«Основы работы в операционной системе Linux»

С.А. Немнюгин

Тема 1. Первое знакомство с ОС UNIX

Операционные системы

Операционная система представляет собой программный комплекс, который выполняет две основные функции:

1. обеспечение удобного интерфейса между пользователем (или его программой) и компьютером;
2. эффективное управление ресурсами компьютера.

Для решения этих задач различные ОС используют различные алгоритмы, что и определяет их производительность, область применения, пользовательский и программный интерфейс, а также другие особенности.
По числу одновременно выполняемых процессов операционные системы делят на два класса:

1. *однозадачные* (например, MS-DOS);
2. *многозадачные* (UNIX, OS/2).

Однозадачные ОС включают средства управления файловой подсистемой, периферийными устройствами и другими ресурсами компьютера, а также обеспечивают удобный пользовательский интерфейс. Многозадачные ОС, кроме того, управляют распределением между процессами совместно используемых ресурсов.
Различают также ОС следующих типов:

1. *однопользовательские* ОС (MS-DOS, RT-11);
2. *многопользовательские* ОС (UNIX, VAX-VMS, SVM для IBM 360/370, Windows NT).

Основное отличие многопользовательских систем от однопользовательских заключается в наличии средств защиты информации каждого пользователя от несанкционированного доступа со стороны других пользователей.
В ОС *разделения времени* каждой задаче по очереди выделяется один «квант» времени, так, что ни одна задача не занимает процессор надолго.
В ОС *реального времени*, определен максимальный интервал времени, в течение которого пользовательская программа получит управление при возникновении внешнего по отношению к ЭВМ события.
Широкое распространение получили многопроцессорные вычислительные комплексы. Операционные системы иногда делят на ОС *с поддержкой многопроцессорной обработки данных* (UNIX, Windows NT) и без такой поддержки.
Современные операционные системы должны эффективно решать основные задачи — обеспечивать удобный пользовательский и программный интерфейс, а также эффективно управлять ресурсами компьютера. Обязательными стали многозадачность, наличие механизмов работы с виртуальной памятью, многооконный графический интерфейс. Современная ОС должна быть надежной,

безопасной. Она должна использовать эффективные алгоритмы распределения важнейших ресурсов, таких как процессорное время и память компьютера.
Важным свойством является переносимость ОС, которая обеспечивается тем, что основная часть кода операционной системы должна быть написана на хорошо стандартизованном языке программирования высокого уровня.
Особое значение имеет безопасность ОС. Безопасность — это защита информации каждого пользователя от несанкционированного доступа, а также защита системы от неправильных действий пользователя.

ОС UNIX

Название UNIX объединяет семейство многозадачных и многопользовательских операционных систем разделения времени, имеющих сходный пользовательский и программный интерфейс. Общие черты UNIX-систем:

1. *мультипрограммная обработка* в режиме разделения времени, основанная на вытесняющей многозадачности;
2. *поддержка многопользовательского режима*, наличие средств защиты данных от несанкционированного доступа;
3. *использование механизмов виртуальной памяти и свопинга*;
4. *иерархическая файловая система*, образующая единое дерево каталогов независимо от количества физических устройств, используемых для размещения файлов;
5. *унификация операций ввода/вывода* на основе расширенного использования понятия «файл»;
6. *переносимость системы*, благодаря написанию ее основной части на языке С;
7. *кэширование диска* для уменьшения среднего времени доступа к файлам;
8. *наличие разнообразных средств взаимодействия процессов*, в том числе и через сеть.

Основные понятия ОС UNIX

Компьютеры, работающие под управлением ОС UNIX, выполняют одну из двух функций — *сервера* или *рабочей станции*. Сервером называют компьютер, который предоставляет в распоряжение пользователей один или несколько видов ресурса. В зависимости от предоставляемого ресурса различают *файловые* серверы, серверы *вычислений*, серверы *печати* и другие.
Для эффективного выполнения функции сервера компьютер должен работать непрерывно и устойчиво. Это условие накладывает особо строгие требования к надежности и устойчивости установленной на сервер операционной системы. ОС UNIX является одной из наиболее подходящих операционных систем для сервера.
Процессы. Запуск на выполнение любой прикладной или системной программы порождает один или несколько *процессов*. ОС UNIX управляет процессами, распределяя между ними ресурсы компьютера. Возможности обычного пользователя ограничены запуском и остановом собственных процессов, а также снижением их приоритета.
Файлы и файловая система. Файл представляет собой совокупность однотипных данных, которой присвоено определенное имя и которая находится в памяти

компьютера (оперативной или дисковой). Пользователь работает с файлами — создает их, редактирует, копирует, удаляет и т. д.

Имена файлов в ОС UNIX могут иметь большую длину. Буквы в нижнем и верхнем регистрах различаются операционной системой. В некоторых интерпретаторах команд (bash и tcsh) в UNIX имеется механизм, облегчающий набор имен файлов. При наборе имени файла достаточно набрать первые символы имени, однозначно идентифицирующие файл, а затем нажать клавишу табуляции Tab. Набранная часть при этом будет дополнена до полного имени, а если набранные символы являются начальными символами имен нескольких файлов, на экран будет выведен список этих имен.

Пользователи. В UNIX принято различать две категории пользователей — *обычные пользователи* и *суперпользователь*. Права обычных пользователей ограничены. Им запрещена запись в системные каталоги и изменение конфигурационных файлов системы. Обычный пользователь не может увеличить приоритет своей программы при ее выполнении. Обычному пользователю может быть запрещено пользоваться некоторыми периферийными устройствами. Обычный пользователь не может зарегистрировать в системе нового пользователя или удалить уже имеющегося. Только суперпользователь может зарегистрировать в системе нового пользователя.

Сеанс работы. Сеанс работы представляет собой последовательность действий, выполняемых пользователем, от входа в систему до подачи команды выхода из нее. В процессе работы пользователь может запускать прикладные программы и те утилиты, право на запуск которых он имеет.

Один пользователь может одновременно вести несколько сеансов — в UNIX нет ограничений на их количество. При непосредственной работе за терминалом в Linux допускается использование нескольких виртуальных консолей, переключение между которыми производится нажатием на комбинации клавиш Alt/F1, Alt/F2,…. С каждой консоли можно открыть и вести отдельный сеанс работы в UNIX.

Интерпретатор команд. Интерпретатор представляет собой специальную программу. Основная задача интерпретатора команд состоит в считывании вводимых пользователем команд, проверке их правильности и выполнении. Интерпретатор команд запускается при входе пользователя в систему. Пользователь может запустить несколько экземпляров одного и того же или разных интерпретаторов команд.

После того, как пользователь открывает сессию работы с текстового терминала, введя свое регистрационное имя и пароль, операционная система для работы с пользователем запускает интерпретатор команд. Интерпретатор сначала выполняет команды, записанные в специальных стартовых командных файлах. После этого на экран выводится «приглашение» командной строки и пользователь может вводить команды. Вводя команду logout или exit, пользователь требует завершить сессию, а интерпретатор перед выходом выполняет команды, записанные в специальных «завершающих» командных файлах.

Пользователь может определить, с какой оболочкой он работает с помощью команды echo $SHELL. При выполнении этой команды на экран будет выведено имя оболочки сессии, назначенной пользователю при регистрации.

Текстовый интерфейс пользователя. *Интерфейс* — это способ взаимодействия между операционной системой и пользователем или программой. При работе в ОС UNIX пользователь может использовать как текстовый, так и графический интерфейс.

