем каталоге. А команда

gzip –r dirname

сожмет все файлы в каталоге `dirname` и во всех вложенных в него каталогах.

Обратная операция декомпрессии файла осуществляется командой **gunzip**:

gunzip filename[*.gz*]

Команда **tar** является мощным средством архивации файловой системы Linux. Команда "упаковывает" файлы и каталоги в один выходной файл. При этом в созданном архивном файле сохраняется вся информации о структуре каталогов, атрибутах файлов и т.п. Итак, команда

tar –c dirname –f filename.tar

создаст новый файл `filename.tar`, в котором будут упакованы все каталоги и файлы, расположенные внутри каталога `dirname`. По умолчанию команда **tar** не проводит компрессию упакованных файлов. Если же мы хотим уменьшить размер создаваемого архивного файла, нужно воспользоваться опцией –**z**:

tar –cz dirname –f filename.tgz

Отметим, что расширения файлов `.tar` и `.tgz` являются рекомендованными для файлов, созданных командой `tar` без компрессии и с компрессией соответственно. (Вместо `.tgz` иногда используется `.tar.gz`).

Обратим внимание на два важных момента. По умолчанию команда **tar** при упаковке убирает символ "/" из имени `dirname`, если такой символ стоит первым в имени. Другими словами, если в качестве `dirname` был указан абсолютный путь, то в архивном файле этот путь будет сохранен как относительный, без лидирующего символа "/". Если необходимо сохранить лидирующий "/" в имени каталога, используйте опцию –**P**:

tar –czP /dirname –f filename.tgz

Вторая особенность связана с архивацией символических ссылок, о которых говорилось в разделе 2.6. По умолчанию команда **tar** сохраняет лишь сами ссылки, т.е. имена указателей, но не собственно файлы, на которые указывают эти ссылки. Если мы хотим заархивировать именно файлы, а не ссылки на них, воспользуемся опцией –**h**:

tar –czh dirname –f filename.tgz

Распаковать созданный командой **tar** файл можно так:

tar –xf filename.tar

или

tar –xzf filename.tgz

если при архивации была применена компрессия. Добавив в команде опцию **–v**, можно будет наблюдать на экране имена файлов в процессе распаковки. Эту опцию можно использовать и при архивации.

Еще раз обращаем внимание на то, что если при архивации не использовалась опция **–P**, то сохраненный в архивном файле путь будет относительным. Это означает, что при распаковке вся структура каталогов будет развернута внутри текущего каталога, в котором выполняется команда **tar –x**.

Посмотреть содержимое архивного файла без его распаковки можно с помощью опции **–t**:

tar –tf filename.tar

или

tar –tzf filename.tar

если архив при создании подвергся компрессии.

Команда **tar** имеет множество опций, позволяющих, например, заменять или удалять отдельные файлы в уже созданном архиве или исключать какие-то файлы из процесса архивации или распаковывать не весь архив, а только его часть и т.п. Поэтому команда **tar** и создаваемые ею файлы очень широко используются для резервного копирования системы и для переноса множества файлов и/или каталогов с одного диска на другой через сеть, дискету, флэш-память или CD диск. Многие пакеты программ для Linux, хранящиеся в Интернет, записаны в `tar` формате. Отметим также, что форматы `gz, tar` и `tgz` распознаются программами-архиваторами для Windows типа WinZip.

2.12. Поиск файлов

Для поиска файлов в файловой системе используется команда **find**. Формат команды **find** несколько отличается от формата других команд в Linux:

find dirname [*expression*]

где `expression` – специальным образом сконструированное выражение, задающее критерий отбора при поиске. **find** обладает огромными возможностями: вы можете искать файлы по именам, времени создания или последнего обращения к файлу, размеру, правам доступа к файлам и т.п. Просмотрите справочные и info страницы команды **find** – там вы найдете множество примеров использования команды. Приведем лишь некоторые.

Команда

*find dirname –name '*pattern*'*

просмотрит список файлов в каталоге `dirname` и во вложенных в него каталогах и выведет на экран те файлы, в имени которых встречается последовательность символов `pattern`. Например,

find ~ –name '.ntup'*

найдет все файлы пользователя, имеющих расширение `.ntup`. Такая команда:

find ~ –mtime –2

выдаст список всех файлов пользователя, модифицированных в течение последних двух дней (48 часов), а

find dirname –daystart –type f –mtime 0

выдаст список тех файлов (исключая каталоги и ссылки), в каталоге `dirname`, которые были модифицированы сегодня. Выполнив

find . –size +30000k

получим список файлов в текущем каталоге, чей размер превышает 30000 Кбайт. А команда

*find dirname –name '*namepattern*' | xargs grep –l 'stringpattern'*

найдет все файлы внутри каталога `dirname`, в именах которых встречается последовательность символов `namepattern`. Затем командой **xargs grep –l 'stringpattern'** будут отобраны те файлы, в тексте которых содержится образец `stringpattern`.

Обратим внимание, что если вместо '`*namepattern*`' в команде **find** указать просто '`namepattern`', то будут найдены файлы, имена которых в точности совпадают с `namepattern`, в отличие от команды **grep**, которая ищет образец имени. Другими словами, если, например, в текущем каталоге существуют два файла с именами `01553.ntup` и `dst01553.ntup`, то команды

ls | grep '01553.ntup'

и

*find –name '*01553.ntup*'*

найдут оба этих файла, а

find –name '01553.ntup'

найдет только первый из них.

Если в вашей системе была создана соответствующая база данных, то еще одним способом поиска файлов может служить команда **locate**. Поскольку **locate** ведет поиск по всей файловой системе, ей удобно пользоваться в случае, когда вы не знаете, в каком каталоге расположен искомый файл. Например,

locate '.ntup'

быстро найдет все файлы в системе (не только в текущем каталоге), имеющих расширение `.ntup`. Команда **locate** ищет файлы по образцу имени, т.е. для приведенного выше примера

locate 01553.ntup

будут найдены оба файла с именами `01553.ntup` и `dst01553.ntup`. Отметим еще одно существенное отличие в работе команд **find** и **locate**. Если первая находит только файлы, собственные имена которых подпадают под шаблон поиска, то **locate** укажет также на все файлы и каталоги, для которых под шаблон попадает полный путь к ним.

Все сказанное выше относится к поиску файлов любых типов. Для исполняемых файлов и программ можно также вести поиск с помощью команды **which**, которая укажет полный путь к исполняемому файлу. Так,

which mkdir

выдаст:

`/bin/mkdir`

т.е. исполняемый файл (в данном случае – команда) **mkdir** найден в каталоге `/bin`. Заметим однако, что с помощью **which** можно найти только те исполняемые файлы, которые расположены в каталогах, перечисленных в списке переменной окружения $PATH (см. раздел 4.2).

Итак:

- команда **find** используется для поиска файлов любого типа в заданном каталоге, часто с применением дополнительных критериев отбора: времени создания файла, его размера и т.п.
- команда **locate** используется для быстрого поиска файлов любого типа по их имени во всей файловой системе, при условии, что эти файлы занесены в специальную базу данных.
- команда **which** используется для поиска по имени исполняемых файлов во всей файловой системе, при условии, что путь к этому исполняемому файлу указан в переменной окружения $PATH.

2.13. Работа со сменными носителями

Файловая система Linux устроена таким образом, что гибкий диск, компакт-диск или флэш-память с интерфейсом USB, вставленные в соответствующий дисковод или разъем, не становятся автоматически доступны пользователю – их необходимо "смонтировать". Это можно сделать либо через соответствующие иконки в графических оболочках KDE или GNOME, либо командой **mount**. Эта команда используется также для монтирования разделов жесткого диска и других устройств. Но если разделы жесткого диска обычно монтируются автоматически при загрузке системы, то для сменных носителей монтирование необходимо задать явно. В команде **mount** обычно необходимо указать условленное имя монтируемого устройства и точку монтирования – имя существующего пустого каталога, к которому это устройство необходимо подсоединить. Так,

mount /dev/fd0 /mnt/floppy

сделает доступным содержимое дискеты в каталоге `/mnt/floppy`. Аналогичная команда для монтирования CD диска выглядит следующим образом:

mount /dev/cdrom /mnt/cdrom

Содержимое CD диска будет доступно в каталоге `/mnt/cdrom`.

Флэш-память или другие сменные устройства с USB интерфейсом обычно монтируется командой

mount /dev/sda1 dirname

где `dirname` – имя каталога, в котором будет видно содержимое носителя. В отличие от каталогов `/mnt/floppy` и `/mnt/cdrom`, для USB устройств в Linux пока не существует общепринятого имени для точки монтирования. Возможно, на вашем компьютере это будет нечто вроде `/mnt/flash` или `/mnt/usb`.

Обычно команда **mount** автоматически распознает тип файловой системы на сменном носителе, Если это не так, то его необходимо указать явно через опцию **–t**, например:

mount –t msdos /dev/fd0 /mnt/floppy

или

mount –t iso9660 /dev/cdrom /mnt/cdrom

Типы поддерживаемых файловых систем можно узнать из справочной страницы команды **mount**.

После того, как устройство смонтировано, вы можете работать с файлами на сменном носителе так же, как и с любыми другими: копировать, удалять (не на CD, конечно) и т.п. Например, команда

cp /mnt/cdrom/dirname/filename .

скопирует в текущий каталог файл `filename` из каталога `dirname` на CD диске. А команда

cp myfile.tgz /mnt/floppy/

перепишет файл `myfile.tgz` из текущего каталога на дискету.

После окончания работы со сменным носителем, устройство необходимо "размонтировать". Это делается командой **umount**:

umount dirname

где `dirname` – имя точки монтирования, которое применялось в соответствующей команде **mount**. Т.е., скажем, для флоппи дисковода:

umount /mnt/floppy

Если вынуть сменный носитель, не проведя размонтирования, то вы можете не обнаружить на нем записанных файлов и даже рискуете испортить всю файловую систему на носителе.

По умолчанию монтировать любые устройства в Linux может только так называемый супер-пользователь (root). Если это правило в вашей системе не изменено, то обычному пользователю не удастся командой **mount** смонтировать сменный носитель. Проверить, как обстоит дело в вашей системе, можно, посмотрев содержимое файла `/etc/fstab`:

more /etc/fstab

Если вы увидите ключи "user" или "users" в строках типа

```
/dev/cdrom /mnt/cdrom  iso9660 noauto,user,owner,ro 0 0
/dev/fd0   /mnt/floppy msdos   noauto,user,owner    0 0
/dev/sda1  /mnt/usb    auto    noauto,user          0 0
```

то обычному пользователю разрешено монтировать CD диски, дискеты и USB устройства. Заметим также, что если устройства для сменных носителей указаны в файле `/etc/fstab`, то команду **mount** можно упростить, указывая в ней только точку монтирования, например:

mount /mnt/usb

для USB устройства.

Для работы с дискетами можно воспользоваться программами из пакета утилит **mtools**, если этот пакет установлен на вашем компьютере. Программы, входящие в состав **mtools**, не требуют наличия у пользователя особых привилегий. Для более подробного ознакомления с этими программами смотрите соответствующую info страницу:

info mtools

2.14. Разные полезные команды

Часто возникает необходимость сравнить содержимое двух файлов. Это можно сделать с помощью команды **diff**:

diff filename1 filename2

На экран будут выведены пронумерованные строки, которые отличаются в файлах `filename1` и `filename2`. Сравнивать можно не только текстовые, но и двоичные файлы, но в этом случае команда просто сообщает – отличаются файлы или нет. Удобным инструментом для сравнения текстовых файлов послужит также программа **tkdiff**, если она установлена на вашем компьютере. Возможностью сравнения текстовых файлов обладает также редактор **emacs** (см. раздел 3.3).

Команда

file filename

попытается определить тип файла `filename`: есть ли это программа, данные, графический файл определенного формата, архив и т.п. Причем тип файла определяется не по имени или расширению, а по содержимому самого файла.

Определить размер, занимаемый файлами на диске, можно с помощью команды **du**. Команда имеет много опций, но самый простой вариант использования:

du –s dirname

Такая команда выдаст размер (в килобайтах), занимаемый всеми файлами в каталоге `dirname`, ***включая*** вложенные каталоги. Без опции –**s** команда **du** распечатает также размеры каждого из вложенных каталогов. Если вы смонтировали флоппи-диск, то команда

du –s /mnt/floppy

поможет быстро оценить – достаточно ли свободного места на дискете. Если необходимо получить размер, занимаемый на диске файлами, ***исключая*** вложенные каталоги, воспользуйтесь опцией –**S**.

Команда **df** выдаст информацию о локальных и сетевых устройствах, смонтированных на вашем компьютере: разделах жесткого диска, сетевых дисках, о сменных устройствах. Будут также представлены данные о размерах занятого и свободного пространства на этих устройствах.

Команда **quota** покажет размер дискового пространства, которое отведено данному пользователю.

Команда

wc filename

подсчитает количество строк, слов и символов в файле `filename`. А, например, такое применение команды **wc**

ls | wc

поможет быстро подсчитать количество файлов и каталогов в текущем каталоге.

Команда **sort** используется для сортировки строк файла или выходного потока в конвейере. Например,

du –S | sort –n

выведет на экран список каталогов, находящихся внутри текущего, в порядке возрастания размера занимаемого ими дискового пространства. А конструкция

ls –AlR | sort +4n | tail

рассортирует все файлы в текущем каталоге и во всех вложенных в него каталогах по их размеру, и выведет на экран список десяти самых больших по размеру файлов.

Команда

touch filename

заменит атрибут `modify time` (время изменения) файла `filename` на текущее время. Если файл `filename` не существует, команда **touch** создаст пустой файл с таким именем.

Программы **od** и **hexdump** служат для вывода на экран содержимого файлов (не обязательно текстовых) в восьмеричном, шестнадцатеричном и других недесятичных форматах.

3. Редактирование текстовых файлов

На вашем компьютере может быть установлено несколько редакторов текстовых файлов. Здесь мы кратко рассмотрим три из них: **vi**, **pico** и **emacs**.

3.1. Редактор vi

Редактором **vi**, на наш взгляд, не очень удобно пользоваться. Однако этот редактор, или его более современная разновидность – **vim** – входит в состав любых UNIX систем – в какой бы операционной системе семейства UNIX вы не оказались, вы можете быть уверены, что редактор **vi** там установлен. Кроме того, он довольно компактен и по этой причине часто включается в набор так называемых rescue дискет, предназначенных для аварийного восстановления системы.

В редакторе **vi** существует несколько режимов работы. Наиболее часто используются два из них – командный режим и режим ввода текста. В командном режиме **vi** нажатие каждой клавиши клавиатуры означает определенную команду редактора. В этот режим вы попадаете при запуске редактора:

vi filename

где `filename` – имя редактируемого файла. В командном режиме **vi** вы можете перемещаться по тексту с помощью клавиш <↑>, <↓>, <←> и <→> (если эти клавиши у вас не работают, попробуйте клавиши <`h`>, <`j`>, <`k`> и <`l`>). Можно пользоваться функциями поиска: */pattern* ищет первое появление образца `pattern` в тексте по направлению вперед, а *?pattern* – по направлению назад. Нажатие клавиши <`X`> удаляет символ сзади курсора, а клавиши <`x`> – над курсором. Стереть текст от позиции курсора до конца строки можно, нажав клавиши <`d`><`$`>, а стереть всю строку: <`d`><`d`>.

Чтобы перейти в режим ввода текста, нажмите клавишу <`i`>. В этом режиме все символы, которые вы набираете на клавиатуре, являются вводимым текстом. Закончив набирать текст, можно вернуться обратно в командный режим, нажав клавишу <`Esc`>.

Еще одним режимом **vi** является так называемый режим последней строки. Все команды, вводимые в этом режиме, начинаются с символа ":". Ввод двоеточия перемещает курсор в нижнюю часть экрана, где следует ввести оставшуюся часть команды. Режим последней строки

используется главным образом для выполнения команд общего характера, в частности:

wq – записать произведенные изменения в файл и выйти из редактора;
w – записать изменения в файл без выхода из редактора;
w filename – записать текущий редактируемый буфер в файл с именем `filename`;
q – выйти из редактора;
q! – не сохранять произведенные изменения и выйти из редактора;
!sh – войти в командную оболочку (shell) системы. Здесь вы можете выполнять любые команды оболочки. Чтобы вернуться обратно в редактор, наберите *exit*;
!command – выполнить команду оболочки (командного интерпретатора операционной системы) `command`.

3.2. Редактор pico

Редактором **pico** можно пользоваться при редактировании небольших фрагментов текста. Он не требует открытия нового X окна, что удобно при редактировании файлов на удаленном компьютере в случае, когда связь не очень быстрая. Кроме того, на основе **pico** построено редактирование текста в **pine** (раздел 8.4) – программе работы с электронной почтой. Рекомендуется запускать **pico** с опцией **–w**, что упростит редактирование длинных строк.