Общение с системой в текстовом режиме заключается в подаче команд и получении результатов их выполнения. Если командный интерпретатор готов принять команду, он выводит на экран приглашение, вид которого может быть разным в разных интерпретаторах. Далее мы будем условно считать, что приглашение имеет вид:

```
#
```

Формат вызова команды в общем случае следующий:

```
# команда [ключи] [аргументы]
```

Почти все команды могут работать в нескольких различных режимах. Для указания режима выполнения команды используются *ключи*. При указании ключей часто, но не всегда, вначале размещается символ «минус», а затем указываются один или несколько символов. В некоторых командах требуется указать объект, к которому применяется команда. Для указания объекта используются *аргументы* команды. Такими объектами могут быть, например, имена файлов или каталогов.

```
# ls
letter  mysound  simulation
```

В этом случае на экран будут выведены только имена файлов, расположенных в текущем каталоге, то есть в том каталоге, в котором пользователь работает в настоящий момент. Имена файлов, начинающиеся с символа «точка» выводиться не будут. Если вызвать эту же команду с ключом `-a`, то выводиться будут имена всех файлов. Использование ключа `-l` приводит к выводу на экран дополнительной информации: права доступа к файлу, имя его владельца, размер и т. д. Допускается совместное использование нескольких ключей. Результатом выполнения команды `ls -al` будет вывод подробной информации обо всех файлах текущего каталога.
Если необходимо просмотреть содержимое не текущего каталога, а какого-либо другого, необходимо указать его имя в качестве аргумента:

```
# ls -al /bin
```

Такая форма вызова команды `ls` позволяет получить список файлов, содержащихся в системном каталоге `/bin`.

Регистрация на UNIX-ЭВМ

UNIX — многопользовательская система. Для ограничения доступа к UNIX-системе используются *регистрационное имя пользователя* и *пароль*. Регистрационное имя (login name) и пароль (password) сообщаются администратором при регистрации нового пользователя. Только получив регистрационное имя и пароль, можно начинать сеанс работы.
Регистрационное имя пользователя может быть практически произвольным, а пароль должен отвечать определенным требованиям безопасности.
При выборе пароля рекомендуем придерживаться следующих правил:

- Пароль должен содержать не менее 6 символов.
- Пароль должен содержать как алфавитно-цифровые, так и специальные символы.
- Пароль должен содержать как прописные, так и строчные латинские буквы.
- Пароль следует менять раз в 1–2 месяца.
- Пароль не следует пересылать по электронной почте.

Начало работы — вход в систему, установка и изменение пароля

Приступая к работе на UNIX-ЭВМ, следует иметь в виду, что этот компьютер, скорее всего, является сервером и предназначен для обслуживания большого числа пользователей. Такой компьютер работает постоянно и выключить его имеет право только суперпользователь.

Сеанс работы в ОС UNIX начинается с ввода регистрационного имени и пароля. Пользователь должен быть зарегистрирован в системе. Вход в систему возможен, если на экране терминала отображается приглашение, вид которого может зависеть от того, с какой разновидностью UNIX собирается работать пользователь и от особенностей конкретной конфигурации.

Пароль при вводе не отображается на экране. После того, как введены регистрационное имя и пароль, система проверяет, имеется ли такой пользователь и правильно ли введен пароль. Если при вводе была допущена ошибка, вход в систему придется повторить еще раз. Если же регистрационное имя и пароль введены правильно, на экране появляется приглашение командной строки, вид которого зависит от установок, сделанных пользователем и от того, какой командный интерпретатор используется.

Сразу же после регистрации, а также периодически — раз в 1–2 месяца следует менять пароль. Для изменения пароля подается команда password. При выполнении этой команды система попросит ввести старый пароль, который используется для подтверждения того, что изменение пароля производится настоящим пользователем, а не посторонним человеком, случайно оказавшимся у терминала. После успешного ввода старого пароля последует запрос на ввод нового пароля. Новый пароль придется набрать еще один раз, когда последует запрос на повторный ввод пароля.

Справочная система man, xman, info

Вызов справки по системе UNIX осуществляется командой:

```
# man [ключи] [тема]
```

Здесь [тема] — название команды, библиотечной функции, системного вызова, файлового формата и т. д. Наиболее важными и полезными ключами команды man являются следующие:

- −k — при использовании этой опции можно указать ключевое слово или только его часть. Ключевым словом обычно является имя команды;
- −f — действие данного ключа аналогично ключу −k, однако выводиться в этом случае будут лишь файлы, содержащие указанное слово целиком.

Здесь и в дальнейшем необязательная часть команды будет размещаться в квадратных скобках.

Команда man использует фильтр для просмотра справочной страницы — more или less. «Прокрутка» файла справки в окне просмотра производится нажатием клавиши Пробел. Возможно также использование клавиш PageDown и PageUp. Построчное перемещение по файлу может производиться нажатием на клавишу Enter или клавиши управления курсором. Выход из просмотра производится нажатием клавиши Q.

В UNIX имеются и другие команды, позволяющие получить справку по той или иной команде или программе: apropos, whatis.

Кроме справочных страниц man в UNIX по некоторым программам имеется более подробная документация. Для просмотра этой документации используется программа info. Вызывается эта команда следующим образом:

```
# info [тема]
```

Здесь [тема] — название программы или команды. После вызова info появляется экран, в средней части которого отображается информация, относящаяся к данной теме, а в верхней части находится заголовок. В заголовке указано имя просматриваемого файла, а также параграфы, логически предшествующие данному параграфу или следующие за ним. Выход из программы info производится нажатием клавиши Q.

Завершение сеанса работы в ОС UNIX

После того, как пользователь выполнил намеченную работу, он должен завершить сеанс работы в операционной системе. Для завершения сеанса работы в текстовом режиме необходимо подать команду logout. Если при этом окажется, что в процессе работы было запущено несколько командных интерпретаторов один из другого, то для последовательного завершения всех вызванных интерпретаторов необходимо будет подать команду exit. Вместо команд logout и exit можно нажать комбинацию клавиш Ctrl/D.

ВНИМАНИЕ

> Запрещается прерывать сеанс работы с UNIX компьютером выключением питания компьютера или нажатием кнопки перезапуска компьютера. Если возникает необходимость в остановке компьютера или его перезагрузке, следует обратиться к системному администратору.

Команды в UNIX

Программы представляют собой *внешние* по отношению к оболочке команды. Команды, код которых находится в исполняемом файле оболочки, называются *встроенными* или *внутренними*. Наиболее важные из них — cd, set, unset, setenv, export. Первая команда предназначена для изменения текущего каталога. Каждая программа в ОС UNIX имеет свой текущий каталог. Следующие две команды предназначены для установки и отмены параметров самой оболочки. Выполнение этих действий также нельзя передать другим программам. Команды setenv и export изменяют «окружающую среду» как других программ, так и оболочки.

Если набранное пользователем имя команды является именем встроенной команды, она немедленно будет исполнена. Если же имя не совпадает с именами встроенных команд, оболочка выполняет поиск исполняемого файла программы с указанным именем.

Файлы в UNIX хранятся в каталогах. Каталоги могут содержаться в других каталогах, то есть быть вложенными. Имена каталогов отделяются друг от друга и от имен файлов символом / (slash). В именах файлов и каталогов могут содержаться любые символы, кроме косой черты. Имя текущего каталога — «.», вышестоящего — «..». Имя главного, *корневого* каталога файловой системы UNIX — /.

Каталоги, в которых оболочка ищет программу, перечислены в переменной окружения PATH. Программа должна находиться в одном из этих каталогов, иначе оболочка выведет сообщение о том, что команда не найдена.