Команда

pico –w filename

загрузит **pico** для редактирования файла `filename`. На рис. 2 приведен пример редактирования файла редактором **pico**. Перемещаться по тексту редактируемого файла можно как с помощью клавиш <↑>, <↓>, <←> и <→>, так и постранично клавишами <`PgUp`> и <`PgDn`> на дополнительной клавиатуре. Набор нового текста в редакторе **pico** осуществляется непосредственно с клавиатуры, без перехода в какой-либо специальный режим. Также обычным образом работают клавиши <`backspace`> и <`Ctrl-d`>, удаляющие символы, предшествующий курсору и непосредственно над курсором.

Постраничная навигация возможна также с помощью комбинаций клавиш <`Ctrl-v`> (вперед по тексту) и <`Ctrl-y`> (назад), что видно из подсказки в двух нижних строках экрана. Тут же можно видеть и другие подсказки: <`Ctrl-k`> удаляет одну строку текста, на которой

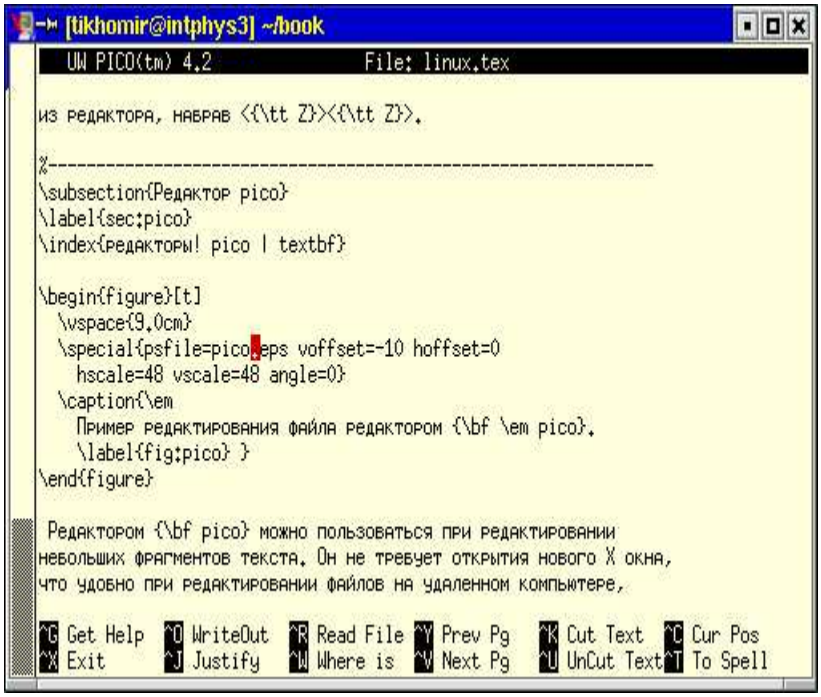

Рис. 2. *Редактирование файла в редакторе* ***pico***

расположен курсор, а <Ctrl-u> – восстанавливает удаленную строку. <Ctrl-w> используется для поиска заданного образца в тексте, а <Ctrl-t> – для проверки орфографии.

Записать сделанные в файле изменения можно, нажав <Ctrl-o>, выйти из редактора – <Ctrl-x>. Отметим, что старая версия отредактированного файла при этом не сохраняется. Нажав <Ctrl-g>, мы получим справку по работе в **pico**.

3.3. Редактор emacs

Редактор **emacs** является очень мощной программой и может быть рекомендован как основной инструмент при редактировании текстовых файлов в Linux. Кроме собственно редактирования, **emacs** может использоваться, например, для компиляции программ, трансля-

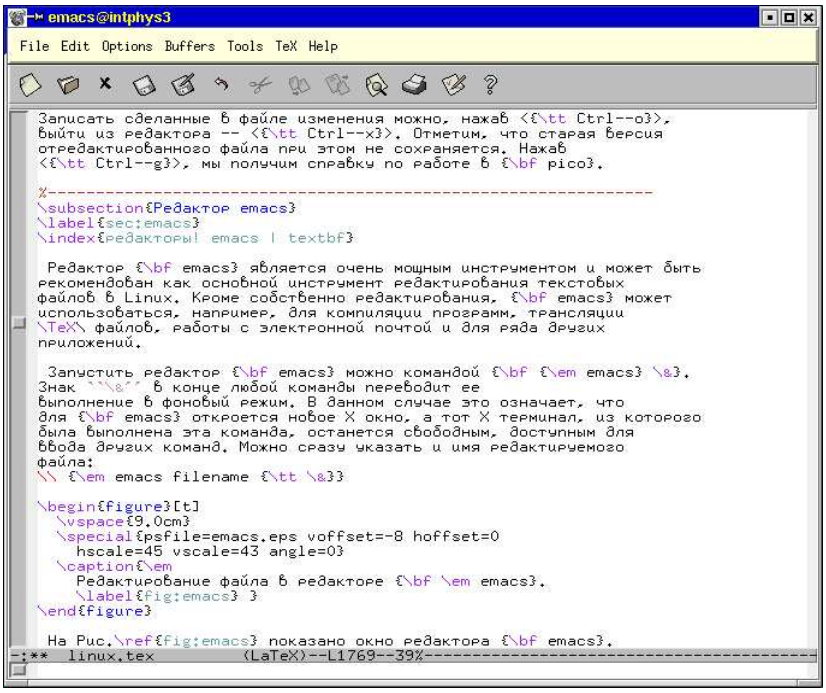

Рис. 3. *Редактирование файла в редакторе* ***emacs***

ции TEX файлов, работы с электронной почтой и для ряда других приложений.

Запустить редактор **emacs** можно командой ***emacs &***. Знак "&" в конце любой команды переводит ее выполнение в фоновый режим. В данном случае это означает, что для **emacs** откроется новое X окно, а тот терминал, из которого была выполнена эта команда, останется свободным, доступным для ввода других команд. Можно сразу указать и имя редактируемого файла:

emacs filename &

На рис. 3 показано окно редактора **emacs**. Как можно видеть, кроме собственно поля с редактируемым текстом, занимающего бóльшую часть окна, в редакторе есть верхняя строка с заголовками выпадающих меню, строка с иконками для наиболее часто употребляемых команд и две строки внизу. Вторая строка снизу является информаци-

онной: в ней показано имя файла – `linux.tex`; режим редактирования (mode), который зависит от типа файла, в данном случае – `LaTeX`; номер строки, в которой стоит курсор (1769); процент от полного размера файла, считая от его начала до текущей позиции. Обратим внимание на два символа "`*`" в левой части информационной строки: они означают, что редактируемый файл был изменен, но эти изменения еще не были записаны на диск. Нижняя строка служит для ввода некоторых команд, имен файлов и т.п.

При интенсивной работе с редактором вы можете держать его все время открытым и подгружать по мере необходимости редактируемые файлы, каждый из которых будет сохраняться в собственном буфере (меню `Buffers`). Сделав необходимую редакцию, можно записать отредактированный буфер в файл на диск и загрузить новый файл или вернуться к другому редактируемому буферу и т.д. Заметим, что при записи на диск новой версии файла `filename` старая версия также сохраняется с именем `filename`$\sim$.

В зависимости от типа редактируемого файла, который определяется по расширению в имени файла, редактор будет находиться в одном из режимов: Text mode для текстовых файлов, Fortran или C modes для программ на языках FORTRAN и C соответственно, LaTeX mode для TEX файлов и т.д. Если **emacs** не знаком с типом редактируемого файла, соответствующий режим будет называться Fundamental mode. В зависимости от режима меняется "поведение" редактора: набор меню, заголовки которых видны в верхней строке; выделение цветом разных фрагментов текста; отступ в тексте при нажатии клавиши <`Tab`> и др.

Меню (верхняя строка редактора) можно открыть, наведя на его заголовок курсор мыши и нажав на ее левую кнопку. Многие команды из меню можно вводить и с клавиатуры, что зачастую бывает быстрее. Приведем список наиболее часто используемых команд:

- <`Ctrl-x`><`Ctrl-f`> : загрузить в буфер редактора новый файл с диска для редактирования. Имя файла будет предложено ввести в нижней строке окна редактора. Отметим, что при вводе имени можно пользоваться клавишей <`Tab`> для автоматического дополнения вводимого имени, как при редактировании командной строки;

- <`Ctrl-x`><`Ctrl-i`> : считать с диска файл и ввести его текст в то место буфера, где расположен текстовый курсор;

- <Ctrl-s> : искать текст по направлению вперед;
- <Ctrl-r> : искать текст по направлению назад. При выполнении этих команд искомый текст будет предложено ввести в нижней строке окна редактора. Поиск будет регистро-независимым, если образец искомого текста будет введен в нижнем регистре. Т.е., если пользователь введет для поиска строку *dimension* то найдены будут и `dimension` и `DIMENSION` и `Dimension`. Если же введенное для поиска слово будет содержать символы верхнего регистра, например, *Dimension*, то и поиск будет проведен с учетом регистра;
- <Esc><%> : поиск и замена одного фрагмента текста на другой;
- <Ctrl-x><2> : разделить окно редактора на две части (например, для редактирования или сравнения двух файлов);
- <Ctrl-x><1> : вернуться к режиму с одним окном редактора;
- <Esc><< > : встать в начало редактируемого буфера;
- <Esc><> > : встать в конец редактируемого буфера;
- <Ctrl-l> : передвинуть текст так, чтобы строка, в которой стоит текстовый курсор, оказалась в центре окна редактирования;
- <Ctrl-spacebar> : поставить невидимый маркер в месте расположения текстового курсор;
- <Ctrl-w> : удалить выделенную область текста. Эта область выделяется либо с помощью мыши, как рассказано ниже, либо определяется как область между маркером и текущим положением текстового курсора;
- <Ctrl-y> : восстановить удаленный текст;
- <Esc><$> : проверка орфографии слова, на котором или сразу за которым стоит текстовый курсор;
- <Ctrl-_> : undo, т.е. отменить последнее сделанное изменение (знак "_" здесь есть underscore – знак подчеркивания);
- <Ctrl-x><Ctrl-s> : записать редактируемый буфер на диск в файл с тем же именем, что и редактируемый;

- <Ctrl-x><Ctrl-w> : записать редактируемый буфер на диск, дав записываемому файлу новое имя;

- <Ctrl-x><Ctrl-c> : закончить редактирование и закрыть редактор;

Кроме того, работают комбинации клавиш, о которых уже говорилось ранее в разделе о редактировании командной строки: <Ctrl-d> удаляет символ над курсором, <Ctrl-a> передвигает текстовый курсор в начало командной строки, а <Ctrl-e> – в конец строки. Нажав <Ctrl-k>, можно удалить часть строки от позиции курсора до конца строки. Заметим, что комбинация <Ctrl-u>, которая в командной строке удаляет всю строку, в **emacs** не работает.

Очень удобно пользоваться буфером обмена, как уже рассказывалось выше. Наведите курсор мыши на начало выделяемого текста и нажмите левую кнопку мыши. Затем, не отпуская левую кнопку, подведите курсор мыши к концу выделяемого текста. Если теперь отпустить левую кнопку, то содержимое выделенного текста будет скопировано в буфер обмена. Можно и по-другому выделить текст: щелкните левой кнопкой мыши в начале выделяемого текста, отпустите кнопку и, переведя курсор мыши в конец текста, щелкните правой кнопкой. Теперь содержимое буфера обмена можно скопировать в любой другое место в редактируемом тексте: подведите курсор мыши к нужному месту и нажмите среднюю кнопку мыши (или одновременно левую и правую, если у вашей мыши нет средней кнопки). Напомним также, что выделять таким образом текст и копировать его в буфер обмена можно не только в окне редактора, но и в любом другом X окне.

Редактор **emacs**, как уже отмечалось, обладает очень широкими возможностями. При знакомстве с ним можно порекомендовать для начала просмотреть Emacs Tutorial и Manual из меню `Help`. Настройки **emacs** содержатся в установочном файле `.emacs` в корневом каталоге пользователя. В разделе 9 будет рассказано, как настроить **emacs** для того, чтобы можно было редактировать тексты на русском языке.

4. Рабочее окружение

Под рабочим окружением подразумевается совокупность изменяемых параметров операционной системы в целом и отдельных приложений, от которых зависит их поведение. Здесь рассматривается настрой-

ка некоторых параметров рабочего окружения для нужд отдельного пользователя.

4.1. Командные оболочки

Командная оболочка – это интерпретатор команд, интерфейс между пользователем и ядром операционной системы. В Linux, как и в других ОС семейства UNIX, существуют два семейства оболочек. Первое семейство основано на Bourne Shell (оригинальная оболочка называлась просто `sh`) и включает в себя также `ksh`, `bash` и `zsh`. В другое семейство, основанное на C Shell (`csh`), входит также `tcsh`. В какой именно оболочке работает пользователь, зависит от того, какой login shell был установлен для него системным администратором. Вы можете определить, каков ваш login shell, набрав команду:

echo $SHELL

Для разных оболочек формат некоторых команд Linux отличается, что будет всегда далее оговариваться. Различен также синтаксис командных файлов-сценариев (scripts). Кроме того, отличаются имена и порядок выполнения так называемых сценариев автозагрузки (startup files) – командных файлов, которые автоматически выполняются при входе в систему. Считается, что оболочки семейства Bourne shells лучше подходят для написания файлов-сценариев, а оболочки C shells – для интерактивного использования, хотя это в значительной мере дело вкуса и привычки.

В принципе, пользователь может в процессе работы сменить shell, просто набрав в командной строке имя новой оболочки. Например, если вы работаете в `tcsh` и хотите временно сменить командную оболочку на `zsh`, наберите

zsh

Однако необходимо помнить, что такая смена командной оболочки не идентична положению, при котором оболочка `zsh` изначально является login shell данного пользователя. Дело в том, в этих случаях могут по разному выполняться ряд сценариев автозагрузки. Кроме того, смена командной оболочки произойдет только в том окне терминала, в котором была выполнена команда **zsh**, и перестанет действовать, когда пользователь выйдет из системы. Если вы решили изменить свой login shell, воспользуйтесь командой **chsh**. Это изменение вступит в силу при следующем входе в систему. Список всех командных оболочек,

которые доступны на вашем компьютере, можно посмотреть в файле `/etc/shells`.

4.2. Переменные окружения

Environment variables, или переменные окружения – это поименованные переменные, которые используются операционной системой, программами или командными файлами-сценариями. Вы можете увидеть, какие переменные окружения установлены у вас и каковы их значения в данный момент с помощью команды **env**.

Установить новое значение какой-либо переменной, например `MYENV`, можно так:

setenv MYENV value

(семейство C shells), или так:

export MYENV=value

(семейство Bourne shells).

При обращении (но не при установке!) к переменной окружения, к ее имени необходимо добавить символ "$". Например, (предполагая C shell):

setenv MYENV filename1
cat $MYENV
setenv MYENV filename2
cat $MYENV

В результате сначала на экран будет выведено содержимое тектового файла `filename1`, а затем – файла `filename2`.

Увидеть текущее значение переменной `MYENV` можно с помощью команды **echo**:

echo $MYENV

Устанавливать значение переменных окружения можно как в интерактивном режиме, т.е. через командную строку, так и внутри файлов-сценариев. Такой файл, допустим, с именем `myscript`, создается с помощью любого текстового редактора и в простейшем случае может содержать в себе одну команду, например:

setenv MYENV value

Файл-сценарий должен быть исполнен особым образом:

source myscript

В этом случае переменные окружения, заданные внутри командного файла, будут переданы в родительский процесс – оболочку, из которой этот файл был запущен – и смогут там в дальнейшем быть использованы. Существуют специальные файлы-сценарии автозагрузки, которые автоматически выполняются, когда пользователь входит в систему, и в которых обычно устанавливают значения основных переменных. Подробнее о сценариях автозагрузки будет рассказано в разделе 4.4.

Рассмотрим подробнее смысл некоторых переменных окружения.

Переменная `HOME` указывает на корневой (домашний) каталог пользователя. Предположим для примера, что вы пишите сценарий, который должен создать в домашнем каталоге любого пользователя, выполняющего этот сценарий, новый каталог с именем `newdir`, чтобы установить в нем какой-то пакет программ. Начальные строки этого файла-сценария могут выглядеть так:

```
cd $HOME
mkdir newdir
```

Тогда любой пользователь, запустивший такой сценарий из любого текущего каталога, получит желаемый результат.

Переменная `PATH` содержит в себе список каталогов, в которых система ищет команды или программы для исполнения. Другими словами, если пользователь хочет выполнить какую-либо команду (или запустить программу), то он либо должен указать полный путь к команде, либо этот путь должен содержаться в списке переменной `PATH`.

Рассмотрим следующий пример (здесь часть первой строки до команды **echo** представляет собой приглашение, выдаваемое системой, а вторая строка есть отклик системы на команду **echo $PATH** , т.е. показывает значение переменной `PATH`):

[tikhomir@lxplus050] ~/public/strtmc % *echo $PATH*
`.:/home/tikhomir/bin:/usr/bin:/bin:/usr/X11R6/bin`

Как можно видеть, каталоги в списке переменной `PATH` разделены знаком ":". В частности, каталог `/bin` содержит многие из рассмотренных ранее команд системы: **cp**, **gzip**, **mkdir** и другие. Поэтому мы и можем вызывать такие команды напрямую, просто по имени команд без указания пути к ним.