Если программы нет в каталогах, перечисленных в пути поиска программ, для ее запуска следует указать абсолютное или относительное имя файла. Это имя обязательно содержит в себе косую черту. Если имя начинается с косой черты — это абсолютное имя файла, например, /bin/ls, /bin/ps, /usr/ucb/ps. Если имя начинается с другого символа, это относительное имя файла, определяемое относительно текущего каталога. Часть имени до первой косой черты, если, конечно, косая черта входит в состав имени файла, должна быть именем каталога, расположенного в текущем каталоге. Для явного указания текущего каталога следует использовать точку: ./test — программа test из текущего каталога, ./bin/clean, и bin/clean — указывают на одну и ту же команду clean из каталога bin, находящегося в текущем каталоге. В составе имени файла можно использовать имя вышестоящего каталога: ../../programs/test.

Во всех интерпретаторах после набора нескольких (или ни одного) символов можно воспользоваться функцией автоматического завершения набора имени файла. Если набранные символы соответствуют единственному имени файла, оно вставляется в командную строку полностью. После имени добавляется косая черта, если это имя каталога, или пробел, если это обычный файл. Данная функция называется *автоматическим дополнением имени файла до полного* (или *автодополнением*).

Во второй версии оболочки bash список возможных имен выводится сразу после нажатия клавиши табуляции. Клавиша табуляции работает в любом месте строки, даже если после курсора имеются другие символы. В этом случае только символы, расположенные от начала слова до курсора, будут рассматриваться как начальные буквы имени файла. Символы, располагающиеся после курсора, не будут учитываться при наборе имени файла и будут отодвинуты в конец строки после дополнения имени до полного.

Функция автоматического дополнения имени действует и применительно к именам команд ОС UNIX. Если набрать первые буквы команды и нажать клавишу табуляции, имя команды будет дополнено, а если символов недостаточно, на экран выводятся все возможные имена команд.

Повторно вызвать ранее введенную команду можно либо клавишами управления курсором ↑ и ↓.

При завершении сессии введенные команды записываются в специальный файл, из которого они считываются интерпретатором в начале новой сессии. Таким образом, пользователь может повторно исполнять команды, использовавшиеся в прошлых сессиях.

Переменные оболочки и переменные окружения

Кроме явного указания ключей в командной строке существует еще один способ передачи ключей и некоторых других данных программам. Для этого можно использовать *переменные окружения*. Пользователь может в оболочке определить переменные, которые будут автоматически передаваться («экспортироваться») каждой исполняющейся программе.

Пользователь может определять внутренние, *локальные переменные оболочки*. Локальные переменные оболочки не передаются вызываемым программам и не

влияют на их работу. Некоторые из локальных переменных влияют на работу самой оболочки.

Каждая переменная имеет имя и набор допустимых значений. Имя (идентификатор) переменной может содержать прописные и строчные буквы латинского алфавита, символ подчеркивания и цифры. Регистр букв в именах переменных различается. Имя переменной не должно начинаться с цифры. Значениями переменных являются строки символов.

Имена переменных окружения записываются *прописными* буквами, а имена локальных переменных оболочки — *строчными* буквами.

Простая локальная переменная в оболочке bash определяется командой:

```
# имя=значение
```

Для того чтобы простую переменную сделать переменной окружения, надо выполнить команду export:

```
# export имя1
```

Для сокращения записи пользователь может определить новые переменные прямо в команде export:

```
# export имя1=строка1
```

Использовать в любом месте командной строки значение любой переменной можно с помощью конструкции ${имя}. При интерпретации командной строки в нее будет подставлен текст, являющийся значением переменной.

Переменная PATH содержит путь поиска программ. Узнать стандартный путь поиска, который установлен при запуске оболочки сессии можно с помощью команды echo:

```
# echo $PATH
/bin:/usr/bin:/usr/X11R6/bin:/usr/local/bin
```

Если текущий каталог не включен в путь поиска, пользователь может добавить как его, так и другие каталоги к уже заданному пути поиска. В оболочке bash это делается так:

```
# export PATH=.:~/bin:$PATH
```

Для того чтобы убедиться в правильности выполнения команды полезно вывести новое значение переменной:

```
# echo $PATH
.:/home/alex/bin:/bin:/usr/bin:/usr/X11R6/bin:/usr/local/bin
```

Задания для практической работы

1. Зарегистрируйтесь в качестве пользователя вашего учебного UNIX-класса. Получите свой пользовательский идентификатор (регистрационное имя).
2. Войдите в систему и установите свой пароль, соблюдая приведенные здесь рекомендации.
3. Подайте команду ls с известными вам ключами и посмотрите, какие файлы находятся в вашем текущем каталоге. С помощью справочной системы man выясните, какие еще ключи можно использовать с командой ls.

4. С помощью справочной системы man выясните назначение команд cp и mv. С помощью команды info info познакомьтесь с возможностями программы просмотра документации info (если эта команда присутствует в системе).
5. Завершите сеанс работы подачей команд logout или exit.
6. Определите назначенную вам системным администратором при регистрации оболочку.
7. Определите, воспринимает ли оболочка команду exit в качестве синонима команды logout.
8. Определите, какие еще оболочки имеются в вашей UNIX-системе.
9. Определите, сохраняется ли список введенных ранее команд при завершении сессии и считывается ли он оболочкой в начале новой сессии.
10. Определите, сколько команд находятся в установленном пути поиска.
11. Сколько команд и какие начинаются на букву a?
12. Потренируйтесь в наборе имен файлов и команд с помощью функции дополнения имен.

Тема 2. Файловая система ОС UNIX

Как устроена файловая система ОС UNIX

Значительную часть времени пользователя занимает выполнение различных действий с файлами — их создание, преобразование, удаление и т. д. *Файл* — это именованная область памяти, в которой хранится некоторая информация и которая размещена на каком-либо носителе информации, например, жестком диске. Файл содержит информацию, характер которой зависит от его типа. Это может быть текстовый файл, содержащий обычный текст или двоичный файл, содержащий, например, машинные команды программы. В ОС UNIX понятие файла более общее. Можно считать, что файлом в ОС UNIX называют любой источник данных, которые могут быть считаны или объект, куда данные могут быть записаны. Таким образом, среди привычных текстовых или двоичных файлов оказываются клавиатура (источник данных) и дисплей (устройство, на которое производится запись текстовой и графической информации).
Количество файлов, размещенных на дисках компьютера, может быть очень большим. Для того, чтобы с файлами было удобно работать, следует их каким-то образом упорядочить. Файлы объединяют в группы по какому-то признаку, например, принадлежности определенному пользователю/ Эти группы называются *каталогами*. Каталоги организуются в иерархическую структуру (в иерархической

структуре объекты связаны отношениями подчинения), которую называют *файловой системой*.
В каталогах могут находиться другие каталоги (в этом случае они называются *подкаталогами*) и т. д. Каталог, содержащий другие каталоги считается по отношению к ним *родительским*. В результате возникает *иерархическое дерево каталогов*. «Корень» этого дерева, то есть каталог, содержащий все другие каталоги, являясь для них «родителем», называется *корневым* или *главным каталогом*. Корневой каталог иногда содержит небольшое число обычных файлов, как правило, системных.
Основными функциями файловой системы являются:

1. размещение файлов и их удаление,
2. выполнение операций чтения и записи в файлы,
3. изменение атрибутов файлов и некоторые другие.

В ОС UNIX файловая система контролирует доступ пользователей к файлам, благодаря чему реализуется многопользовательский характер системы. Файловая система UNIX обеспечивает доступ к периферийным устройствам компьютера, таким, как дисководы гибких дисков или компакт-дисков, модемы и другие.
Каждый файл имеет имя, которое используется для доступа к нему. Имя может определять положение файла в иерархическом дереве файловой системы. Допустимые имена определяются правилами, которые могут отличаться в разных операционных системах.
Файловая система ОС UNIX имеет один корневой каталог, его имя не связано с конкретным физическим устройством, на котором располагается файловая система.
Удаленный файл уже невозможно восстановить, поэтому, выполняя операцию удаления файла, советуем соблюдать осторожность, чтобы не потерять важные и необходимые данные.

Имена файлов в ОС UNIX

Полное имя файла состоит из двух частей пути и собственно имени. *Путь* указывает последовательность каталогов, в которой последним (если смотреть слева направо) является каталог, содержащий данный файл, предпоследним родительский по отношению к первому каталог, затем идет каталог следующего уровня и т. д. Имена каталогов отделяются друг от друга символом /. Полный или абсолютный путь начинается с указания корневого каталога, который обозначается символом /, например /home/student/session/good_by_physfac/. Относительный путь начинается с подкаталога текущего каталога, например

dean/students/hello_phys_fac/. Есть два стандартных обозначения точка «.», для текущего каталога и две точки «..» для каталога верхнего уровня (родительского каталога).
Имя указывает на конкретный файл и является его идентификатором. В системах BSD, Linux и других имя может иметь длину до 255 символов, включая произвольное количество суффиксов (“расширений”).
В именах файлов можно использовать любые символы из кодовой таблицы ASCII. Это, прежде всего, символы латинского алфавита, цифры, а также специальные символы. Система различает символы в верхнем и в нижнем регистрах, (например, имена cat и Cat относятся к разным файлам). Нельзя использовать в именах файлов символ /. В именах файлов ОС UNIX разрешено использовать управляющие символы. В таблице ASCII эти символы занимают первые позиции, в их число входят, например, управляющий символ звукового сигнала, символ перевода каретки и другие. При работе в оболочке bash можно ввести такую команду:
```
# cat readme > $'\015'
```
В этом случае будет создан файл, имя которого состоит из одного символа с восьмеричным кодом 015 (код символа «возврат каретки»).
В имени файла можно выделить две составные части. Первая, это *базовое имя*, а вторая — суффикс. *Суффиксом* называют часть имени файла, следующую за точкой. В отличие от таких операционных систем, как MS-DOS, MS Windows, интерпретацией содержимого файла занимается не операционная система, а прикладная программа, работающая с данным файлом, поэтому в ОС UNIX суффиксы в именах файлов не играют большой роли. В MS-DOS признаком исполняемого файла является один из трех суффиксов — .EXE, .COM или .BAT. В ОС UNIX исполняемые файлы не обязаны иметь имена с каким-либо стандартным суффиксом.
Файлы, имена которых начинаются точкой, содержат настройки для различных программ. Удаление, неправильное редактирование или повреждение таких файлов могут привести к некорректной работе программ.

Типы файлов ОС UNIX

В ОС UNIX имеется 6 типов файлов:

1. обычный файл;
2. каталог;
3. специальный файл устройства;

4. именованный канал;
5. ссылка;
6. сокет.

Атрибуты файлов

Любой файл, находящийся в файловой системе ОС UNIX, характеризуется набором атрибутов, описывающих его свойства.
Права доступа к файлу. ОС UNIX является многопользовательской операционной системой, поэтому она должна обеспечивать возможность ограничения доступа к файлам любого пользователя. Реализуется эта возможность с помощью определения прав доступа к файлу, которые определяются для трех категорий пользователей. Первая категория состоит из одного пользователя, являющегося собственником файла (user). Вторая — все пользователи, принадлежащие к той же группе, что и владелец файла (group). Третья категория — это все остальные пользователи (other). Для каждой категории отдельно устанавливаются права на чтение файла, на запись в него и на выполнение файла.
Для обозначения класса доступа имеется представление в виде строки символов. Отсутствие права обозначается символом «минус», а наличие прав следующими символами:

1. на чтение — символом r;
2. на запись — символом w;
3. на выполнение — символом x.

Суперпользователь имеет неограниченный доступ к файловой системе, поскольку при выполнении им различных операций с файлами операционная система не производит проверок соответствия его «полномочий» установленному для файла классу доступа.
Владелец файла и группа-владелец файла. Атрибутами любого файла являются имена его владельца и группы. При регистрации нового пользователя в системе администратор определяет его принадлежность определенной группе пользователей.
Если исполняемый файл запускается на выполнение, порожденный ей процесс получает те же права доступа к системным ресурсам, что и собственник файла.
SUID, SGID. Эти атрибуты устанавливаются для исполняемых файлов и позволяют изменить права пользователя на время выполнения соответствующих программ.

Обычный пользователь может хранить свои файлы только в своем домашнем каталоге. Этот каталог назначается в момент регистрации нового пользователя в системе и находится, как уже говорилось, в /home.
Все остальные каталоги являются системными и, как правило, допускают просмотр их содержимого, а также чтение ряда системных файлов. Запись в такие каталоги (кроме /tmp) запрещена.

Основные команды для работы с файлами

Команды вывода информации о файлах и файловой системе

Команда ls. Это команда вывода списка файлов, содержащихся в указанном каталоге:
ls [ключи] [каталог]
Команда pwd. Команда pwd выводит полное имя текущего каталога с указанием пути доступа к нему:
pwd
Команда df. Команда df сообщает объем дискового пространства как полный, так и занятый файлами, а также размер дискового пространства, доступный пользователям:
df [ключи] [*файл1 файл2…*]
Если данная команда используется без аргументов, выводится информация о доступном и используемом дисковом пространстве для всех смонтированных файловых систем. Размеры выводятся в килобайтах.
Команда du. Команда du сообщает размер дискового пространства, занятого файлами, указанными в качестве аргументов:
du [ключи] [*файл*]
а также каталогами и подкаталогами указанного каталога. Если файл не указан, сведения выводятся о текущем каталоге.

Перемещение по файловой системе

Команда cd. Команда cd предназначена для изменения текущего (рабочего) каталога:
cd [*каталог*]
Команда с указанием каталога делает этот каталог текущим. Если же каталог не указан, а определено значение переменной окружения HOME, текущим окажется каталог, указанный в HOME. Это домашний каталог пользователя. Посмотреть значение переменной HOME можно с помощью команды:

echo $HOME

Команды создания и удаления файлов

Обычные файлы создаются прикладными программами. Так, например, текстовые файлы могут быть созданы с помощью текстового редактора или вычислительной программы, производящей запись результатов своей работы в файл.

Команда mkdir. Для создания каталогов используется команда mkdir:

mkdir [ключи] *имя_каталога1 имя_каталога2 …*

Имя вновь создаваемого каталога не должно совпадать с именами уже существующих файлов.

Команда rmdir. Команда rmdir удаляет пустые каталоги:

rmdir [ключи] *каталог1 каталог2 …*

Каталог удаляется, только если он пуст, в противном случае выдается сообщение об ошибке.

Команда rm. Команда rm удаляет файлы, указанные в качестве ее аргументов. По умолчанию, она не удаляет каталоги, однако использование определенных ключей позволяет применять ее и к каталогам. Формат вызова команды:

rm [ключи] *файл1 …*

Команды изменения атрибутов файлов

Команда chmod. Утилита chmod позволяет установить права доступа к файлам:

chmod [ключи] {режим доступа} *файл*

Изменение прав доступа может заключаться в установке определенных прав в какой-либо триаде доступа и/или отмене таких прав. Триадой доступа называются три права доступа для какой-либо категории пользователей. Режим доступа задается в формате:

пользователь операция право_доступа

Поля следуют друг за другом без пробелов.

Поле пользователь определяет триаду доступа и может содержать следующие символы:

4.u владелец файла;

5.g группа-владелец файла;

6.o все остальные пользователи;

7.a все пользователи.

Поле операция показывает, как следует изменить право доступа:

8.+ добавить право доступа к тем правам, которые уже установлены;

9.– □ отменить указанное право доступа, оставив все остальные права неизменными;

10.= □ установить только указанные права доступа.