Поиск команд или программ ведется по списку каталогов в переменной `PATH` слева направо – об этом необходимо помнить, поскольку в системе и у пользователя могут оказаться команды или программы

с одинаковыми именами. В этом случае будет выполнена команда из того каталога, который расположен в списке раньше.

Отметим, что текущий каталог, представленный знаком "." в первой позиции списка в данном примере, по умолчанию может и не включаться системными сценариями автозагрузки в состав переменной `PATH`. Если это так, то вы не сможете запустить, скажем, программу **myprog**, расположенную в текущем каталоге, просто набрав в командной строке *myprog*, а должны вызвать ее так:

./myprog

Другая возможность – включить текущий каталог в список переменной `PATH`.

Чтобы добавить какой-либо каталог, скажем `/opt/bin`, в список переменной `PATH`, необходимо выполнить команду

setenv PATH ${PATH}:/opt/bin

(семейство C shells), или

export PATH=${PATH}:/opt/bin

(семейство Bourne shells).

В отсутствии пути к какой-либо программе в переменной `PATH`, как правило, кроется ответ на часто задаваемый вопрос: "Почему у другого пользователя программа или команда находится и запускается, а у меня (на том же компьютере) – нет?". Скорее всего, путь к этой программе у другого пользователя прописан в переменной `PATH`, а у вас – нет.

Значение переменной `PRINTER` указывает на имя принтера, используемого по умолчанию.

Переменная `PWD` указывает на текущий каталог.

Вид приглашения (prompt), которое выводится системой в начале командной строки терминала, получается заданием специальных последовательностей символов для переменной `PS1` (Bourne shells) или в команде **set prompt** (C shells). Например, команда

set prompt='%B[%n@%m] %b %~ % '

задает приглашение в следующем виде: внутри квадратных скобок жирным шрифтом (часть, ограниченная знаками "`%B`" и "`%b`") печатается имя пользователя (`%n`), потом знак "@" и имя компьютера (`%m`). Затем, после одного пробела, печатается имя текущего каталога, начиная от корневого каталога пользователя (`%~`), еще один пробел, знак "`%`" и снова пробел. Если пользователь работает в оболочке семейства

Bourne shells, то аналогичного эффекта он может добиться такой командой:

export PS1='[\u@\h] \W \$ '

Результат можно видеть в первой строке приведенного ранее в этом разделе примера с командой **echo $PATH**. Отметим, что принято завершать строку приглашения знаком "%" при работе в оболочках семейства C, и знаком "$" – при работе в оболочках Bourne. Это позволяет сразу, взглянув на строку приглашения, понять – в каком семействе оболочек работает пользователь.

Переменная `EDITOR` указывает на редактор, который будет использоваться по умолчанию некоторыми программами и утилитами.

Переменная `TERM` показывает тип используемого терминала. Обычно эта переменная имеет значение `xterm` при работе в графическом режиме или `linux` или `vt100` в текстовом режиме работы. Переменная `DISPLAY` указывает на адрес графического дисплея. Она важна, когда необходимо открыть новое X окно. При работе на локальном компьютере никаких проблем с использованием переменных `TERM` и `DISPLAY`, как правило, не возникает. Однако они могут появиться при работе на удаленном компьютере. О том, как правильно установить переменную `DISPLAY`, рассказано в разделе 4.5.

Как уже говорилось, значения большинства переменных окружения обычно устанавливают в системных сценариях автозагрузки или сценариях автозагрузки пользователя (раздел 4.4). Более подробно о назначении различных переменных окружения (как и о многом другом) можно прочесть в справочных страницах, посвященных соответствующим командным оболочкам: *man bash*, *man tcsh* или *man zsh*.

4.3. Псевдонимы

Для упрощения ввода часто используемых команд пользователь может создавать короткие псевдонимы (aliases). Команда **alias** имеет следующий формат:

alias newcommand 'oldcommand'

(семейство C shells), или

alias newcommand='oldcommand'

(семейство Bourne shells). Например, вот так в C shell можно задать сокращение для команды **ls –Alt – – color | more**, которая постранично

выдает на экран подробный листинг файлов из текущего каталога, отсортированных по времени последней модификации, выделяя разные типы файлов различным цветом:

alias cl '/bin/ls –Alt --color | more'

Теперь вместо того, чтобы набирать в командной строке длинную команду, пользователь может вводить ее короткий псевдоним **cl**. А

alias rm '/bin/rm –i'

переименует существующую команду **rm** так, что у пользователя будет требоваться подтверждение каждый раз, когда он захочет удалить файл.

Увидеть все действующие для данного пользователя псевдонимы можно, выполнив команду **alias** без параметров. Команда **unalias** позволит отменить существующий псевдоним, если это необходимо.

Как правило, псевдонимы устанавливаются при входе в систему в сценариях автозагрузки. В разделе 4.4 приведено несколько примеров использования псевдонимов.

4.4. Сценарии автозагрузки

Сценарии автозагрузки (startup files или profiles) – это командные файлы, которые автоматически выполняются каждый раз, когда пользователь входит в систему, запускает новую командную оболочку, открывает новый терминал или выполняет команду **su** (см. раздел 5). Существуют системные сценарии, которые выполняются для всех пользователей и личные сценарии автозагрузки каждого из пользователей. Личные сценарии автозагрузки должны быть расположены в домашних каталогах пользователей. Имена файлов автозагрузки предопределены и различны для разных командных оболочек. Кроме того, разные сценарии автозагрузки исполняются в разных ситуациях: одни – когда пользователь входит в систему, другие – когда запускает новую оболочку и т.д. Более детальную информацию можно получить из справочных или HOWTO страниц соответствующих командных оболочек. Здесь мы отметим лишь несколько пользовательских файлов автозагрузки для наиболее распространенных оболочек: `bash`, `tcsh` и `zsh`.

- Bash: `$HOME/.bash_profile` (или `$HOME/.profile`) и `$HOME/.bashrc` файлы.

- Tcsh: `$HOME/.login` и `$HOME/.tcshrc` (или `$HOME/.cshrc`) файлы.

- Zsh: `$HOME/.zprofile`, `$HOME/.zlogin` и `$HOME/.zshrc` файлы.

В файлы `$HOME/.bash_profile`, `$HOME/.profile`, `$HOME/.login`, `$HOME/.zprofile` и `$HOME/.zlogin` обычно включают команды, которые необходимо выполнить лишь один раз в момент начала сессии: задают значения переменных окружения, параметры используемого терминала и т.п. В остальных файлах задают переменные, которые должны передаваться в каждый вновь создаваемый процесс, в частности, aliases. Рассмотрим в качестве примера сценарий `.tcshrc`. Здесь строки, начинающиеся со знака "#", представляют собой комментарий, а строка, заканчивающаяся знаком "\", продолжается на следующей строке.

```
# Default editor
setenv EDITOR emacs
# Command prompt
set prompt='%{\033]0;%n@%m:%~\007%}%B[%n@%m]%b %~ % '
# Pager for man command
setenv PAGER less
#Command list for ambiguous completion (then press <Tab>):
set autolist
# My favorite fonts
setenv myfont \
 '-misc-fixed-medium-r-normal--14-110-100-100-c-70-iso8859-1'
# Set KOI-8 fonts
setenv koi8 \
 '-misc-fixed-medium-r-normal--14-130-75-75-c-70-koi8-r'
# Define emacs windows
alias em 'emacs -fn $myfont -geometry 80x39+590+0 &'
alias emkoi 'emacs -fn $koi8 -geometry 80x39+590+0 &'
# Define different xtrem windows
alias xt 'xterm -fn $myfont -geom 80x40+0+0 &'
alias xtkoi 'xterm -fn $koi8 &'
# Prevent accidental file removing
alias mv '/bin/mv -i'
alias cp '/bin/cp -i'
alias rm '/bin/rm -i'
# Goto other computers
```

```
alias mycomp 'ssh mycomp.gdeto.ru'
alias xmycomp 'xterm -T Mycomp -e ssh mycomp.gdeto.ru &'
# Colors for ls --color command
setenv LS_COLORS 'di=34;1;4:ex=31;1;4:ln=32;4'
# Aliases for some ls commands
alias ll '/bin/ls -al'
alias lt '/bin/ls -alt'
alias lh '/bin/ls -alt | head'
alias l '/bin/ls -alt | more'
alias cll '/bin/ls -al --color'
alias clt '/bin/ls -alt --color'
alias cl '/bin/ls -alt --color | more'
# Add /cern/pro/bin to PATH
setenv PATH ${PATH}:/cern/pro/bin
```

Здесь команда

`setenv EDITOR emacs`

задает редактор (**emacs**), используемый по умолчанию некоторыми программами и утилитами. Команда

`set prompt= '%{\033]0;%n@%m:%~\007%}%B[%n@%m]%b %~ % '`

устанавливает вид приглашения в командной строке (здесь использована несколько более сложная команда, чем в примере, приведенном в разделе 4.2 – она динамически меняет приглашение еще и в заголовке окна X терминала).

`setenv PAGER less`

назначает программу **less** для просмотра справочных страниц с помощью команды **man**.

`set autolist`

устанавливает способ дополнения команд и имен файлов в командной строке при использовании клавиши <Tab> (раздел 1.2): если дополнение является неоднозначным в данном контексте, будут подсказаны все возможные варианты.

Строки `setenv myfont ...` и `setenv koi8 ...` назначают переменным окружения `myfont` и `koi8` определенные шрифты. Затем создаются псевдонимы **em** и **emkoi**, запускающие редактор **emacs** с определенными параметрами окна (опция `-geometry`) и использующие только что заданные латинский (`$myfont`) и русский KOI-8 (`$koi8`) шрифты соответственно:

```
alias em 'emacs -fn $myfont -geometry 80x39+590+0 &'
alias emkoi 'emacs -fn $koi8 -geometry 80x39+590+0 &'
```

Аналогичным образом вводятся псевдонимы **xt** и **xtkoi**, открывающие окно X терминала.

Строка

```
alias mv '/bin/mv -i'
```

и две другие, следующие за ней, переобозначают команды **mv, cp** и **rm** таким образом, чтобы перед тем, как удалить любой файл, у пользователя запрашивалось подтверждение.

Команды

```
alias mycomp 'ssh mycomp.gdeto.ru'
alias xmycomp 'xterm -T Mycomp -e ssh mycomp.gdeto.ru &'
```

создают псевдонимы для входа через SSH протокол на удаленный компьютер `mycomp.gdeto.ru`. Первая из команд – **mycomp** – устанавливает связь в том же окне терминала, где она выполняется, а другая – **xmycomp** – открывает новое X окно и делает слово `Mycomp` заголовком этого окна.

Строка

```
setenv LSCOLORS 'di=34;1;4:ex=31;1;4:ln=32;4'
```

задает значение переменной окружения `LSCOLORS`, которое позволяет команде **ls** – – **color** выделять в листинге разным цветом разные типы файлов: синим цветом – каталоги, красным – исполняемые файлы, зеленым – ссылки. Далее даны примеры назначения псевдонимов для некоторых наиболее часто используемых вариантов команды **ls**.

Наконец, строка

```
setenv PATH ${PATH}:/cern/pro/bin
```

добавляет в список переменной окружения `PATH` путь к исполняемым файлам, расположенных в каталоге `/cern/pro/bin`.

Сделаем здесь небольшое отступление, чтобы отметить ряд моментов, связанных с применением в командной строке или в файле-сценарии различных скобок и кавычек. Фигурные скобки в команде `setenv PATH...` выделяют переменную `PATH` – в противном случае было бы не вполне ясно, к чему именно относится знак "$", т.е. где в строке заканчивается имя переменной окружения.

Как уже говорилось ранее, несколько команд можно объединить в группу, заключив их в круглые скобки. Выполните следующие команды и посмотрите, как в результате будет отличаться файл `out.txt`:

pwd; ls –l > out.txt

и

(pwd; ls –l) > out.txt

Применение одинарных и двойных кавычек обычно имеет одинаковый эффект. Например, обе команды

echo 'Hello, World!'

и

echo "Hello, World!"

просто выведут на экран текст, заключенный в кавычках. Также не различается их применение в команде **alias**. Однако, если в кавычках заключена переменная окружения, то их значение становится различным. Одинарные кавычки оставляют заключенный в них текст в качестве текстовой константы. Сравните:

echo '$HOME'

выведет на экран просто текст: `$HOME`. А вот

echo "$HOME"

так же, как и

echo $HOME

выдаст имя корневого каталога пользователя.

Совсем другой смысл имеют обратные кавычки. Они означают, что выражение, заключенное внутри них, должно быть выполнено раньше остальной части командной строки. Рассмотрим такой пример:

ls –l `which latex`

Как мы знаем, команда **which** находит и выводит на экран полный путь к команде или программе, указанной в качестве ее аргумента. Выполненная сама по себе, команда ***which latex*** выдала бы на экран такой путь:

```
/usr/bin/latex
```

Но, поскольку команда **which** заключена в обратные кавычки, то результат ее работы не будет выведен на экран, а будет подставлен в качестве параметра в командную строку. Таким образом командная строка преобразуется к следующему виду:

ls –l /usr/bin/latex

т.е. на экран будет выведен листинг:

```
-rwxrwxrwx  1  root root 3  Jun 2  2006 /usr/bin/latex -> tex
```

4.5. Настройка X окружения

Настройка графической среды пользователя – довольно большая тема, тем более что пользователь может работать в разных графических средах и настройки каждой из них будут отличаться. Здесь мы лишь кратко коснемся некоторых вопросов настройки X окружения.

Такие команды, как **xterm** или **emacs**, открывающие новые X окна, имеют ряд одинаковых опций, которые можно указывать при вызове. Например,

xterm –geometry 80x40+490+0 `&`

задает геометрию открываемого окна терминала xterm: размер рабочего поля терминала составит 80 символов по горизонтали и 40 символов по вертикали, а левый верхний угол окна будет расположен на расстоянии 490 пикселей по горизонтали и 0 пикселей по вертикали от левого верхнего угла экрана. Аналогично можно задать размеры и положение окна редактора **emacs**.

Другой пример:

emacs –bg lightyellow –fg black `&`

Таким образом можно задать цвет фона (опция –**bg**) и цвет шрифта (опция –**fg**) открываемого окна. Цвета задаются либо с помощью буквенно-цифрового кода (см. *man X* или *man XFree86*), либо по имени, как в приведенном примере. Имена цветов, которые можно применять, смотрите в файле `rgb.txt`, который обычно расположен в каталоге `/usr/X11R6/lib/X11`. Если на вашем компьютере установлена графическая оболочка GNOME, то увидеть палитру поименованных цветов вместе с соответствующими RGB кодами можно с помощью команды **gcolorsel**.

Еще одна важная опция (–**fn**) позволяет задать шрифт, который будет использоваться в окне. Имя шрифта может быть указано либо явным образом, например

xterm –fn \
'-misc-fixed-medium-r-normal–14-110-100-100-c-70-iso8859-1' `&`

либо через заданную заранее переменную окружения, как это было показано на примере файла `.tcshrc` в разделе 4.4:

```
setenv myfont \
 '-misc-fixed-medium-r-normal--14-110-100-100-c-70-iso8859-1'
alias em 'emacs -fn $myfont -geometry 80x39+590+0 &'
```

Список всех доступных в системе X шрифтов можно вывести на экран, выполнив команду **xlsfonts**. Программа **xfontsel** позволит рассмотреть, как выглядит тот или иной шрифт. Выбрав подходящий шрифт, нажмите мышкой клавишу `select`. Тогда имя шрифта окажется в буфере обмена и его можно будет перенести в командную строку терминала или в окно редактора. Имена некоторых шрифтов имеют сокращенные псевдонимы (aliases), например `6x10`, `9x15`, `7x13bold` и т.п. Имена таких псевдонимов хранятся в файлах `/usr/X11R6/lib/X11/fonts/<fontdir>/fonts.alias`, где `<fontdir>` есть имя каталога со шрифтами определенного семейства.

Важной опцией команды **xterm** является опция **–e**: после нее следует имя команды (возможно, с параметрами) которая должны быть выполнена в открываемом окне терминала. Например,

xterm –e mc `&`

откроет новый X терминал и запустит в нем команду **mc**. Отметим, что опция **–e** должна быть последней в командной строке.

Опция **–T** задает текст заголовка X терминала, что может быть полезно, например, когда вы со своего компьютера входите на удаленный: в качестве заголовка можно указать имя компьютера. Так, команда

xterm –T Mycomp –e ssh mycomp.gdeto.ru `&`

открывает окно X терминала для входа через SSH протокол на удаленный компьютер `mycomp.gdeto.ru` и делает слово `Mycomp` заголовком этого окна.

Многие настройки уже открытого X терминала можно изменить, если навести курсор мыши на поле терминала и нажать одновременно клавишу `<Ctrl>` и одну из кнопок мыши. Обратим внимание на две возможности. Иногда бывает, что терминал переключается в режим “абракадабры”, когда “портится” выводимый на экран текст. Такое может произойти иногда, если пользователь по ошибке выводит на экран командой типа **more** какой-нибудь двоичный файл или при сбое какой-либо программы, выводящей на экран текст. Тогда нужно нажать `<Ctrl>` и среднюю клавишу мыши и выбрать в появившемся меню `Do Soft Reset`. Другая полезная функция – сменить размер шрифта в уже открытом терминале – появится при нажатии `<Ctrl>` и правой клавиши мыши.

Многие параметры различных X приложений можно указать в файле `.Xdefaults`, расположенном в корневом каталоге пользователя. Пре-

дупреждение: прежде, чем редактировать свой файл, сохраните копию! Ниже приведен фрагмент файла `.Xdefaults` (знак "!" здесь означает строку-комментарий):

```
! emacs
emacs*Background: lightyellow
emacs*Foreground: black
emacs*pointerColor: red3
emacs*cursorColor: Orchid
emacs*font: fixed
emacs.geometry: 80x32
! xterm (and friends)
XTerm*highlightSelection:  true
! Uncomment this to use color for the bold attribute
XTerm*VT100*colorBDMode:  on
XTerm*VT100*colorBD:  blue
! Uncomment this to display the scrollbar
XTerm*scrollBar:  true
! Number of lines of scrollback to save
XTerm*saveLines:        50000
XTerm*background:       lightyellow
XTerm*foreground:       black
XTerm*cursorColor:      red3
XTerm*pointerColor:     red
! Activate PgUp & PgDn keys
xterm*VT100.Translations: #override\n\
    <KeyPress>Prior : scroll-back(1,page)\n\
    <KeyPress>Next : scroll-forw(1,page)
```

Большинство строк здесь не нуждаются в комментариях: они задают цвета, размеры окна и семейство шрифтов, которые будут использоваться командами **xterm** и **emacs** по умолчанию. Отметим только два момента. Строка

```
XTerm*saveLines: 50000
```

задает количество строк в окне X терминала, которые запоминаются в специальном буфере и могут быть просмотрены, если двигать ползунок на линейке прокрутки нажатой средней кнопкой мыши или с помощью клавиш <PgUp> и <PgDn> на клавиатуре. Строки

```
xterm*VT100.Translations: #override\n\
```

```
<KeyPress>Prior : scroll-back(1,page)\n\
<KeyPress>Next : scroll-forw(1,page)
```

как раз и активируют клавиши <PgUp> и <PgDn> для прокрутки. Эти клавиши можно использовать и при прокрутке других окон, скажем, в **emacs** или в **mozilla**. Если прокрутка окон с помощью клавиш <PgUp> и <PgDn> у вас не работает, попробуйте нажать клавишу <Shift> и, не отпуская ее, нажать <PgUp> или <PgDn>.

Такого рода параметры (они называются X ресурсы) можно указывать в файле .Xdefaults и для ряда других программ, таких, например, как **xman** или **gv**. Имена и значения используемых X ресурсов указаны в соответствующих справочных страницах программ. Значения X ресурсов, принятые различными программами по умолчанию, содержатся в файлах каталога /usr/X11R6/lib/X11/app-defaults. В разделах 9 и 10.1 будут приведены еще примеры использования файла .Xdefaults для задания X ресурсов программ **xterm** и **gv**.

5. Процессы, задания, пользователи

Каждое задание, выполняемое на компьютере под управлением операционной системы Linux, называется процессом. Часть процессов запускается при загрузке системы. Они управляют работой некоторых устройств компьютера, обслуживают различные запросы в системе. Выполняются эти процессы в фоновом режиме и обычный пользователь может и не подозревать об их существовании.

Другие процессы запускаются пользователями – сначала при входе в систему, затем при запуске команд или программ. Например, открывая новый X терминал, пользователь запускает новый процесс. Если из этого терминала вызывается, скажем, программа **emacs**, то создается новый процесс, дочерний по отношению к тому, в котором он был создан, и т.д.

Список процессов можно вывести на экран командой **ps**. Введенная без параметров, эта команда выдаст в коротком формате список процессов, которые были запущены из данного терминала. Чтобы вывести на экран в подробном формате список всех процессов, созданных пользователем, например, tikhomir, наберите:

ps a –u tikhomir

Вы увидите что-то в таком роде:

```
 PID TTY    STAT TIME COMMAND
6675 ?      S    0:00 -/bin/tcsh -c /usr/share/apps/switchdes
6826 ?      R    0:17 xterm -bg lightyellow -geom 80x40+0+0 -
6828 pts/1  S    0:00 -csh
6845 pts/1  S    0:01 /usr/X11R6/bin/xterm -bg LightYellow -f
6847 pts/2  S    0:00 ssh lxplus.cern.ch
7043 pts/1  S    0:25 /usr/bin/emacs -bg LightYellow -cr Orch
7084 pts/1  S    0:00 gv posobie.ps
7174 ?      S    0:01 xterm -bg lightyellow
7176 pts/3  S    0:00 -csh
7230 pts/1  S    0:05 gs -dNOPLATFONTS -sDEVICE=x11alpha -dNO
7271 ?      S    2:55 /usr/lib/mozilla-seamonkey-1.0.9/mozill
7305 ?      S    0:00 (dns helper)
7307 pts/1  R    0:00 ps a -u tikhomir
```

В первой колонке указан PID (process identifier) – уникальный номер процесса в системе. Этот номер необходимо указывать при обращении к процессу, например, чтобы прервать его, если он “повис” и вы не можете прекратить его стандартными средствами. Так, чтобы прервать программу **mozilla**, которая в предыдущем примере имеет номер 7271, наберите команду:

kill 7271

Более сильная команда, которую следует применять, если “мягкий” вариант **kill** не дает желаемого эффекта:

kill –9 7271

Такая команда безусловно прервет процесс, хотя и не гарантирует, что прерывание пройдет корректно и не оставит за собой других незавершенных процессов, незакрытых файлов и т.п.

Команда **top** показывает периодически обновляющийся список процессов, занимающих наибольшее процессорное время на данном компьютере. На рис. 4 представлен пример работы команды **top**. Здесь для каждого процесса можно видеть: PID; имя пользователя, запустившего процесс; приоритет задачи (чем меньше число, тем выше приоритет); размер памяти, отведенный под данный процесс; процент загрузки CPU (в приведенном рисунке суммарная загрузка превышает 100%, поскольку примером послужил двухпроцессорный компьютер) и памяти; общее время, набранное задачей к настоящему моменту. Состояние процесса указано в столбце `STAT`: `R` означает активный (running) процесс, `S` (sleeping) – процесс, находящийся в состоянии “спячки” и гото-

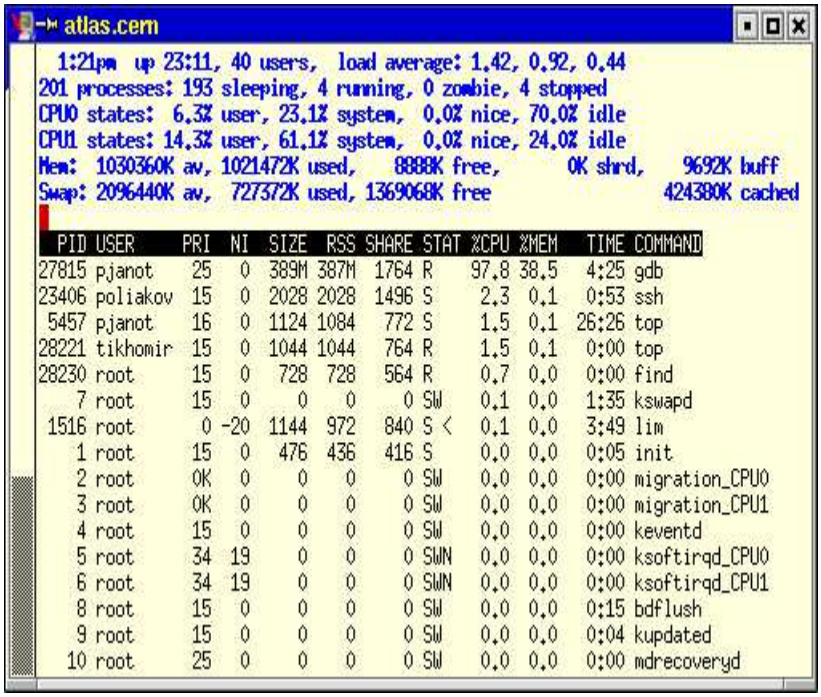

Рис. 4. *Пример работы программы* ***top***

вый заработать при наступлении какого-то события в системе (например, если на печать направлено задание). Кроме того, в верхней части экрана показано: сколько времени работает данный компьютер без перезагрузки, сколько пользователей на нем работает, как загружен CPU и память и др. Данная информация может быть полезна, например, если вы работаете на кластере из нескольких компьютеров и хотите выбрать наименее загруженный из них. Кроме того, если вы запустили программу, например, на языке C, можно контролировать – не создает ли она утечку памяти (memory leakage): память, занимаемая программой, указана в столбце `SIZE`.

Каждый процесс может выполняться либо в прямом (foreground), либо в фоновом (background) режиме. В прямом режиме родительский процесс будет ожидать окончания работы запущенного из него дочернего процесса, в фоновом – родительский и дочерний процессы

работают независимо. Если вы запустите из X терминала редактор **emacs** в прямом режиме, просто набрав в командной строке

emacs

то обнаружите, что X терминал оказывается заблокирован – вы не можете вводить в нем новые команды. Это происходит потому, что родительский процесс – X терминал (точнее, командная оболочка) – ждет окончания работы дочернего – **emacs**. Это состояние будет сохраняться до момента окончания работы дочернего процесса, то есть, в данном случае – до тех пор, пока вы не закроете окно редактора **emacs**. Чтобы запустить программу в фоновом режиме (если это имеет смысл), добавьте в конце командной строки знак `&`. Например, команда

emacs `&`

запустит редактор **emacs**, оставив терминал (`xterm`), из которого команда была выполнена, свободным для ввода других команд. В таком случае процесс `emacs` будет называться по отношению к процессу `xterm` заданием (job). Увидеть все задания, запущенные из данного терминала и не завершенные к данному моменту, можно с помощью команды **jobs**. Например, эта команда выдаст на экран:

```
[5]  + Running                 emacs
[6]  - Running                 gv linux.ps
```

Здесь в квадратных скобках указан номер задания. Знак "+" указывает на так называемое "текущее задание", к которому можно применять команду **fg** (см. ниже) без параметра.

Чтобы прервать задание, если это необходимо, наберите

kill `%`*n*

где `n` – номер задания.

В принципе, можно переводить те или иные процессы из прямого в фоновый режим и обратно: <`Ctrl-z`> "подвесит" запущенный в прямом режиме процесс, команда **bg** переведет этот "подвешенный" (suspended) процесс в фоновый режим, а **fg %n** вернет задание с номером `n` обратно в прямой режим исполнения. Однако такие возможности применяются редко, и пользоваться ими нужно с осторожностью.

С помощью команд **who** и **users** можно увидеть, кто из пользователей подключен в данный момент к системе. Команда **w** сделает то же в более подробном формате. Команда

finger username

позволит получить некоторую информацию о пользователе `username`, даже если тот не подключен в данный момент к компьютеру.

Если необходимо узнать username всех пользователей, зарегистрированных на данном компьютере, то это можно попытаться сделать с помощью команды

ls $HOME/..

т.е. распечатав имена каталогов, находящихся на одном уровне с домашним каталогом пользователя – обычно эти имена совпадают с именами пользователей. Но такой способ работает не всегда, поскольку система домашних каталогов пользователей может быть организована и другим образом.

Свой пароль пользователь может сменить командой **passwd**. Для смены пароля могут применяться и другие команды. Скажем, если способ авторизации на вашем компьютере основан на системе NIS или Kerberos, командыми для смены пароля служат соответственно **yppasswd** и **kpasswd**. Уточнить это необходимо у вашего системного администратора.

Команда **whoami** покажет – какой пользователь является хозяином терминала, с которого выполнена эта команда. Несмотря на кажущуюся абсурдность, ситуаций, когда эта команда может быть полезной, не так уж мало в условиях, когда одним компьютером пользуется несколько пользователей или, наоборот, один пользователь может входить в систему под разными username – например, системный администратор.

Иногда возникает необходимость одному пользователю войти в систему под именем другого пользователя – например, этот второй просит первого на короткое время предоставить ему терминал, чтобы отправить e-mail, а перезагружать компьютер нет смысла. Простейший способ сделать это – команда

su username

Разумеется, система попросит ввести пароль пользователя `username`. Если необходимо, чтобы при login процессе выполнились сценарии автозагрузки второго пользователя, необходимо добавить опцию –l:

su username –l

Напротив, если требуется, чтобы сохранились переменные окружения первого пользователя, нужно набрать:

su username –m

Команда **su**, введенная без параметров, предполагает, что выполнивший ее хочет войти в систему как root – пользователь с привилегиями системного администратора.

Команда **date** покажет текущее время на компьютере. Команда

uname –a

выведет краткую информацию о данном компьютере, в частности – о версии ядра (kernel) установленной операционной системы. Различную информацию о компьютере и используемых ресурсах можно также получить с помощью команды **procinfo** или просмотрев содержимое файлов `/proc/cpuinfo` и `/proc/meminfo`.

6. Печать в Linux

Печать из командной строки Linux осуществляется командой **lpr**:

lpr `[`*–P printername*`]` *filename*

где `filename` – имя файла, направляемого на печать, а `printername` есть имя принтера. Вы можете узнать имя вашего принтера (или нескольких) у системного администратора или "подсмотреть" в файле `/etc/printcap`. Принтер может быть как локальным, подключенным непосредственно к вашему компьютеру, так и сетевым. Печать из Linux возможна и на принтер, подключенный к другому компьютеру под управлением Windows, если на Linux установлен пакет Samba.

Если на вашей системе установлен только один принтер, то его имя в команде **lpr** и других, приведенных ниже, можно не указывать. Если принтеров несколько, но вы пользуетесь преимущественно одним из них, то чтобы не набирать каждый раз имя принтера, можно задать его через переменную `PRINTER`:

setenv PRINTER printername

(для семейства C shells) или:

export PRINTER=printername

(для семейства Bourne shells).

Исторически сложилось так, что в Linux одним из наиболее часто используемых форматов файлов, предназначенных для печати, является формат PostScript. PostScript файлы могут быть направлены на принтер непосредственно командой **lpr**, если сам принтер поддерживает этот формат, либо через программу **gv** (раздел 10.1). Во втором случае можно предварительно просмотреть PostScript файл, выбрать

для печати отдельные страницы. Многие программы в Linux либо прямо конвертируют свой вывод в PostScript файл, либо используют при выводе на принтер программу **ghostscript** (часть программы **gv**) в качестве фильтра.

Для печати больших текстовых файлов иногда удобно пользоваться программой **a2ps**, которая переводит текст в формат PostScript, форматируя его по несколько (по умолчанию – по две) страниц на лист и печатая красивую рамку, в которой указаны время печати, имя файла, номер страницы. Вы можете либо сначала создать PostScript файл:

a2ps filename -o filename.ps

который потом можно просмотреть и распечатать, либо сразу направить текст на печать, без создания промежуточного файла:

a2ps filename

Похожими функциями обладает программа **mpage**. Она также умеет форматировать входной файл в несколько страниц на лист, причем входным файлом может служить не только текстовый, но и PostScript файл. Рекомендуемые опции для программы **mpage**:

mpage –2 –bA4 –S –l1 –H –f

Команда

lpq [*–P printername*]

покажет состояние очереди на печать на принтер `printername`.

Команда

lprm [*–P printername*] *n*

удаляет из очереди задание с номером `n`, который указан в выводе команда **lpq**. Следует помнить, что в Linux задание на печать хранится в специальном файле, и вы не сможете удалить это задание, просто выключив принтер или даже выключив компьютер. Как только компьютер снова будет включен, система обнаружит незавершенное задание и начнет выполнять его. Поэтому для удаления задания пользуйтесь командой **lprm**.

Если на вашем компьютере установлена графическая оболочка KDE или GNOME, то в них печать осуществляется просто нажатием мышкой на соответствующую иконку или через меню `File→Print` в окне запущенного приложения.

7. Разработка программ

В этом разделе мы коснемся вопросов разработки собственных программ пользователя и работы с этими программами. Рассмотрены будут программы и библиотеки, созданные с помощью алгоритмических языков программирования типа C или Fortran. Мы коснемся также вопроса о написании сценариев оболочки.

7.1. Компиляция программ

Существует множество компиляторов с различных алгоритмических языков для Linux. Рассмотрим некоторые вопросы трансляции и компиляции программ, написанных на языках C и Fortran.

Команды для трансляции и компиляции программ на языке C и C++ называются **gcc** и **g++** соответственно. Предполагается, что файлы с текстами программ на языке C имеют расширение `.c`, а файлы с текстами программ на языке C++ – `.C`, `.cc` или `.cxx`. Простейшая команда для трансляции и компиляции, например, программы `hello.c`:

g++ hello.c

По умолчанию после успешной компиляции создается файл с именем `a.out`, который можно запустить (выполнить), набрав в командной строке:

./a.out

или просто

a.out

если текущий каталог входит в список переменной окружения PATH. Для того, чтобы дать исполняемому файлу другое имя, например, `hello`, используйте опцию **–o**:

g++ hello.c –o hello

Если результатом трансляции должен стать двоичный файл, а не исполняемый (т.е. необходимо опустить стадию компиляции), применяется опция **–c**:

g++ –c hello.c

В этом случае создастся двоичный файл `hello.o`. Собрать несколько файлов в один исполняемый можно, перечислив их в команде:

g++ main.c hello.o –o main

Компилятор с языка Fortran называется **g77**. Файлы с текстами программ по умолчанию имеют расширения `.f` или `.f77`. Большинство опций команды **gcc** справедливы и для **g77**.