Поле право_доступа определяет, какое именно право доступа должно быть изменено:

11.r □ право на чтение из файла;

12.w □ право на запись в файл;

13.x □ право на выполнение файла.

Сделать командный файл a.sh исполняемым можно командой:
chmod a+x a.sh
Правом на выполнение такого файла будут обладать все категории пользователей.

Копирование и переименование файлов

Команда ср. Команда ср копирует файлы и каталоги. С её помощью можно копировать как отдельные файлы, так и их группы:
ср [ключи] *файл-источник целевой-файл*
ср [ключи] *файл-источник1… каталог*
В результате выполнения первой команды создается точная копия файла-источника (целевой-файл). В первом случае имя копии может отличаться от имени исходного файла, а во втором оно то же самое, однако копия располагается в другом каталоге. В первом случае у команды ср может быть только два аргумента, а во втором количество копируемых файлов может быть любым. Пример применения команды копирования ср:
ср Alexander Deanery/Dean
Команда mv. Команда mv переносит или переименовывает файлы:
mv [ключи] *файл_источник* целевой-*файл*
mv [ключи] *файл_источник1… каталог*
Если последним аргументом команды является имя каталога, указанные файлы переносятся в этот каталог. Исходные файлы при этом удаляются.

Утилита конкатенации cat

Утилита cat называется утилитой *конкатенации* (слияния) файлов. Ее действие заключается в том, что содержимое текстовых файлов, имена которых заданы в качестве аргументов команды, выводится в стандартный файл вывода:
cat [ключи] [*имя_файла1 имя_файла2…*]

Утилиты more и less

Утилиты more и less предназначены для просмотра больших файлов. Утилита less распространяется свободно, а more поставляется вместе с коммерческими ОС. Утилиту less можно считать усовершенствованной версией утилиты more. У нее больше возможностей, в том числе при работе в интерактивном режиме.

Утилиты more и less являются фильтрами, поэтому допустимо два типа вызовов:

more [ключи] *имя_файла*
less [ключи] *имя_файла1…*
или
программа | more [ключи]
программа | less [ключи]

Архивирование и сжатие файлов

В операционных системах UNIX раздельно существуют программы-архиваторы и программы-компрессоры. Исключением является утилита zip, совмещающая в себе обе функции. К архиваторам относится утилита tar. Существует несколько программ-компрессоров, использующих разные методы сжатия данных. Наиболее известными являются утилиты compress и gzip. Утилита gzip обеспечивает более высокую степень сжатия и свободно доступна в сети Интернет в исходных кодах и в виде исполняемых файлов для различных платформ. Обычный способ создания архивов заключается в последовательном применении архиватора и компрессора, например, утилит tar и gzip. Другим способом является применение GNU-версии утилиты tar с соответствующими ключами. Для работы с наиболее известными архивами MS-DOS имеются утилиты zip, unzip, unarj.
Архиватор— это программа, предназначенная для создания и обслуживания *архивов*. Архив представляет собой группу файлов, объединенных в один файл, который позволяет сохранять содержимое файлов вместе с характерной информацией о них. Эта информация включает — имя файла, идентификатор владельца, статус файла, временные отметки. При помощи архиватора в архив можно записать целое дерево каталогов с сохранением его структуры. Естественным применением архиваторов является создание резервных копий файловых систем.
Компрессор — это утилита, предназначенная для сжатия файлов. В мире DOS/Windows функции архиватора и компрессора обычно выполняет одна программа. Например, широко известные утилиты WinZip или WinRAR могут работать как чистый архиватор или как комбинация архиватора с компрессором, в зависимости от выбранной

степени сжатия. В мире UNIX функции архивирования и сжатия файлов традиционно выполняют разные утилиты. Обычно сначала при помощи архиватора создается архивный файл, который затем сжимается при помощи программы-компрессора.

Утилита tar

Утилита tar предназначена для выполнения различных операций с архивами — создания архива, просмотра архива, извлечения файлов из архива и т. д.:

tar [ключи] [*имя_файла*…]

Аргументы *имя_файла* задают имена файлов и каталогов, которые обрабатываются данной операцией. Если аргумент *имя_файла* является именем каталога, операция применяется ко всем его подкаталогам. К имени архивного файла обычно добавляется суффикс .tar.

Утилита tar может выполнять над архивами следующие операции:

14.операция –c создает новый архив и помещает в него указанные файлы. Например, команда:

tar –cvf archive.tar file1 file2 file3

tar cvf archive.tar file1 file2 file3

создаст новый архив archive.tar, выводя в стандартный вывод имена помещаемых в архив файлов file1, file2, file3.

15.операция –x извлекает из архива указанные файлы. Например, команда:

tar –xf archive.tar file1

извлекает файл file1 из архивного файла archive.tar. Если имя извлекаемого файла не задано, извлекаются все файлы в порядке их расположения в архиве — первый помещенный в архив извлекается первым.

16.операция –t выводит в стандартный вывод список файлов, содержащихся в указанном архиве:

tar –lf archive.tar

17.операция –r добавляет к существующему архиву указанные файлы, помещая их в конец архива. Например, команда:

tar —rf archive.tar file1

добавляет файл file1 в конец архива archive.tar.

18.операция —delete удаляет из архива файлы, перечисленные в командной строке:

tar —delete —file=archive.tar *файл1 файл2*...

Размер архива, создаваемого утилитами tar или cpio, даже несколько больше суммарного размера архивируемых файлов, так как помимо собственно данных файлов содержит учетную информацию. Поэтому архивный файл, как правило, сжимается утилитой-компрессором. В UNIX существует несколько программ, предназначенных для сжатия файлов. К их числу относятся утилиты compress, gzip, zip, bzip2, compact, pack, использующие различные методы сжатия данных. Каждая из перечисленных программ осуществляет сжатие данных на том же месте, то есть не создает новый, сжатый файл, а преобразует к сжатому виду содержимое существующего файла.

Утилиты compress, uncompress, zcat

Утилита compress входит в состав всех распространенных версий UNIX, как коммерческих, так и свободно доступных. Команды семейства compress имеют следующий синтаксис:

```
# compress [ключи] [–b bits] [file ...]

# uncompress [ключи] [file ...  ]
```

Сжатый файл получает новое имя путем добавления к исходному имени суффикса .Z.
Утилита uncompress просматривает список своих аргументов file и осуществляет декомпрессию всех файлов с суффиксом .Z. Суффикс .Z в именах файлов отбрасывается. Восстановленные файлы сохраняют атрибуты сжатых файлов.

Утилиты gzip, gunzip, zcat

Утилита gzip обеспечивает более высокую степень сжатия по сравнению с утилитой compress. Командная строка для вызова утилит семейства gzip имеет следующий синтаксис:

```
# gzip [ключи] [–S suffix] [file ... ]

# gunzip [ключи] [–S suffix] [file ... ]
```

Сжатый файл получает новое имя путем добавления к исходному имени суффикса .gz.
Утилита gunzip может осуществлять преобразование к исходному виду сжатых файлов, полученных в результате работы одной из программ gzip, compress, zip, pack. Она просматривает список своих аргументов file и восстанавливает в первоначальном виде все файлы, имеющие соответствующий внутренний формат и одно из расширений — .gz, –gz, .z, –z, _z, .Z, .tgz, .taz. Перечисленные суффиксы в именах файлов

отбрасываются. Восстановленные файлы сохраняют атрибуты сжатых файлов.

Утилиты bzip2, bunzip2

Еще одно семейство утилит образуют программы bzip2, bunzip2, обеспечивающие более высокую степень сжатия файлов, чем gzip:

```
# bzip2 [ключи] [file1…]

# bunzip2 [ключи] [file file2…]
```

Обработанные файлы получают новое имя добавлением к исходному имени суффикса .bz2.