Приведенные здесь команды трансляции и компиляции имеют огромное число опций. Отметим некоторые из них.

Опция **–O** задает уровень оптимизации кода при трансляции. По умолчанию оптимизация не проводится, вместо этого минимизируется время трансляции. Опция **–O** эквивалентна **–O1** и означает минимальный уровень оптимизации. Более высокие уровни задаются опциями **–O2** и **–O3** и должны, в принципе, приводить к более быстрому выполнению программы. С другой стороны, более высокий уровень оптимизации увеличивает время трансляции программы и несет в себе ряд ограничений и потенциальных опасностей. Поэтому обычно при разработке больших и сложных программ применяется следующий подход. В стадии разработки и отладки программа компилируется без оптимизации или с минимальным уровнем оптимизации. И лишь при компиляции окончательного варианта программы включатся желаемый уровень оптимизации. При этом проверяется, что: а) результат работы оптимизированной программы совпадает с результатом той версии программы, которая получена в режиме без оптимизации; б) оптимизированная программа действительно работает существенно быстрее.

Опция **–W** задает "уровень предупредительности" компилятора, т.е. насколько подробно он будет выводить на экран warnings – предупреждения, не являющиеся фатальными ошибками. В частности, опция **–w** подавляет предупреждения, **–W** задает средний уровень предупредительности, а **–Wall** – высокий (но не максимальный) уровень.

Опция **–I**, за которой следует имя каталога, указывает на каталоги, в которых содержатся include файлы.

Опция **–L**, за которой следует имя каталога, указывает на каталоги, в которых содержатся библиотечные файлы, необходимые при компиляции главной программы. Сами библиотеки затем указываются через опцию **–l**. Опция **–l** предполагает следующий стандарт имен библиотечных файлов: имя начинается с приставки `lib`, а расширение файла – `.a`. Рассмотрим такой пример:

g77 mygraf.f –W –O –o mygraf –L/cern/pro/lib –lgraflib \
–L/usr/X11R6/lib –lX11 –lXt

Здесь транслируется программа из файла `mygraf.f` и к ней подключаются библиотека `libgraflib.a` из каталога `/cern/pro/lib`, а также

библиотеки `libX11.a` и `libXt.a` из каталога `/usr/X11R6/lib`. Если имена библиотечных файлов не удовлетворяют приведенному выше стандарту, то такие библиотеки могут включаться в командную строку явным образом:

g77 mygraf.f /home/tikhomir/lib/graflib.a –W –O –o mygraf

Такая команда подсоединит к транслируемой программе из файла `mygraf.f` библиотеку с нестандартным именем `graflib.a` из каталога `/home/tikhomir/lib/`.

Пути к каталогам `/lib` и `/usr/lib` включены в команды компиляции по умолчанию. Таким образом, ко всем библиотекам, например, в каталоге `/usr/lib` можно обращаться без указания пути к ним через опцию **–L**:

g77 mygraf.f –W –O –o mygraf –lcrypt –ldl

Поскольку путь здесь явным образом не указан, то библиотеки с именами `libcrypt.a` и `libdl.a` будут искаться и загружаться из каталога `/usr/lib`. Отметим, что порядок перечисления библиотек в командах компиляции является существенным (см. раздел 7.2).

Еще одной важной опцией компиляторов является опция **–g**, которая делает код пригодным для использования с интерактивным отладчиком – программой **gdb**. Код, сгенерированный с использованием опции **–g**, занимает несколько больше места, чем без нее. Еще важнее то, что эта опция может находиться в конфликте с опцией оптимизации кода **–O**. Поэтому опцией **–g** рекомендуется пользоваться только в процессе написания и отладки программ. После того, как программа отлажена, эту опцию обычно убирают, а вместо нее ставят опцию оптимизации. В разделе 7.3 рассказано, как пользоваться отладчиком **gdb**. А в разделе 7.4 приведен пример простого сценария для компиляции программ пользователя.

До сих пор рассматривались только статические библиотеки, или архивы (static libraries). В разделе 7.2 будет рассказано об использовании еще одного вида библиотек – загружаемых, или динамических (shared libraries).

7.2. Библиотеки программ

Команда **ar** позволяет создавать статические библиотеки (архивы) и проводить с ними различные операции. Формат команды:

ar option... libname `[`*objfile...*`]`

Здесь `option...` – опции команды **ar**, `libname` – имя библиотеки, а `objfile...` – имена двоичных файлов – модулей, составляющих библиотеку. Заметим, что опции команды **ar** могут указываться без предшествующего им знака "–". Опция **r** позволяет создать библиотеку или добавить новые модули к уже существующей библиотеке:

ar r libmy.a fun1.o fun2.o fun3.o

Данная команда создаст библиотечный файл `libmy.a`, если такой еще не создан, и занесет в него двоичные модули из файлов `fun1.o`, `fun2.o`, и `fun3.o`. А команда

ar t libmy.a

выдаст список двоичных модулей, содержащихся в библиотеке `libmy.a`. Опция **x** позволит "извлечь" модуль из библиотеки и сохранить его в виде отдельного файла:

ar x /cern/pro/lib/libkernlib.a umcom.o

сохранит модуль `umcom` из библиотеки `/cern/pro/lib/libkernlib.a` как двоичный файл с именем `umcom.o` в текущем каталоге.

Иногда при компиляции программы вы можете получить сообщение типа "`undefined reference to lenocc.o`", означающее, что не найден модуль `lenocc.o`, к которому есть обращение от одной из транслируемых подпрограмм. Возможно несколько причин, по которым модуль не был найден. Во-первых, есть вероятность, что `lenocc.o` содержится в библиотеке, которая не перечислена в команде компиляции. Тогда необходимо найти в файловой системе библиотеку, содержащую этот модуль и включить ее в команду компиляции. В разделе 7.4 приведен пример простого сценария, позволяющего быстро находить желаемую подпрограмму в библиотечных файлах.

Во-вторых, напомним, что порядок, в котором перечислены библиотеки в команде компиляции, является существенным. Поэтому, если в команде перечислены несколько библиотек, и мы получили сообщение "`undefined reference to lenocc.o`" – это еще не значит, что данного модуля `lenocc.o` действительно нет ни в одной из перечисленных библиотек. Просто может оказаться, что порядок перечисления библиотек в команде компиляции неверный, и его необходимо изменить. Общее правило здесь такое: библиотека, в которой содержится вызываемая подпрограмма, должна стоять в команде компиляции после библиотек, содержащих модули, которые эту подпрограмму вызывают. Пусть например, в вашей программе `myprog.f` вызывается подпрограмма `subr1`,

которая содержится в библиотеке `lib1.a`. Подпрограмма `subr1` вызывает другую подпрограмму `subr2`, которая заключена в библиотеке `lib2.a`. Тогда такая команда компиляции

g77 myprog.f lib1.a lib2.a ...

сработает правильно, а вот

g77 myprog.f lib2.a lib1.a ...

выдаст сообщение об ошибке: "`undefined reference to subr2`".

Наконец, возможно, что требуемого модуля действительно нет ни в одной из библиотек в системе. Тогда, естественно, необходимо установить соответствующую библиотеку с этим модулем.

Как уже говорилось выше, помимо статических библиотек, существуют shared libraries – загружаемые или динамические библиотеки. При использовании статических библиотек подпрограммы, входящие в ее состав, подгружаются к создаваемой исполняемой программе на стадии компиляции. Подпрограммы из загружаемых библиотек не подключаются к исполняемому файлу при компиляции, но могут быть подгружены либо при запуске программы, либо уже во время ее исполнения. Это позволяет собрать относительно компактный исполняемый модуль и подключать к нему после его запуска модули из загружаемых библиотек по мере необходимости. Пусть, например, пишется программа, которая моделирует отклик какой-либо экспериментальной установки на различные физические события. Для ее работы может понадобиться целый ряд программ – генераторов различных классов таких событий. Но нет смысла подключать все эти программы к исполняемому модулю – размер последнего может оказаться очень большим. Вместо этого можно создать для каждой программы-генератора свою загружаемую библиотеку и подгружать ее только в том случае, если именно этот генератор нам необходим для конкретного расчета.

Имена файлов загружаемых библиотек имеют расширения `.sl` и `.so`. Чтобы вместо обычного двоичного файла в результате трансляции создалась загружаемая библиотека, указывается опция –**shared**:

g77 pp2g.f –W –O –shared –o pp2g.so

Имена тех загружаемых библиотек, которые необходимы при запуске данной программы, можно получить с помощью команды **ldd**:

[tikhomir@pascal] ~ % *ldd /home/tikhomir/bin/atlsim*

```
libm.so.6 => /lib/i686/libm.so.6 (0x4002c000)
```

```
libXm.so.1 => /usr/X11R6/lib/libXm.so.1 (0x4004f000)
libXt.so.6 => /usr/X11R6/lib/libXt.so.6 (0x40151000)
libXp.so.6 => /usr/X11R6/lib/libXp.so.6 (0x4019d000)
libXext.so.6 => /usr/X11R6/lib/libXext.so.6 (0x401a4000)
libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0x401b1000)
libnsl.so.1 => /lib/libnsl.so.1 (0x40286000)
libcrypt.so.1 => /lib/libcrypt.so.1 (0x4029c000)
libdl.so.2 => /lib/libdl.so.2 (0x402c9000)
libc.so.6 => /lib/i686/libc.so.6 (0x42000000)
libSM.so.6 => /usr/X11R6/lib/libSM.so.6 (0x402cc000)
libICE.so.6 => /usr/X11R6/lib/libICE.so.6 (0x402d4000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

В этом примере команда **ldd** показывает: какие загружаемые библиотеки необходимы для работы программы `atlsim` (в левой стороне строки), а затем, после знаков "=>" – найдены ли в файловой системе необходимые библиотеки и где именно. Если хотя бы одна из библиотек не найдена, мы не сможем использовать программу `atlsim`, о чем будет сообщено при ее запуске.

Для исправления такой ситуации необходимо, во-первых, либо найти соответствующую библиотеку, например, командой **find** или **locate** (раздел 2.12), либо установить эту библиотеку, если ее вообще нет на вашей системе. Во-вторых, необходимо сообщить программе-загрузчику: где та должна искать динамические библиотеки при запуске главной программы. Ситуация с поиском загружаемых библиотек в некотором смысле аналогична ситуации с поиском команд или программ при их вызове в командной строке: система должна знать, где искать эти команды или библиотеки. Если в случае команд и программ поиск ведется в тех каталогах, которые перечислены в списке переменной окружения `PATH` (см. раздел 4.2), то и поиск загружаемых библиотек будет вестись только в определенных каталогах. Во-первых, это каталоги `/lib` и `/usr/lib`. Во-вторых, это каталоги, которые перечислены в системном файле `/etc/ld.so.conf`. Например, команда

more /etc/ld.so.conf

может показать следующее:

```
/usr/kerberos/lib
/usr/X11R6/lib
/usr/lib/qt-3.1/lib
/usr/local/lib
```

Обычный пользователь не может изменить список каталогов в файле `/etc/ld.so.conf` и не может добавлять свои файлы в перечисленные выше каталоги. Поэтому, если для какой-либо программы необходимо обращение, допустим, к загружаемым библиотекам, расположенным в каталоге `/home/tikhomir/lib`, это можно сделать только включив путь `/home/tikhomir/lib/` в список специальной переменной окружения `LD_LIBRARY_PATH`:

setenv LD_LIBRARY_PATH /home/tikhomir/lib

(для семейства C shells) или:

export LD_LIBRARY_PATH=/home/tikhomir/lib

(для семейства Bourne shells). Если необходимо перечислить несколько каталогов, то они разделяются в списке символом ":", как и в переменной `PATH`. Посмотреть соответствующий список каталогов можно командой

echo $LD_LIBRARY_PATH

Итак, загружаемые библиотеки, необходимые для запуска какой-либо программы, будут искаться в каталогах `/lib`, `/usr/lib`, либо в каталогах, перечисленных в файле `/etc/ld.so.conf`, либо в каталогах, перечисленных в списке переменной окружения `LD_LIBRARY_PATH`.

Вопрос о том, какие библиотеки – статические или загружаемые – лучше использовать, зависит от конкретных обстоятельств и решается в каждом случае индивидуально. Использование загружаемых библиотек, как уже говорилось, позволяет создавать более компактные выполняемые программы. С другой стороны, программы, в которые явным образом включены модули из статических библиотек, обладают тем преимуществом, что они без проблем могут быть перенесены и исполнены на другом компьютере. Для программ, откомпилированных в расчете на последующее подключение загружаемых библиотек, необходимо, чтобы эти библиотеки присутствовали на том компьютере, куда переносится выполняемый файл.

При работе с большими программными пакетами часто используются команды **make** или **gmake**. Эти команды позволяют автоматически определять: какие части пакета должны быть перекомпилированы в случае, если в процессе развития данного пакета в коды вносятся какие-либо изменения. Инструкции для команд **make** и **gmake** прописываются в специальном файле с именем `Makefile`, где задается: в каких каталогах лежат тексты программ, в каких – include файлы, в каких – уже готовые библиотеки; в каком порядке и с какими опция-

ми программы должны компилироваться; каковы зависимости между файлами с кодами программ, include файлами и библиотеками; что должно получиться в результате – статические или загружаемые библиотеки, исполняемые файлы и т.д. Чтобы правильно "собрать" такой пакет, внимательно прочитайте содержимое файлов с именами типа `README` и `INSTALL`, в которых должна быть инструкция по установке.

7.3. Исполнение программ

После успешной компиляции программа запускается на исполнение просто указанием ее имени в командной строке. Здесь мы отметим ряд моментов, связанных с выполнением программ.

Напомним, что приостановить исполнение программы и выводимый ею на экран текст можно, нажав клавиши <`Ctrl-s`>. Чтобы продолжить работу программы и вывод на экран, нажмите <`Ctrl-q`>. Совсем прервать программу можно, нажав <`Ctrl-c`>.

При работе над каким-либо крупным проектом достаточно часто возникает ситуация, когда в системе существует несколько версий программы (исполняемого файла) с одним и тем же именем, но расположенных в разных каталогах. Например, одна старая версия была сохранена для того, чтобы было возможно воспроизвести результаты, полученные с ее помощью некоторое время назад и включенные в некий важный отчет. В другом каталоге хранится новая версия, с который мы работаем в данный момент. И еще одна, самая новая, версия программы находится в стадии разработки и со временем заменит ту, с которой мы работаем сейчас. Убедиться в том, что мы используем именно ту версию программы, которую необходимо, поможет команда **which**:

which program

покажет полный путь к программе `program`. Эта же команда может оказаться полезной и для указания пути к командам Linux: здесь также возможны случаи, когда в системе установлены, скажем, несколько версий компилятора **gcc**. Команда **which** поможет нам убедиться в том, что мы пользуемся правильной версией.

Если программа была скомпилирована с использованием опции **–g** (раздел 7.1), то при необходимости для ее отладки можно воспользоваться программой **gdb**. Запустите отладчик, набрав

gdb

Вы попадаете в оболочку программы **gdb**. Теперь нужно загрузить саму исполняемую программу:

file program

где `program` – имя исполняемого файла. Командой

run

программа будет запущена на выполнение. Если при выполнении программы происходит какая-либо фатальная ошибка, то диагностика поможет определить ее характер и местоположение в тексте программы. Вы можете выполнять программу пошагово или задавать так называемые breakpoints – точки остановки в исполняемой программе, до которых ее выполнение происходит в автоматическом режиме, а при достижении этих точек будет приостановлено и управление передано программе **gdb**. В этот момент можно проверить – чему равны значения каких-либо переменных исполняемой программы, изменить значения этих переменных, назначить новые breakpoints или отменить существующие и т.п. Все это позволяет более точно идентифицировать характер и место фатальных ошибок в исполняемой программе. Выйти из программы **gdb** можно, набрав команду *quit*.

В случае, если запускаемая программа должна проработать несколько часов или даже суток, очень полезной будет команда **at**. Эта команда позволяет запустить программу в любое заданное время в специальном пакетном (batch) режиме. При этом пользователь может и вовсе выйти из системы – программа будет запущена и выполнена без его участия. Для запуска задания с помощью команды **at** необходимо подготовить простой сценарий. Рассмотрим такой пример файла-сценария (пусть он называется, допустим, `pp2g.bat`):