Текстовый редактор pico

Текстовый редактор pico— это простой редактор, предназначенный для работы в текстовом режиме. Он известен благодаря своей интеграции с почтовой программой pine. Запускается pico командой:

pico [ключи] [*имя_файла*]

Выход из редактора осуществляется нажатием клавиш Ctrl+X.

При запуске редактора имя файла указывать не обязательно. Если оно задано, pico пытается открыть указанный файл для редактирования, а если это не удается, создает новый файл. Если имя файла не задано, создается безымянный буфер, а файлу при сохранении на диск должно быть присвоено имя.

Экран редактора. Экран редактора состоит из нескольких частей. Верхняя строка — это *строка статуса*. Она выделяется инверсной подсветкой и содержит имя редактируемого файла, его статус (изменялся ли файл с момента его последней записи на диск) и краткие сведения о самом редакторе. Далее идет *рабочая область* — окно, в котором отображается редактируемый файл и производится редактирование. Затем следует строка, в которой выводятся сообщения редактора: запросы на подтверждение записи файла на диск и т. д. В этой же строке вводятся пользовательские команды. В нижней части экрана выводится список комбинаций клавиш, связанных с основными командами редактора.

Ввод текста и команд. Ввод текста осуществляется нажатием соответствующих клавиш. Команды pico являются Ctrl-комбинациями. Ctrl-комбинация представляет собой одновременное нажатие клавиши Ctrl и какой-нибудь другой клавиши.

Команды перемещения по тексту в редакторе pico:

- Ctrl+P или 1 — переход к предыдущей строке;

- Ctrl+F или 1 — переход на одну позицию вправо;
- Ctrl+N или 1 — переход к следующей строке;
- Ctrl+B или 1 — переход на одну позицию влево;
- Ctrl+A или Home — переход в начало строки;
- Ctrl+E или End — переход в конец строки;
- Ctrl+V или Page Down — переход вперед на одну страницу;
- Ctrl+Y или Page Up — переход назад на одну страницу;
- Ctrl+C — определить текущее положение курсора.

Удаление символов. Удаление символов производится с помощью клавиш Delete и Backspace. Если нажатие этих клавиш не приводит к желаемому результату, можно использовать комбинацию Ctrl+D, которая аналогична Delete.
Удаление и восстановление строк. Удаление строки производится с помощью клавиш Ctrl+K. В зависимости от настройки редактора удаляется либо вся строка, в которой находится курсор, либо часть строки от указателя до конца строки. Удаленная строка записывается в регистр редактора (аналог буфера обмена MS Windows) и может быть вставлена в любое место текста командой Ctrl+U. Строка в буфере не удаляется до нового выполнения команды Ctrl+K. Если последовательно удаляются несколько строк командами Ctrl+K, то все строки запоминаются в регистре, образуя единый блок. Важно, чтобы между удалениями строк пользователь не выполнил какие-либо иные команды, даже простое перемещение по тексту
Операции с блоками. Для выполнения любой операции с блоком текста его необходимо вначале выделить. Выделение блока начинается нажатием клавиш Ctrl+^, которые устанавливают границу блока в том месте, где стоит курсор. Затем курсор перемещается в конец блока или в его начало. Текст, находящийся в буфере, будет выделен инверсной подсветкой. Удаление блока производится с помощью команды Ctrl+K. Блок при этом перемещается в единственный регистр редактора и хранится там до тех пор, пока он не будет замещен новой порцией удаленного текста. Текст, находящийся в регистре, можно вставить в редактируемый текст командой Ctrl+U.
Поиск по образцу. Поиск по образцу выполняется с помощью команды Ctrl+W. При этом редактор попросит ввести образец для поиска. Поиск производится, начиная с текущего положения курсора и до конца текста. Регистр букв в образце не различается.

Проверка правописания. Проверка правописания английского текста начинается при нажатии клавиш Ctrl+T. При выполнении проверки специальной программой-*корректором* производится сравнение слов, содержащихся в текстовом файле, с их образцами из системного словаря. В ходе такой проверки можно исправить случайно допущенные опечатки, что особенно полезно при использовании pico для подготовки электронных писем в почтовом пакете pine.
Операции с файлами. Сохранить на диске редактируемый файл можно командой Ctrl+O. Для сохранения с последующим выходом из редактора используется команда Ctrl+X. При записи буфера на диск необходимо подтвердить свое намерение его сохранить и указать имя файла, в который буфер будет записан. С помощью команды Ctrl+R в редактируемый файл можно вставить другой файл. Эта же команда может использоваться для загрузки в редактор нового файла. При нажатии клавиш Ctrl+R редактор попросит ввести имя загружаемого файла. Если имя не вводить, pico даст возможность перейти в режим файлового обозревателя, когда на экран выводится список файлов, содержащихся в текущем каталоге. В режиме файлового обозревателя допускается перемещение между файлами и каталогами.
Отмена редактирования. Редактор pico не позволяет отменить неправильно произведенное редактирование.

Редактор Emacs

Версии emacs имеются для разных операционных систем. Запуск редактора производится командой:
emacs [ключи] [*имя_файла*]
Интерфейс и стиль работы с emacs зависит от того, в каком режиме был он вызван. В текстовом режиме используются командные последовательности и мышь. В командных последовательностях используются клавиши Esc и Meta («метаклавиша»). На некоторых терминалах клавиша Meta имеется, однако чаще в качестве ее заменителя используется клавиша Alt или клавиша Esc. В этом случае запись Meta+Q следует понимать как Esc Q.
После вызова emacs на экране появляется рабочее окно редактора, которое занимает весь экран. При запуске emacs в X Window открывается собственное графическое окно. Рабочее окно emacs состоит из *меню* (верхняя строка), *области редактирования*, в которую выводится «видимый» участок буфера, *строки статуса* (вторая снизу) и *минибуфера* — самой нижней строки экрана. В строке статуса отображается информация о состоянии буфера (изменялся ли он с момента последней записи на диск) и режиме работы редактора.

Минибуфер используется emacs для вывода своих сообщений и для ввода команд пользователя.
В нижней части экрана располагается область отображения, в которой выводятся подаваемые команды. Ввод аргумента или имени функции завершается нажатием клавиши Enter. До нажатия клавиши Enter текст в этой строке можно редактировать. Отменить действие не полностью набранной команды, которая состоит из нескольких клавиш или требует ввода данных в минибуфер, можно нажатием клавиши Ctrl+G.
Команды emacs перемещения по файлу:

- Ctrl+F → Перемещение на одну позицию вправо
- Ctrl+B Перемещение на одну позицию влево
- Ctrl+N ↓ Перемещение на следующую строку
- Ctrl+P ↑ Перемещение на предыдущую строку
- Ctrl+A Home Перемещение в начало строки
- Ctrl+E End Перемещение в конец строки
- Meta+< Перемещение в начало файла
- Meta+> Перемещение в конец файла
- Meta+G Перейти к строке с заданным номером
- Ctrl+X = Определить положение курсора

Отмена операций редактирования. Отмена действия последней команды производится с помощью командной последовательности Ctrl+X U. Указатель при этом переходит в положение, в котором он находился перед выполнением отмененной команды. Повторное применение любой из этих комбинаций клавиш отменяет предпоследнюю команду и т. д. Количество команд, для которых имеется возможность отмены, определяется размером специальной области, в которой хранится история команд редактирования.
Блоки. Для того, чтобы выделить блок текста, необходимо установить маркер на одной границе блока и переместить указатель на другую его границу. Далее используется командная последовательность Ctrl+Пробел или Ctrl+@ (set-mark-command). Отказаться от установки маркера можно командой Ctrl+G. В режиме Transient Mark блок выделяется инверсной подсветкой. Перейти в этот режим можно с помощью команды Meta+X transient-mark-mode. Повторение данной команды отменяет режим. В разных буферах положения маркера независимы.

В emacs можно работать с прямоугольными блоками текста. Для того, чтобы выделить такой блок, необходимо установить маркер сначала в левом верхнем, а затем в нижнем правом углу блока. Отмеченный блок интерпретируется как строковый или как прямоугольный в зависимости от того, какие команды подаются.
Команды редактирования блоков.
Ctrl+W Удаление строкового блока

(kill-region)

Ctrl+X R S Запись строкового блока в регистр

(copy-to-register)

Возможность вставки предварительно удаленного текста может быть использована для перемещения блоков текста из одного места текстового файла в другое. Команды вставки:
Ctrl+Y Вставить последний удаленный текст

(yank)

Поиск и замена. Рассмотрим команды поиска по образцу и замены. В emacs используются два типа поиска — пошаговый и обычный. В первом случае поиск начинается еще до завершения ввода образца целиком. После ввода первого символа указатель перемещается к первому найденному символу. При вводе каждого последующего символа emacs ищет уже введенную часть образца и перемещает указатель в соответствующее место файла. Поиск прекращается при нажатии клавиши Enter. Откорректировать образец для поиска можно с помощью клавиши Delete. Выйти из режима поиска (если, например, требуемый образец уже найден) можно с помощью команды Ctrl+G или с помощью любой другой команды, не являющейся командой поиска. Во втором случае (обычный поиск) поиск начинается только после ввода образца. Найденный образец может выделяться инверсной подсветкой.
Ctrl+S Пошаговый поиск по направлению к концу файла

(isearch-forward)

Ctrl+R Пошаговый поиск по направлению к началу файла

(isearch-backward)

Ctrl+S Ctrl+S Продолжение поиска с использованием последнего образца

Ctrl+S Enter *строка* Enter Обычный поиск строки по направлению к концу файла

Ctrl+R Enter *строка* Enter Обычный поиск строки по направлению к началу файла

Если в образце используются буквы из верхнего регистра, emacs при поиске будет различать регистр букв, а если только буквы из нижнего регистра — регистр не будет учитываться.

Продолжить поиск в направлении назад можно с помощью команды Ctrl+R. При выполнении пошагового поиска можно скопировать в образец слово, которое находится после указателя — для этого используется

Основной командой поиска и замены в emacs является команда поиска и замены с запросом Meta+X query-replace. Областью действия этой команды может быть буфер в целом или выделенный блок текста. Поиск и замена с запросом применяются в том случае, когда требуется выборочно заменить включения в текст образца. Это может быть сделано с помощью команды:

Meta+% *строка поиска* Enter *строка замены* Enter

или

Meta+X query-replace Enter *строка поиска* Enter *строка замены* Enter

Операции с файлами. Для того, чтобы загрузить для редактирования файл, необходимо ввести соответствующую команду и ввести имя файла. Для ввода имени файла используется минибуфер, при этом можно использовать возможность автоматического дополнения имени файла. При выполнении операций с файлами по умолчанию используется имя того каталога, из которого был загружен последний файл. Для того, чтобы определить это имя, используется команда Meta+X pwd. Изменить текущий каталог можно командой Meta+X cd с последующим вводом имени каталога. При чтении файла имя текущего каталога появляется в минибуфере, что дает возможность отредактировать это имя или просто вспомнить, какой каталог является текущим. Основные операции с файлами и каталогами перечислены:

Ctrl+X Ctrl+F Загрузить файл. Для ввода имени файла используется минибуфер. При выполнении этой команды emacs создает буфер, копирует в него файл и отображает содержимое буфера на экране. В случае успешной загрузки файла его содержимое появляется в окне редактирования, если же при загрузке возникли какие-либо проблемы (обычно это неправильно указанное имя файла), в области отображения появится сообщение об ошибке

(find-file)

Ctrl+X Ctrl+R Загрузить файл в режиме просмотра. Редактирование в этом режиме запрещено

(find-file-read-only)

Ctrl+X Ctrl+S Записать текущий буфер на диск

(save-buffer)

Ctrl+X S Записать буферы на диск. При выполнении этой команды emacs выводит запрос о том, следует ли сохранить содержимое каждого буфера. Варианты ответных действий здесь такие же, как при выполнении операции поиска и замены

(save-some-buffers)

Meta+~ Считать, что буфер не редактировался. Эта команда может оказаться полезной в том случае, когда пользователь изменил содержимое буфера, но сохранять результаты этого редактирования не собирается. В этом случае есть вероятность ошибочного сохранения буфера, например, при завершении работы с emacs. Данная команда позволяет избежать этой опасности
(not-modified).

Ctrl+X Ctrl+W Записать текущий буфер в файл, имя которого указывается пользователем

(write-file)

Meta+X set-visited-file-name Изменить имя файла из текущего буфера

Ctrl+X Ctrl+C Завершить работу с emacs с выводом запроса о сохранении результатов редактирования

При загрузке файла emacs создает новый буфер и присваивает ему такое же имя, что и имя файла. Если ранее уже был загружен файл с таким именем, в имени буфера появится <2>. Если делается попытка загрузить тот же самый файл, emacs просто перейдет в соответствующий буфер, проверив предварительно, не изменилось ли содержимое файла со времени его последнего сохранения. Для создания нового файла достаточно загрузить файл с несуществующим (то есть новым именем). При этом создается пустой буфер. Файл на диске будет создан только после записи буфера на диск.
Emacs создает резервные копии файлов, что позволяет отказаться от ошибочно проведенного редактирования даже после записи отредактированного файла на диск. Резервная копия файла имеет имя, отличающееся от имени настоящего файла только символом ~ в конце имени.
Справка. Справку по работе с emacs можно вызвать нажатием клавиш Ctrl+H или F1. Командная последовательность Ctrl+H K, за которой следует ключевое слово, позволяет получить информацию по заданной теме (обычно это команда emacs). Например, командная последовательность Ctrl+H K Ctrl+N выведет справку о команде Ctrl+N.

Двойное последовательное нажатие Ctrl+H выведет справочную информацию о работе со справочной системой emacs.

Последовательность Ctrl+G отменяет выполнение введенной команды. Emacs запрещает использовать при работе с минибуфером команды, активизирующие минибуфер. Это делается для того, чтобы предотвратить рекурсивное использование минибуфера.
При вводе команд можно использовать автоматическое дополнение, которое заключается в том, что достаточно набрать часть команды, а emacs попытается самостоятельно дополнить набранную часть до полной. Функция дополнения связана с одной из клавиш: Tab, Enter или Пробел. Клавиша ? позволяет просмотреть список возможных вариантов дополнения. Так, например, если командная последовательность Meta+X использует минибуфер для ввода команды, клавиша ? позволит просмотреть перечень имеющихся команд, имена которых совпадают с введенной частью команды. Используя этот прием вместо Meta+X insert-buffer можно использовать последовательность Meta+X Insert Пробел B. Регистр букв при автоматическом дополнении учитывается.

Задания для практической работы

1. Создайте в своем домашнем каталоге каталоги piggy_1 и piggy_2/piggy_3. Скопируйте в первый из этих каталогов файлы wild_wolf1 и wild_wolf2, а во второй файл wild_cat.
2. Удалите каталоги, созданные в упражнении 1 вместе с их содержимым.
3. Создайте в своем домашнем каталоге каталог Welcome, открытый для записи и чтения всем пользователям.
4. Создайте в своем домашнем каталоге каталог Welcome? (со знаком вопроса в конце имени) и сделайте так, чтобы все пользователи кроме вас могли записывать в него файлы, но не могли бы просматривать его содержимое.
5. Создайте текстовый файл небольшого размера. Это можно сделать, например, следующим образом. Подайте команду:

   # cat > secret_service

   После этого появится приглашение в виде угловой скобки. Наберите любой текст. Нажмите клавиши Ctrl+D. Присвойте этому файлу право на чтение только своей группе. Предложите пользователю, не принадлежащему вашей группе прочитать этот файл. Если ему это удастся сделать, повторите материал о правах доступа к файлам.
6. Создайте небольшой текстовый файл. Измените его атрибуты так, чтобы другой пользователь смог его удалить.