```
ln -sf pp2g.inp.1 pp2g.inp
pp2g >> pp2g.out
mv pp2g.hbook pp2g.hbook.1
ln -sf pp2g.inp.2 pp2g.inp
pp2g >> pp2g.out
mv pp2g.hbook pp2g.hbook.2
ln -sf pp2g.inp.3 pp2g.inp
pp2g >> pp2g.out
mv pp2g.hbook pp2g.hbook.3
```

В этом сценарии трижды запускается программа `pp2g`. Программа должна считывать входные данные, необходимый для ее работы, из дискового файла с именем `pp2g.inp`. Для того, чтобы выполнить про-

грамму с тремя разными наборами входных параметров, заключенных в файлах pp2g.inp.1, pp2g.inp.2 и pp2g.inp.3 соответственно, каждый раз перед запуском программы создается ссылка pp2g.inp на файл с входными параметрами. Затем запускается программа pp2g и то, что она в интерактивном режиме должна была бы выводить на экран, записывается в файл pp2g.out. (Заметим, что если системный вывод при выполнении команды **at** не перенаправлен в дисковый файл, то по окончании задания соответствующий текст будет послан пользователю через e-mail). Допустим также, что по окончании своей работы программа pp2g сохраняет результаты в дисковом файле pp2g.hbook. Для того чтобы следующий запуск программы pp2g не перезаписал этот выходной файл, он переименовывается последовательно в файлы pp2g.hbook.1, pp2g.hbook.2 и pp2g.hbook.3. Запустить такой сценарий pp2g.bat можно следующим образом:

at –f pp2g.bat `time`

где `time` – желаемое время запуска. Простейшие примеры формата времени `time` таковы: ключевое слово `now` означает, что необходимо запустить сценарий немедленно, а, например, 19:40 – сегодня в 19 часов 40 минут. Формат задания времени `time` можно посмотреть в справочной странице команды **at**. Файл-сценарий pp2g.bat должен иметь атрибут доступа "x". Напомним также, что текущее время в системе выдается командой **date**.

С помощью команды **atq** можно узнать о состоянии своих заданий, направленных на исполнение командой **at**. При необходимости можно удалить задание командой **atrm**.

Если вы проводите свои расчеты на кластере из нескольких компьютеров, то там, возможно, установлена какая-либо другая batch-система, например LSF или PBS. Такие системы позволяют автоматически направлять задание на наименее загруженный компьютер в кластере.

Иногда при возникновении фатальной ошибки в исполняемой программе в текущем каталоге образуется файл с именем `core` или `core.N`, где `N` – некое целое число. Это – образ памяти компьютера, связанный с исполняемой программой в момент возникновения ошибки. Если вы не эксперт и не собираетесь анализировать данный файл, просто удалите его.

7.4. Сценарии

Сценарий (script) – это текстовый файл, который выполняется из командной строки или из другого сценария. Мы уже видели примеры сценариев в разделах 4.4 (файл `.tcshrc`) и 7.3 (файл `pp2g.bat`). Отметим здесь еще несколько моментов.

Сценарии могут быть достаточно сложными, представляя собой по существу целые программы. Внутри сценариев могут существовать циклы, арифметические, логические или текстовые операторы, вызов системных функций и т.д. Синтаксис операторов в сценариях отличается для разных оболочек. Просмотрите справочные страницы соответствующих оболочек и найдите в системе какие-нибудь готовые сценарии (например, в каталоге `/etc`), чтобы разобраться в правилах, по которым эти файлы составляются.

Выполнить сценарий можно двумя способами. Если сценарий не создает какие-либо новые переменным окружения или псевдонимы, которые должны быть переданы в родительский процесс, а представляет собой просто набор команд, как в приведенном выше примере сценария `pp2g.bat`, то тогда файлу нужно присвоить атрибут "`x`" и выполнить его как обычную команду, просто введя имя в командной строке и нажав <Enter>:

a+x pp2g.bat
./pp2g.bat

Если же в сценарии задаются переменные окружения или псевдонимы, которые должны быть переданы в родительский процесс, то такой сценарий должен быть исполнен особым образом, как это было отмечено в разделе 4.2:

source myscript

В этом случае файл `myscript` может и не иметь атрибута "`x`".

В сценарий можно передавать параметры. Рассмотрим такой пример файла-сценария с именем `flc`:

```
g77 $1.f -W -O -o $1 \
-L/cern/pro/lib -lpacklib -lmathlib -lkernlib
```

Этот сценарий предназначен для компиляции программ на языке Fortran с подключением к программе ряда библиотек. Параметром, передаваемым в сценарий, является имя файла с текстом программы (без расширения `.f`). Внутри сценария этот (первый и единственный)

параметр обозначается как $1. В сценарии этот параметр встречается дважды как аргумент команды **g77**: как имя файла с компилируемой программой (к параметру $1 здесь присоединяется расширение файла `.f`) и как имя создаваемой исполняемой программы после опции **–o**. Вторая строка файла задает необходимые для компиляции библиотеки. Теперь мы можем скомпилировать любую программу, выполнив сценарий с параметром:

flc myprog

если `myprog.f` – имя файла с текстом программы.

Приведенный пример можно немного усложнить, изменив первую строку файла `flc` следующим образом:

```
g77 $1.f $2 $3 $4 -o $1 \
```

Теперь в качестве параметров $2, $3 и $4 можно передавать опции для команды **g77**:

flc myprog –W –O2

В разделе 7.2 мы рассматривали ситуацию, когда компилятор не может найти некий модуль, расположенный в одной из библиотек, но неизвестно – в какой именно. Следующий файл-сценарий (пусть его имя будет, скажем, `slib`) поможет быстро найти библиотеку, в которой содержится искомый модуль.

```
cd $2
for i in *
  do echo $i
  ar t $i | grep -i $1 2>/dev/null
  done
```

Такой сценарий вызывается с двумя параметрами, первый из которых является именем искомого модуля, а второй – именем каталога, в котором расположены библиотеки, предположительно содержащие этот модуль. Например,

slib lenocc /cern/pro/lib

проведет поиск модуля `lenocc` во всех библиотеках, расположенных в каталоге `/cern/pro/lib`.

Синтаксис команд сценария для разных командных оболочек различен. По умолчанию предполагается, что сценарий выполняется в соответствии с синтаксисом той оболочки, из которой он был запущен. Это не всегда удобно. Например, вы написали сценарий в соответствии

с синтаксисом оболочки `csh`, но хотите, чтобы он мог быть запущен и из любой другой оболочки. В этом случае необходимо специальным образом оформить первую строку сценария, которая в данном случае должна выглядеть так:

```
#!/bin/csh
```

Это означает, что данный сценарий всегда будет выполняться с использованием синтаксиса оболочки `csh`, вне зависимости от того, из какой оболочки он запущен. Напомним, что в любой другой строке сценария, кроме первой, символ "`#`" в первой позиции строки означает комментарий.

8. Работа в сети

8.1. Доступ к удаленным компьютерам

Для доступа к удаленным компьютерам часто используется команда **telnet**:

telnet hostname

где `hostname` – сетевое имя или IP адрес удаленного компьютера, например, `mycomp.gdeto.ru` или 192.168.1.12. В случае удачного соединения у пользователя будут запрошены его имя и пароль на удаленном компьютере. В приведенном выше примере пользователь после соединения будет вводить команды в том же окне терминала, в котором он набрал команду **telnet**. Более сложный пример:

xterm –T Mycomp –e telnet mycomp.gdeto.ru &

Здесь открывается новое окно X терминала с заголовком `Mycomp` и в этом окне выполняется команда **telnet**. После соединения все команды на удаленном компьютере будут вводится в этом новом окне.

При соединении с помощью команды **telnet** пользователя ожидают проблемы в том случае, если он на удаленном компьютере захочет выполнить команду, которая должна открыть новое X окно, например, команду **emacs**. Вероятно, что он получит сообщение типа "Can't open display...". Это означает, что неверно установлена переменная окружения `DISPLAY`. Наберите команду

echo $DISPLAY

на локальном и на удаленном компьютерах – вы увидите, что значения этих переменных различны, например, на локальном компьютере

значение переменной DISPLAY есть localcomp.tut.ru:0.0, а на удаленном – mycomp.gdeto.ru:0.0 . Чтобы иметь возможность запускать любые X программы на удаленном компьютере, мы должны сообщить тому – где именно он должен открыть X окно. Именно этот смысл имеет переменная окружения DISPLAY – она указывает, где система должна открывать новые X окна. Таким образом, значение переменной DISPLAY должно указывать на локальный компьютер, т.е. на тот, на котором и должны открываться X окна. В нашем примере необходимо на удаленном компьютере выполнить команду

setenv DISPLAY localcomp.tut.ru:0.0

(для семейства C shells) или:

export DISPLAY=localcomp.tut.ru:0.0

(для семейства Bourne shells).

Но этого может оказаться недостаточным – при попытке открыть новое окно на удаленном компьютере вы по-прежнему будете получать сообщения, смысл которых сводится к тому, что ваш локальный компьютер не разрешает удаленному отрывать здесь X окна. Чтобы разрешить удаленному компьютеру сделать это, выполните на локальном команду:

xhost mycomp.gdeto.ru

Эта команда разрешает удаленному компьютеру mycomp.gdeto.ru открывать на вашем локальном компьютере новые X окна.

Можно сразу при соединении сообщить удаленному компьютеру имя своей переменной DISPLAY, указав ее в команде **xterm**:

xterm –display localcomp.tut.ru:0.0 –T Mycomp \
–e telnet mycomp.gdeto.ru &

Однако в любом случае надо быть готовым к тому, что поведение X приложений на локальном и удаленном компьютерах может оказаться различным. Это связано с возможно разной настройкой клавиатуры и шрифтов на локальном и удаленном компьютерах.

Основной недостаток соединения командой **telnet** заключается в том, что это соединение абсолютно не защищено от перехвата. Весь ввод-вывод с удаленным компьютером, в том числе и передача имени пользователя и пароля осуществляется открыто, без какого-либо шифрования. Если некий злоумышленник перехватит этот обмен между компьютерами, то он может получить доступ к системе от вашего имени со всеми вытекающими отсюда последствиями. Поэтому для соеди-

нения с удаленными компьютерами крайне желательно использовать протокол, который шифрует обмен данными между компьютерами, например, SSH (Secure SHell). Соединение с удаленным компьютером по протоколу SSH устанавливается так:

ssh hostname [*–l username*]

либо так:

ssh username@hostname

где `hostname` – имя удаленного компьютера, а `username` – имя пользователя на удаленном компьютере, которое в первом примере можно опустить, если оно совпадает с именем пользователя на локальном компьютере. При соответствующей настройке SSH автоматически передаст информацию о переменной `DISPLAY` и разрешит удаленному компьютеру открывать на локальном новые X окна. Если на вашем компьютере такие настройки не действуют по умолчанию, то добавьте при вызове опцию –**X**:

ssh hostname –X

Завершить соединение с удаленным компьютером можно командой **exit**.

Мы советуем использовать SSH протокол для соединения с удаленными компьютерами всегда, когда это только возможно. Единственная проблема может заключаться в том, что SSH просто не установлен на локальном или на удаленном компьютере, но таких компьютеров со временем становится все меньше. Программы для соединения по SSH протоколу существуют не только для Linux и других клонов UNIX, но и для Windows. Один из таких пакетов – PuTTY – можно скачать по адресу http://www.chiark.greenend.org.uk/~sgtatham/putty.

8.2. Обмен файлами между компьютерами

Для обмена файлами между компьютерами существует программа **ftp**.

ftp hostname

соединит пользователя (после ввода пароля) с удаленным компьютером `hostname`. Далее необходимо пользоваться собственными командами программы **ftp**, полный список которых можно получить, набрав *help*. С помощью команды

get filename

пересылается файл `filename` с удаленного компьютера в текущий каталог локального. Чтобы, напротив, отправить файл `filename` с локального компьютера на удаленный, введите

put filename

Перед пересылкой файлов можно выполнить команды *binary* или *ascii* для указания типа передаваемого файла – двоичный или текстовый соответственно. По умолчанию в большинстве случаев в Linux установлен двоичный обмен, в отличии от версии **ftp** для Windows, где по умолчанию предполагается текстовый обмен.

При указании имен файлов в командах *get* и *put* можно указывать полный путь с именами каталогов. Если указано только имя файла, то предполагается, что файл расположен в текущем каталоге, как на локальном, так и на удаленном компьютерах. Текущим каталогом сразу после соединения являются: на локальном компьютере – каталог, из которого была выполнена команда **ftp**, а на удаленном – обычно домашний каталог пользователя. Чтобы сменить текущий каталог на удаленном компьютере, пользуйтесь командой

cd dirname

где `dirname` – имя каталога на удаленном компьютере, который станет текущим. Увидеть имя текущего каталога на удаленном компьютере можно с помощью команды *pwd*. В некоторых версиях программы **ftp** существует команда *lcd*, которая меняет текущий каталог на локальном компьютере.

При необходимости передать или получить сразу несколько файлов, воспользуйтесь командами *mput* и *mget* соответственно. Так, например, командой

mget `*`*.dat*

мы получим с удаленного компьютера все файла с расширением `.dat`. Если вы не хотите, чтобы у вас спрашивалось подтверждение на передачу каждого из файлов, попадающий под указанный шаблон, наберите команду *prompt*.

Командой *hash* можно заставить **ftp** выводить на экран символ "`#`" при приеме или передаче каждого килобайта информации, что может быть полезным для контроля при передаче больших файлов. Чтобы выйти из программы **ftp**, наберите *quit*.

Кроме программы **ftp** на вашем компьютере могут быть установлены и другие программы, предназначенные для обмена файлами. Программа **ncftp** является несколько более развитой версией **ftp**: она

имеет более удобный пользовательский интерфейс, несколько дополнительных возможностей, таких, как возможности делать закладки (bookmarks) на часто посещаемых адресах. Кроме того, **ncftp** поддерживает возможность "докачки" файлов (если такая возможность поддерживается на удаленном компьютере). Другой программой, поддерживающей возможность "докачки", является программа **wget**. Эта программа может использоваться для перекачки не только отдельного файла, но и целой структуры файлов и каталогов. Отметим также, что **wget** может быть запущена в фоновом режим и работать тогда, когда пользователь уже вышел из системы. Кроме того, **wget** сама восстановит прерванное соединение, если такое произойдет, и продолжит докачку. Еще одна программа – **gftp** (Gnome ftp) имеет довольно удобный графический интерфейс.

Оболочка Midnight Commander (программа **mc**, раздел 2.4) также может использоваться для переноса файлов между компьютерами: наберите в командной строке **mc** команду

cd ftp://username@hostname/dirname

где `username` – имя пользователя на удаленном компьютере, `hostname` есть сетевое имя удаленного компьютера, а `dirname` – имя каталога на удаленном компьютере. Тогда, после удачного соединения, на одной из панелей Midnight Commander будет отображено содержимое каталога `dirname` на удаленном компьютере. Возможна навигация по файловой системе удаленного компьютера. Пересылка файлов между компьютерами осуществляется просто как копирование из одной панели **mc** в другую.

Все упомянутые выше программы для переноса файлов работают в открытом режиме, передавая данные, включая имя пользователя и пароль, в незашифрованном виде. Существенно повысить безопасность работы в сети при переносе файлов можно, если вместо **ftp** пользоваться программой **sftp** (Secure FTP), работающей по протоколу SSH. Работа с программой **sftp** напоминает работу с **ftp** – большинство команд у этих программ совпадают.

Если вам нужно переслать только один файл, или несколько файлов, подпадающих под один шаблон, то вместо **sftp** будет удобнее воспользоваться командой **scp** (Secure CoPy), также работающей по протоколу SSH. Формат команды **scp** при передаче файла с локального компьютера на удаленный таков:

scp filename username@hostname:dirname/filename1

При обратной операции – передаче файла с удаленного компьютера на локальный:

scp username@hostname:dirname/filename1 filename

В приведенных примерах `username` – имя пользователя на удаленном компьютере, `hostname` – имя удаленного компьютера, `dirname` – имя каталога на удаленном компьютере (по умолчанию – домашний каталог пользователя), `filename` – имя файла на локальном компьютере, `filename1` – имя файла на удаленном. Если `filename` и `filename1` должны совпадать, вместо имени файла можно поставить символ "**.**", аналогично команде **cp**. Например, команда

scp pp2g.f tikhomir@mycomp.gdeto.ru:star/soft/pomeron/.

передаст файл `pp2g.f` из текущего каталога локального компьютера в каталог `star/soft/pomeron/` пользователя `tikhomir` на удаленном компьютере `mycomp.gdeto.ru`, сохранив имя файла без изменения. А команда

scp tikhomir@mycomp.gdeto.ru:lib/.so .*

передаст все файлы с расширением `.so` из каталога `lib` пользователя `tikhomir` с удаленного компьютера `mycomp.gdeto.ru` в текущий каталог пользователя на локальном компьютере.

Если скорость обмена между локальным и удаленным компьютерами мала, то перед передачей больших файлов рекомендуется сначала попробовать уменьшить их размер с помощью команды **gzip** (см. раздел 2.11).

8.3. Просмотр Web страниц

Существует версии **netscape**, **mozilla**, а также некоторых других Web-браузеров для Linux. Графические оболочки KDE и GNOME имеют собственные встроенные программы для просмотра Web страниц.

В условиях, когда связь с удаленным компьютером медленная, может оказаться удобным использовать другую программу для просмотра Web страниц: **lynx**. Программа запускается из командной строки и работает в окне X терминала в текстовом режиме. Из-за отсутствия поддержки какой-либо графики, **lynx** значительно быстрее, чем **mozilla** и другие подобные программы, загружает Web страницы. Запустить **lynx** со стартовой страницы, например, `www.redhat.com`, можно так:

lynx www.redhat.com

Навигация по Web странице осуществляется клавишами <↑>, <↓>, <←>, <→> и <Enter>. В нижней части окна выводятся подсказки к некоторым командам, полный список которых можно просмотреть, нажав клавишу <h>. Отметим только, что загрузить файл или ссылку, который в данный момент выделен в окне **lynx**, можно, нажав клавишу <d>. Многочисленные настройки программы устанавливаются через Options Menu, в которое можно войти, нажав <o>.

8.4. Работа с электронной почтой

Для работы с электронной почтой в Linux существует множество различных программ. Это может быть и встроенные почтовые агенты в браузерах типа **netscape** или **mozilla** и специальные программы графических оболочек, например **kmail** в KDE. Одной из наиболее распространенных программ для работы с электронной почтой в Linux является **pine**. Ее преимущество заключается в том, что она поставляется практически со всеми дистрибутивами Linux и то, что она может работать в текстовом режиме терминала. Опишем здесь работу с программой **pine** более подробно.

Главное меню программы **pine** показано на рис. 5. Вы можете выбрать один из пунктов главного меню, либо нажав на клавиатуре один из символов ("?", "C" и т.д.), показанных с левой стороны экрана, либо выделив клавишами <↑> и <↓> нужную строку и нажав <Enter>. В двух нижних строках в окне **pine** даны подсказки, которые являются контекстно-зависимыми: их содержание меняется в зависимости от того, в каком из режимов вы работаете.

В **pine** пользователь имеет набор папок (folders), в каждой из которых содержится определенный класс писем. Список всех папок можно увидеть, выбрав `FOLDER LIST` в главном меню **pine**. В папке `INBOX` хранятся входящие письма. Папка `sent-mail` содержит письма, отправленные пользователем в текущем календарном месяце. В конце каждого месяца папка `sent-mail` переименовывается в папку с именем, например, `sent-mail-nov-2006`. Существуют папки, предназначенные для хранения писем, отправление которых было на какое-то время отложено (папка `postponed-msgs`), или для хранения особо важных полученных писем (папка `saved-messages`). Пользователь может создать и свои собственные папки. По умолчанию та папка, которую пользователь смотрел последний раз (обычно это папка `INBOX`), становится текущей и открывается при выборе пункта меню `MESSAGE INDEX`.

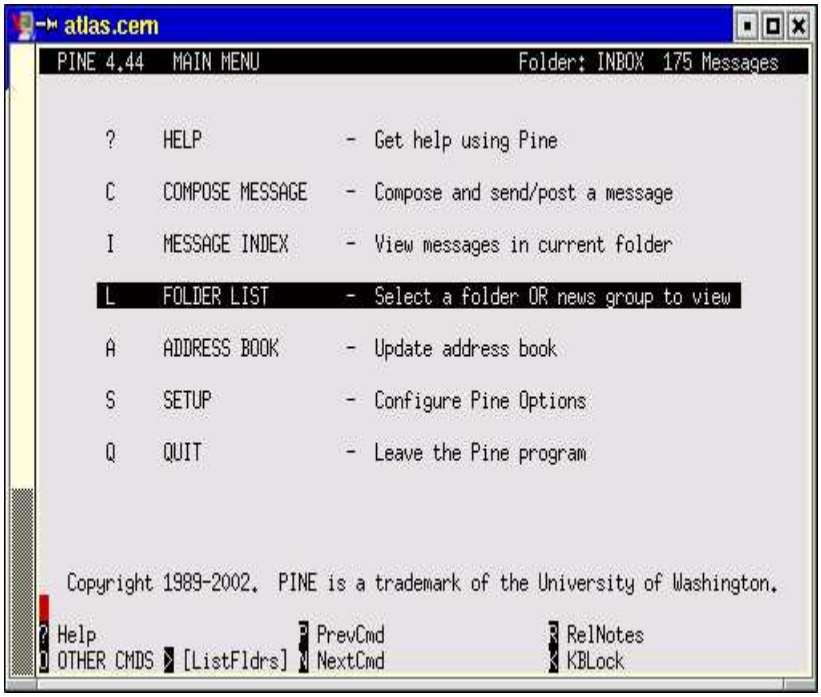

Рис. 5. *Главное меню программы* ***pine***

Для отправления письма следует выбрать в главном меню пункт `COMPOSE MESSAGE`. Мы попадаем в режим составления письма. Здесь в поле `To:` указываем электронный адрес получателя письма, в поле `Cc:` – адрес другого получателя, которому мы, возможно, хотели бы послать копию письма. При необходимости можно указать несколько адресов, разделяя их запятой. В поле `Subject:` указываем заголовок (тему) письма.

Затем в основном поле этого окна составляется текст письма. Редактор **pico**, который по умолчанию встроен в программу **pine**, нами уже рассматривался в разделе 3.2. После того, как письмо составлено, его можно отправить, нажав <`Ctrl-x`>. Копия отправленного письма будет сохранена в папке `sent-mail`. Если вы решили отложить на какое-то время отправку письма, нажмите <`Ctrl-o`> – письмо будет сохранено в папке `postponed-msgs`. Когда в следующий раз вы за-

пустите **pine** и захотите войти в меню `COMPOSE MESSAGE`, вам будет задан вопрос – не хотите ли вы продолжить отложенное (postponed) письмо? Отложенное письмо (или несколько) будут храниться в папке `postponed-msgs` до тех пор, пока пользователь их не удалит. Если пользователь передумал отправлять письмо, он всегда может нажать <`Ctrl-c`> – письмо не будет отправлено и не будет сохраняться в какой-либо папке[3].

Если к отправляемому письму необходимо присоединить один или несколько документов – например, файл в PostScript формате или документ MS Word, переведите курсор в строку `Attchmnt:` и нажмите <`Ctrl-j`>. Вам будет предложено ввести имя файла для подсоединения к письму.

Открыв папку `INBOX`, мы получим список входящей корреспонденции. По умолчанию список пришедших писем отсортирован по времени получения. Те письма, которые вы еще не прочли, отмечены символом "N" с левой стороны строки. Письма, на которые был дан ответ командой reply (см. ниже), отмечены буквой "A". Знаком "+" отмечены письма, которые отправлены лично вам (а не списку адресатов, среди которых есть и ваш адрес). Если вы собираетесь удалить какое-либо из сообщений, то подведите к нужной строке курсор и нажмите клавишу <`d`> – в левой части строки появится символ "D" – это означает, что письмо подготовлено к удалению. Таким же образом можно удалить сообщения и из любых других папок – `sent-mail` или `postponed-msgs`.

Выделив какое-либо из полученных писем в списке папки `INBOX` и нажав <Enter>, мы попадаем в режим просмотра письма. При желании на полученное письмо можно ответить, нажав клавишу <`r`> (reply) – вы автоматически перейдете в режим отправки писем, где в строке `To:` уже будет стоять адрес вашего респондента. Вам также будет задан вопрос – хотите ли вы, чтобы текст присланного вам письма был включен в текст вашего ответа? Далее вы можете составлять, редактировать и отправлять письмо обычным образом.

Вы можете также переслать полученное вами письмо третьему лицу – для этого, выделив соответствующее письмо, нажмите клавишу <`f`> (forward). Вы опять перейдете в режим отправки писем, где текст полученного вами письма будет уже включен в текст отправляемого. Остается только ввести адрес человека, которому вы направляете письмо и, если это необходимо, вставить в текст какую-либо преамбулу.

[3]Но оно будет записано в файл `dead.letter` в домашнем каталоге пользователя.

Еще две важных операции, которые вы можете сделать с полученными письмами: сохранить особо важные из них в специальной папке `saved-messages`, или записать копию письма в виде файла на диске. Для первой операции, выделив письмо, нажмите клавишу <s>, а для второй – клавишу <e>.

Если вы хотите сразу из программы **pine** распечатать полученное письмо, то нажмите клавишу <%>. Но заметьте – это работает только тогда, когда у вас определена переменная окружения `PRINTER`. В противном случае вам придется сначала записывать письмо в виде файла на диск, а потом, уже выйдя из программы **pine**, этот файл распечатывать. Заметим, что подсказки для команд `save`, `export` и `print`, которым соответствуют клавиши <s>, <e> и <%>, изначально не видны в двух нижних строках окна программы **pine** – чтобы увидеть подсказки к этим и другим командам, нажмите клавишу <o> (`OTHER CMDS`).

Выбрав пункт `ADDRESS BOOK` в главном меню программы **pine**, вы сможете назначить короткие nicknames (псевдонимы) для электронных адресов тех корреспондентов, с которыми вы ведете активную переписку. Тогда при составлении письма в поле `To:` можно будет вместо полного адреса указывать nickname – программа сама переведет этот псевдоним в полный адрес.

Когда несколько человек работают над одним проектом, часто бывает необходимым направить письмо сразу всем участникам проекта. Для этого в `ADDRESS BOOK` можно составить список рассылки: в качестве nickname указать некое ключевое слово, например, `project`, а в поле для реальных адресов перечислить через запятую адреса всех участников проекта. Тогда при составлении письма в поле `To:` достаточно будет указать: `project` – и **pine** включит в число получателей всех, кто в `ADDRESS BOOK` перечислен под псевдонимом `project`.

Наконец, в пункте `SETUP` главного меню можно изменить огромное число настроек программы **pine**. Эти настройки сохраняются в файле `.pinerc` в домашнем каталоге пользователя. Файл `.pinerc` – текстовый и для изменения настроек можно просто редактировать этот файл. Но любую из этих возможностей – меню `SETUP` и редактирование файла `.pinerc` – следует использовать с крайней осторожностью. Если вы не вполне понимаете, что именно вы делаете, то лучше не рисковать и оставить все настройки, как они есть. В любом случае перед тем, как изменять настройки программы **pine**, сохраните текущую версию файла `.pinerc`.

При необходимости вы можете настроить свою почтовую систему на Linux компьютере таким образом, чтобы приходящие на этот компьютер письма перенаправлялись на какой-либо другой адрес, например тот, который вы чаще всего просматриваете. Для этого создайте в вашем домашнем каталоге файл с именем `.forward`. В этот файл записывается e-mail адрес, на который вы хотите перенаправлять письма. Таких адресов может быть и несколько, по одному в одной строке файла. Чтобы при этом приходящие письма сохранялись и в почтовом ящике на данном компьютере, нужно вставить в файл `.forward` имя пользователя. Например, пользователь с именем `user1` хочет перенаправлять приходящие ему письма на адрес `vasya123@mail.ru` и оставлять копии на локальном компьютере. Тогда файл `.forward` должен выглядеть следующим образом:

```
user1
vasya123@mail.ru
```

8.5. Интерактивный диалог с пользователями

Представим, что вы хотите послать срочное сообщение одному из пользователей, зарегистрированному вместе с вами на одном компьютере. Пусть username этого пользователя, например, `vova`. Сначала убедитесь, что `vova` реально работает в данный момент на этом компьютере. Воспользуйтесь для этого командой **who** (раздел 5), добавив опции –**i** и –**w**:

who –iw | grep vova

Вы увидите, например, следующее:

```
vova      -   pts/3    Jan  2 14:58 00:01
vova      +   pts/5    Jan  2 15:11 00:04
```

Таким образом мы видим, что у пользователя `vova` на данном компьютере открыты два окна терминала: `pts/3` и `pts/5`. Последний столбец каждой строки показывает – сколько времени (в часах и минутах) пользователь был неактивен в данном окне, т.е. какое время назад он последний раз набирал там что-то на клавиатуре. Если все эти времена составляют, например, несколько часов, то пользователь, скорее всего, не работает в данный момент на компьютере – просто он ушел, не закрыв соответствующее окно терминала. Во втором столбце стоят знаки “+” или “–”. Они показывают – разрешает или нет пользователь `vova`

посылать ему сообщение на данный терминал. Пользователь может разрешить или запретить это с помощью команды **mesg**. Некоторые команды, например **pine**, запущенные в данном окне, автоматически запрещают посылку сообщений на этот терминал.

Как видно из приведенного примера, пользователь `vova` был недавно активен на терминале `pts/5` и прием сообщений на данный терминал разрешен. Тогда мы можем послать ему сообщение командой **write**:

write vova pts/5

Введя данную команду, вы можете далее набрать текст сообщения, переходя на новую строку, если это необходимо, нажав <`Enter`>. Вводимый текст немедленно появится у пользователя `vova` в окне терминала `pts/5`. Завершить ввод сообщения можно, нажав <`Ctrl-d`>.

Другая программа, предназначенная для интерактивного общения с пользователями в системе – **talk**. Формат этой команды сходен с форматом команды **write**. Для приведенного выше примера можно набрать:

talk vova pts/5

Предположим, что имя пользователя, выполнившего команду **talk** – `tikhomir`, и работает он на компьютере с именем `mycomp.gdeto.ru`. Тогда у пользователя `vova` на экране терминала `pts/5` появится сообщение"`talk: connection requested by tikhomir@mycomp.gdeto.ru`" и будет выдан звуковой сигнал. Пользователь `vova` может ответить просто

talk tikhomir

Тогда между пользователями `tikhomir` и `vova` установится связь в режиме реального времени. Экран терминала каждого из пользователей будет разделен на две части. В верхней части будет появляться текст, который в данный момент времени вводит на клавиатуре сам пользователь, а в нижней – его vis-à-vis. Завершить работу программы **talk** можно, нажав <`Ctrl-c`>.

Отметим, что системные администраторы компьютеров из соображений безопасности могут отключить сервисы **write** и **talk**.

9. Русификация в Linux

Здесь рассмотрены некоторые вопросы, связанные с использованием русского языка в Linux. Возможно, ваш дистрибутив Linux уже является полностью русифицированным – тогда вы можете пропустить эту главу.

К сожалению, вопрос поддержки иностранных (по отношению к английскому) языков в Linux решается достаточно сложно – практически каждая программа требует отдельной настройки. Не будем здесь рассматривать все вопросы, связанные с использованием русского языка в Linux, а коснемся лишь самых важных моментов. Для получения дополнительной информации вы можете посмотреть Cyrillic HOWTO страницу, наиболее современная версия которой, по-видимому, находится на Web сайте по адресу `www.inp.nsk.su/~baldin`.

Отметим вначале, что в Linux, как и в других операционных системах семейства UNIX, основной кодировкой русских шрифтов является KOI8-R, хотя при желании можно использовать и Windows кодировку CP-1251 или MS-DOS кодировку CP-866. Вы можете конвертировать текстовые файлы из одной кодировки в другую с помощью команды **iconv**. Например,

iconv –f CP1251 –t KOI8R –o newfile oldfile

преобразует текстовый файл `oldfile` в кодировке CP-1251 в `newfile` в кодировке KOI8-R.

Далее необходимо убедиться, что на вашей машине установлены русские шрифты. Напомним, что нас интересует, в первую очередь, работа в графическом режиме, т.е. речь идет о шрифтах для X Window. Список всех доступных X шрифтов выдается командой **xlsfonts**. Чтобы увидеть, есть ли в системе шрифты в кодировке KOI, наберите в командной строке

xlsfonts | grep –i koi

Соответственно, если вас интересуют Windows шрифты, попробуйте

xlsfonts | grep –i win

или

xlsfonts | grep 1251

Если эти команды не выдадут на экран ничего – то у вас нет доступных русских шрифтов. В этом случае необходимо обратиться к вашему системному администратору.

Если русские X шрифты в вашей системе установлены, можно открыть X терминал, в котором будут видны символы кириллицы:

xterm –fn fontname &

где `fontname` – это одно из имен шрифтов, которое выдала команда **xlsfonts**. Для использования в окнах терминала или редактора **emacs** следует выбирать шрифты с фиксированной шириной символов – это шрифты семейств `courier` или `fixed`. После того, как вы подберете для себя подходящие по размерам и начертанию шрифты, то для удобства рекомендуется запомнить эти имена, присвоив их каким-либо переменным окружения. Как это сделать и как потом использовать такие переменные, показано в примере раздела 4.4.

Укажем еще на одну возможность для просмотра русских символов в окне X терминала. Как было сказано в разделе 4.5, в корневом каталоге пользователя есть файл с именем `.Xdefaults`, в котором можно переобозначить так называемые X ресурсы некоторых программ. В частности, программа **xterm**, которая открывает X терминал, имеет ресурсы, связанные с используемыми шрифтами. Шрифт в окне X терминала можно изменить, если навести курсор мыши на окно и одновременно нажать <`Ctrl`> на клавиатуре и правую кнопку мыши. По умолчанию в открывшемся меню можно выбрать шрифты разного размера. Изменив соответствующие X ресурсы, мы сможем вместо изменения размера задать переключение шрифтов в русскую кодировку. Вот какие строки можно для этого включить в файл `.Xdefaults`:

```
XTerm*fontMenu*font1*Label:  KOI8-R
XTerm*VT100*font1: \
-misc-fixed-medium-r-normal--14-130-75-75-c-70-koi8-r
XTerm*fontMenu*font2*Label:  CP-1251
XTerm*VT100*font2: \
-cronyx-fixed-medium-r-semicondensed--13-120-75-75-c-60-rawin-r
```

Мы заменили в меню смены шрифтов X терминала редко используемые опции `Unreadable` и `Tiny` на шрифты в русских кодировках KOI8-R и CP-1251. Теперь можно переключаться на русские шрифты динамически в уже открытом X терминале. Заметим, что все сказанное здесь относится именно к терминалу, который открывается командой **xterm**. Если вы пользуетесь другими типами терминалов, то у них будут свои способы настройки.

Итак, теперь мы можем видеть русские символы в окне X терминала. Этого достаточно, например, для того, чтобы читать письма на

русском языке в программе **pine** (раздел 8.4) или чтобы просмотреть русский текст из файла, выведенный на экран командой **more**.

Чтобы видеть русские шрифты в окне редактора **emacs**, отредактируйте файл `.emacs`, вставив в него следующие строки:

```
(standard-display-european t)
(set-input-mode (car (current-input-mode))
   (nth 1 (current-input-mode))
   0)
```

Первая инструкция необходима для русификации **emacs** в текстовом (консольном) режиме, вторая – для работы в графической X моде. Теперь вы можете запустить редактор **emacs**, который понимает русские буквы. Удобно ввести короткий псевдоним (alias) для команды запуска редактора, как это показано в том же примере раздела 4.4:

alias emkoi 'emacs –fn `$`*koi8 &'*

где переменная окружения `koi8` должна быть определена ранее и указывать на выбранный шрифт.

Для того чтобы иметь возможность вводить с клавиатуры символы на русском языке, необходима соответствующая программа. Если у вас установлена оболочка KDE с поддержкой национальных клавиатур, то вы можете назначить режимы дополнительной клавиатуры через KDE Control Center. В противном случае вам, возможно, придется устанавливать и настраивать программу переключения клавиатуры самостоятельно. Существует множество различных подходов к этой проблеме. Один из вариантов – это пакет Александра Лукьянова **xrus** (другое название – **xruskb**), который можно скачать из Интернет по следующим адресам:
`ftp://ftp.yars.free.net/pub/software/unix/X11/`
`ftp://ftp.relcom.ru/pub/x11/cyrillic/`
После того, как данные пакет установлен, вы можете запустить переключатель клавиатуры командой

xrus /usr/local/xruskb-1.13.0/keymaps/jcuken-koi8.xmm

где `/usr/local/xruskb-1.13.0` – каталог, в котором был установлен пакет **xrus**, а `jcuken-koi8.xmm` – файл с желаемой раскладкой клавиатуры и кодировкой. Переключение между латинскими и русскими символами в **xrus** по умолчанию осуществляется одновременным нажатием двух клавиш <`Shift`>.

Теперь у вас появилась возможность набирать русские символы в окне X терминала, в редакторах **pico** или **emacs**.

Рассмотрим еще настройку некоторых программ для работы с русскими шрифтами. Чтобы иметь возможность писать письма на русском языке в программе **pine**, отредактируйте файл `.pinerc` в вашем корневом каталоге. В нем необходимо найти строку `character-set` и установить

character–set=koi8-r

или, если предыдущий пример не срабатывает, то:

character–set=ISO–8859–5

Чтобы видеть русские символы в Midnight Commander **mc**, установите в нем опцию "Full 8 bit output" через меню `Options→Display bits`. А установленная опция "Full 8 bit input" позволит также вводить русские символы с клавиатуры во встроенном редакторе **mc**.

В разделе 10.2 рассказано, как использовать программу **aspell** или для проверки русской орфографии. Если вы хотите использовать эту программу в редакторе **emacs**, не забудьте сначала переключиться на русский словарь через меню `Tools→Spell Checking→Select Russian Dict`.

Если на вашем компьютере установлены русские PostScript шрифты, то для печати текстового файла, содержащего кириллицу, можно конвертировать его в PostScript формат с помощью программы **a2ps** (раздел 6) и затем распечатать:

a2ps –encoding=KOI8 filename –o filename.ps

Без опции `-o` вывод программы **a2ps** будет направлен сразу на принтер.

Если на компьютере установлен пакет TEX с поддержкой русского языка (см. раздел 10.2), то у вас не должно быть проблем с переводом русского текста в PostScript формат и с последующей печатью – TEX содержит все необходимое. Данная книга набрана именно в TEX (более точно, использовался пакет LATEX 2_ε) с такой преамбулой в TEX файле:

```
\documentclass [10pt,a5paper,twoside]{article}
\usepackage[koi8-r]{inputenc}
\usepackage[english,russian]{babel}
```

Параметры программы **lynx** (раздел 8.3), предназначенной для просмотра Web страниц в текстовой моде, устанавливаются через Op-

tions Menu, в которое можно попасть, нажав клавишу <o>. Для просмотра Web страниц на русском языке необходимо установить соответствующие кодировки: KOI8-R или CP-1251.

Полная русификация операционной системы включает в себя также представление в соответствующих форматах даты и времени, национальной валюты, представления действительных чисел, сообщений операционной системы и т.п. Форматом вывода такого рода управляют переменные окружения, описанные в справочной странице locale:

man 7 locale

Если вам действительно все это необходимо, то попробуйте установить значение переменной окружения `LC_ALL` как `ru_RU.KOI8-R`.

Если на вашем компьютере есть соответствующие переводы, то возможен просмотр справочных страниц различных команд на русском языке. Для этого необходимо задать переменную `LANG`:

setenv LANG ru_RU.KOI8-R

(для семейства C shells) или:

export LANG=ru_RU.KOI8-R

(для семейства Bourne shells).

Разумеется, все отмеченные здесь команды будут корректно работать только в окне X терминала, который поддерживает соответствующую русскую кодировку.

10. Разные программы

В этом разделе мы расскажем о некоторых полезных программах и командах, которые обычно входят в дистрибутивы Linux.

10.1. Работа с графикой

Одной из наиболее часто используемых в Linux программ является программа просмотра файлов в формате PostScript – **gv** (рис. 6). PostScript формат является одним их самых распространенных форматов для хранения и качественной печати документов со смешанным текстовым и графическим содержанием. В PostScript формате хранятся очень многие документы в сети – научные статьи, книги, различная документация и т.д. Многие программы умеют экспортировать свой вывод в PostScript формат. Наконец, документ в PostScript формате

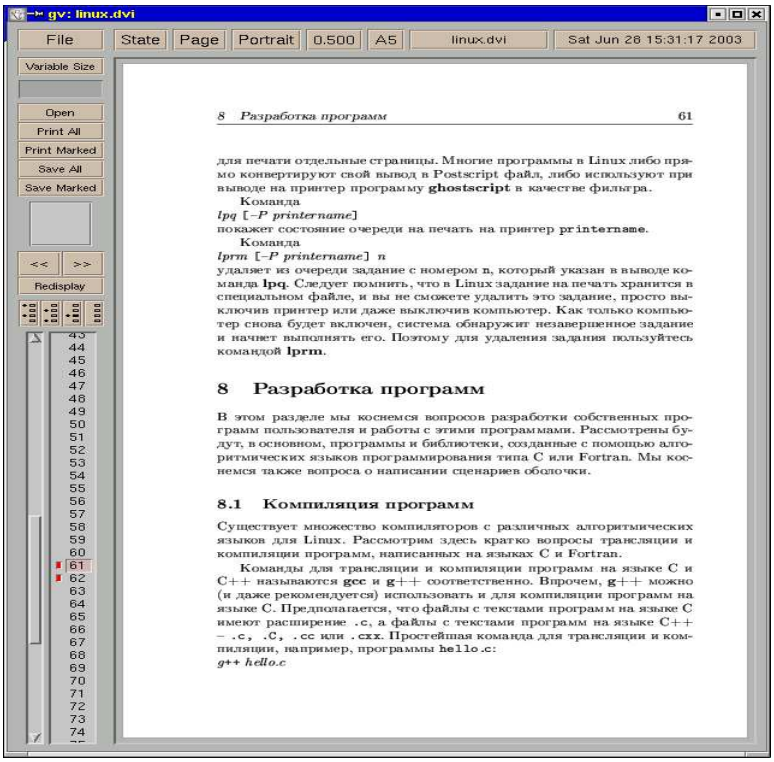

Рис. 6. *Программа* ***gv***

может быть распечатан практически на любом принтере при наличии программы-интерпретатора языка PostScript. Таким интерпретатором является программа **gs** (или **ghostscript**). Программа **gv** (или **ghostview**) является надстройкой над **ghostscript**. Кроме PostScript фалов, с помощью **gv** можно также просматривать файлы в PDF формате.

В программе **gv** можно отобрать для печати четные или нечетные страницы (две левые кнопки под кнопкой `Redisplay`), либо отдельные страницы, щелкнув средней кнопкой мыши (или одновременно левой и правой, если средней кнопки нет) на номере страницы с левой стороны панели. В примере на рис. 6 отмечены страницы 61 и 62. Напечатать отмеченные страницы можно, нажав на кнопку `Print Marked`.

Отметим еще одну возможность программы **gv**. Наведите курсор мыши на основное окно программы, нажмите среднюю кнопку мыши и, не отпуская ее, сдвиньте курсор мыши в сторону. В окне выделится прямоугольник. Теперь отпустите кнопку мыши – рядом с курсором появится панелька с цифрами от 2 до 64: это коэффициенты увеличения (zoom) области в прямоугольнике. Нажав левой кнопкой мыши на одну из цифр, вы откроете новое окно с увеличенным фрагментом. Такая возможность бывает полезна, если необходимо лучше рассмотреть какой-то фрагмент текста или рисунка.

Иногда для того, чтобы шрифты в окне программы **gv** выглядели аккуратнее, нужно включить опцию `Antialias` в меню, которое открывается нажатием кнопки `State`. Это позволит сгладить некоторую "зазубренность" экранных шрифтов. Если такая опция не включена у вас в программе **gv** по умолчанию, вы можете сами это сделать, изменив соответствующий X ресурс (см. раздел 4.5). Включите в файл `.Xdefaults`, расположенный в вашем корневом каталоге, такую строчку:

```
gv.antialias: True
```

Для просмотра файлов в PDF формате существует также программы и **xpdf** и **acroread**.

Довольно удобная программа векторной графики **xfig** может быть использована для рисования несложных рисунков, диаграмм, блок-схем (рис. 7). **xfig** умеет экспортировать свой вывод в различные графические форматы: PostScript, gif, jpeg, tiff и другие.

Очень мощный графический пакет растровой графики **gimp** позволяет работать с несколькими слоями изображения, создавать различные спецэффекты, в том числе и анимационные. Он может работать с очень большим числом различных графических форматов.

С помощью команды **convert** можно конвертировать графический файл из одного формата в другой. Если вас интересуют программы для работы с каким-то конкретным форматом графики, попробуйте набрать

man -k format

где `format` – интересующий вас графический формат, например, `jpeg`, `gif` или `PostScript`.

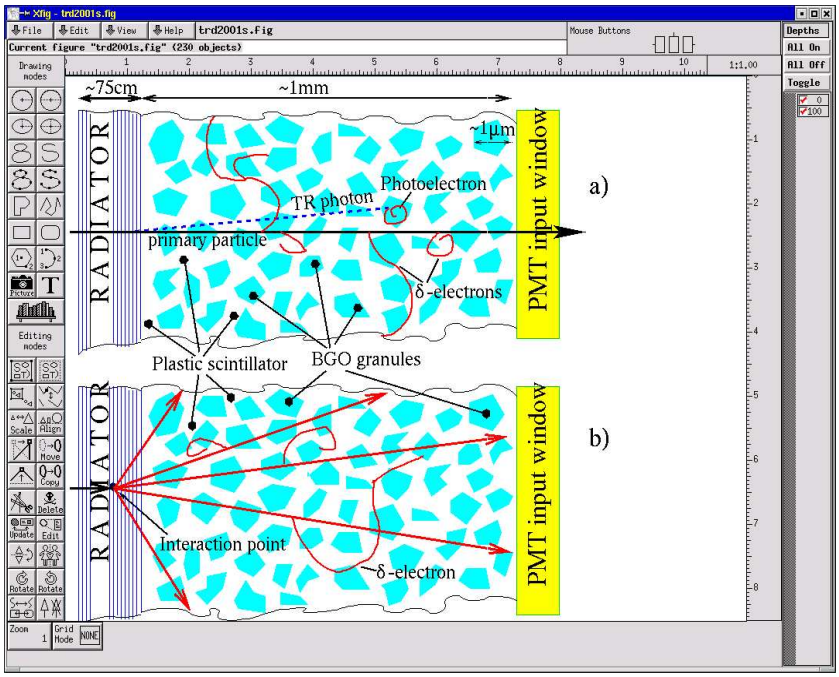

Рис. 7. *Программа* ***xfig***

10.2. Офисная работа

Наиболее мощным офисным пакетом для Linux в является **Open Office**. В него входят текстовый редактор, программы для работы с электронными таблицами, средства создания электронных презентаций, и другие полезные программы. Важно, что **Open Office** способен читать и сохранять документы не только в собственных форматах, но и в соответствующих форматах программ Microsoft Office. Web страница проекта расположена на `www.openoffice.org`.

В научной среде наиболее распространенным пакетом для написания статей является TEX. TEX обеспечивает полиграфическое качество представления материалов. Статьи в формате TEX принимаются практически всеми редакциями научных журналов. Данная книга также набрана в TEX. Основные команды при работе с TEX файлами в Linux таковы:

latex filename

где `filename` – имя TEX файла (если файл имеет расширение `.tex` или `.latex`, то его можно не указывать). После "трансляции" TEX файла командой **latex** создается файл с именем `filename.dvi`, который можно просмотреть с помощью программы **xdvi**, перевести в PostScript формат командой

dvips filename`[`*.dvi*`]` *–o filename.ps*

или сразу направить на печать, если не указывать опцию **–o**.

Программа **aspell** предназначена для проверки орфографии текста в файле. Формат вызова **aspell** для проверки текстового файла:

aspell –c filename

Можно включить специальные опции – фильтры – для проверки текстов в HTML, TEX, e-mail и некоторых других форматах. При наличии соответствующего словаря, **aspell** может проверять тексты и на русском языке. Пользователь может также создавать свои собственные словари и подключать их дополнительно к основному. Эта программа встроена в редактор **emacs** (раздел 3.3).

Команда **cal** распечатает в окне терминала календарь (по умолчанию – текущего месяца).

xcalc представляет собой калькулятор для несложных вычислений.

Множество полезных программ и утилит содержится в оболочках KDE и GNOME: калькуляторы, органайзеры, различные редакторы, программы для записи CD, отправки факсов, различные мультимедийные программы и многое, многое другое. Однако обзор этих программ выходит за рамки данной книги.

Список литературы

1. Костромин В. Linux для пользователя. СПб.: БХВ-Петербург, 2005. – 658 с.

2. Фролов И. Как установить, настроить и использовать Linux. М.: Познавательная книга плюс, 2001. – 224 с.

3. Водолазкий В.В. Путь к Linux: учебный курс. СПб.: Питер, 2002. – 400 с.

4. Ивановский С. Операционная система Linux. Справочное руководство. М.: Познавательная книга плюс, 2001. – 221 с.

5. Баррет Даниэл Дж. Linux: основные команды. Карманный справочник. М.: КУДИЦ-Образ, 2007. – 288 с.

6. Бендел Д. Использование Linux. М.; СПб.; К.: Издательский дом "Вильямс", 2002. – 784 с.

7. Визерспун К. и К. Освой самостоятельно Linux за 24 часа. М.; СПб.; К.: Издательский дом "Вильямс", 2002. – 352 с.

8. Лебланк Д.-А. Linux для "чайников". М.; СПб.; К.: Издательский дом "Вильямс", 2007. – 464 с.

9. Петерсен Р. Энциклопедия Linux. Наиболее полное и подробное руководство. СПб.: БХВ-Петербург, 2003. – 1008 с.

Приведем также несколько Интернет-ссылок, по которым пользователь может найти полезную информацию, касающуюся Linux.

Сайты о Linux общего характера:
http://www.linux.org
http://www.linux.com
http://www.tldp.org – сайт Linux Documentation Project
http://www.redhat.com – сайт одной из самых известных фирм, создающих дистрибутивы Linux – RedHat Linux Home Page
ftp://sunsite.unc.edu/pub/Linux – ftp сервер Sunsite Linux
ftp://ftp.funet.fi/pub/Linux – ftp сервер Finnish University and Research network
http://www.ibiblio.org/pub/Linux – Metalab Linux Archives

Linux в научных исследованиях:
http://www.sai.msu.su/sal – Scientific Applications On Linux

http://linux.cern.ch – Linux в Европейском Центре Ядерных Исследований (ЦЕРН)

Русскоязычные сайты:
http://www.linux.ru
http://www.linux.org.ru
http://www.linux.ru.net
http://www.opennet.ru
http://unixware.ru
http://www.linux.org.ru/books – обширная документация по Linux на русском языке

Предметный указатель