7. Создайте небольшой текстовый файл. Создайте символическую ссылку на него. Затем создайте две-три жесткие связи из текущего каталога и других каталогов. Сравните размеры символической ссылки и жестких связей. Объясните различие. Удалите исходный файл. Что произошло со связями обоих типов? Объясните.
8. Создайте с помощью редактора pico текстовый файл. Проверьте, не были ли при наборе допущены ошибки.
9. Объедините с помощью утилиты cat файлы, указанные преподавателем в единый файл и поместите его в созданный вновь каталог с именем Love Songs (с учетом пробела в имени каталога).
10. Создайте пустой файл empty. При помощи утилиты tar создайте новый архив empty.tar, содержащий единственный файл empty. Сравните размеры файлов empty и empty.tar. Объясните разницу в размерах двух файлов. Добавьте в архив empty.tar еще один пустой файл empty1. Как изменился размер архивного файла? Создайте архивный файл, содержащий один пустой файл, используя коэффициент блокирования 1:
    # tar –cf zero.tar –b 1 empty
    Сравните размер архивного файла zero.tar с размером архивного файла empty.tar. Объясните разницу в размерах.
11. Напишите команду для создания при помощи утилиты tar сжатого архивного файла, содержащего все графические файлы формата GIF из каталога /usr.
12. Напишите последовательность команд для создания архива tar, содержащего все обычные файлы текущего каталога, имена которых начинаются с точки.

Тема 3. Процессы

Что такое процесс

В ходе сеанса работы пользователь запускает процессы. Некоторые процессы выполняются «от имени» операционной системы. Многозадачная операционная система управляет процессами, приостанавливая/завершая выполнение одних, активизируя/создавая другие. Пользователь может управлять только теми процессами, которые он запустил на выполнение.

Операционная система создает процесс, когда пользователь запускает программу на исполнение. Нельзя отождествлять процесс и программу, которая исполняется в данном процессе. Программой являются машинный код и начальные данные, содержащиеся в исполняемом файле на диске или скопированные ОС для исполнения в оперативную память. Для каждого процесса ОС создает дополнительный набор данных, который называется *средой выполнения процесса*. Данные из этого набора называются *атрибутами процесса*. Среди них: переменные окружения, текущий каталог, стандартные файлы ввода, вывода и ошибок и другие. Процесс в ОС UNIX — это программа и связанные с ней системные данных. Обычно для работы одной программы запускается один процесс, но одна программа может породить и несколько процессов.

Процесс использует ресурсы компьютера: аппаратные (процессор, оперативную память, периферийные устройства и т. д.) и программные (программы, системные

таблицы и т. д.). Управление процессами со стороны ОС заключается в распределении и предоставлении им необходимых ресурсов.

Новый процесс может быть порожден только другим процессом, работающим в системе. Процесс, запускающий на выполнение другой процесс, называется *родительским* процессом, а порожденный им процесс — *дочерним*.

При завершении любой процесс передает операционной системе *статус своего завершения* — целое число от 0 до 255 включительно. Программа, которая завершается *нормально*, возврщает операционной системе статус, равный нулю. Если программа завершается *с ошибкой*, возвращается ненулевое значение. В оболочке bash статус завершения содержится в специальном параметре ?.

Оболочка bash различает несколько ошибочных ситуаций. Если программа *не найдена* в пути поиска программ, оболочка возвращает статус 127. Если файл с указанным именем найден в пути поиска, но *не является исполняемым*, возвращается статус 126. Если выполнение программы *прерывается сигналом*, возвращается статус 128+N, где число N — номер сигнала.

Пример определения статуса завершения команд в bash:

```
# ls l*
ls.txt
# echo $?
0
```

Команда ps

Команда ps выводит информацию о запущенных процессах. Вывод оформлен в виде таблицы. В первой строке содержатся заголовки колонок таблицы, в последующих строках выводятся сведения о каждом процессе. В следующем примере приведен пример вызова программы и выводимая информация в двух различных форматах:

```
% ps u
USER       PID %CPU %MEM   SZ  RSS    TTY STAT    STIME  TIME COMMAND
komolkin 16302  0.0  1.0  256  628  pts/4 A     23:59:47  0:00 -tcsh
komolkin 14796  0.0  0.0  168  268  pts/4 A     00:00:16  0:00 ps u
```

Атрибуты и параметры процессов

Каждый процесс характеризуется набором *атрибутов* — присущих этому процессу признаков, отличающих его от других процессов. Кроме того, процесс характеризуется параметрами, которые могут меняться в течение жизненного цикла процесса. Большинство атрибутов процесса наследуются от родительского процесса, некоторые атрибуты могут быть установлены операционной системой, а некоторые могут быть изменены самим процессом в ходе его выполнения. Рассмотрим наиболее важные для пользователя атрибуты и параметры процессов.

Идентификатор процесса (PID, Process IDentifier). Идентификатор процесса является целым положительным числом. Каждый процесс в системе имеет уникальный идентификатор. Операционная система управляет процессами, используя идентификатор процесса.

Идентификатор родительского процесса (PPID). Этот атрибут процесс получает во время своего запуска и может использовать его для получения информации о статусе

родительского процесса или для отправки ему сигналов. Команда ps с ключом –f выводит информацию об идентификаторе родительского процесса в колонке PPID.

Реальный и эффективный идентификаторы пользователя и группы (UID, GID, EUID и EGID). Реальные идентификаторы совпадают с идентификаторами пользователя, который запустил процесс, и группы, к которой он принадлежит. Реальные идентификаторы наследуются дочерним процессом и не могут быть изменены. Права доступа процесса к ресурсам ЭВМ определяются эффективными идентификаторами. Пользовательские процессы имеют те же права доступа, что и пользователь. Однако из этого правила есть исключение. Если у исполняемого файла установлен атрибут SUID, при запуске программы эффективный идентификатор пользователя устанавливается равным идентификатору владельца файла. Права доступа такого процесса к ресурсам компьютера совпадают с правами доступа владельца файла. То же относится и к эффективному идентификатору группы — если файл имеет атрибут SGID, эффективный идентификатор группы процесса будет равен идентификатору той группы, которой принадлежит файл. Таким образом, пользователь, запустивший программу с установленными атрибутами SUID или SGID, получает права доступа к ресурсам компьютера, отличные от тех, что он имеет обычно.

Пример:

```
# ps -o pid,ppid,ruser,rgroup,user,group,comm
  PID  PPID    RUSER   RGROUP     USER    GROUP COMMAND
 9324 16200 komolkin    staff komolkin   system ps
16200 14662 komolkin    staff komolkin    staff tcsh
```

При вызове ps в данном примере используются имена колонок: ruser и rgroup — для вывода UID и GID, и user и group — для EUID и EGID.

Открытые файлы. Кроме стандартных файлов ввода, вывода и ошибок процесс может открыть другие файлы. Дочерний процесс наследует все файлы, открытые родительским процессом. Если, например, перед запуском дочернего процесса родительский процесс перенаправил стандартный файл вывода в файл на диске, вся информация, направляемая дочерним процессом в стандартный файл вывода, будет записана на диск.

Управляющий терминал. Управляющим терминалом является терминал, на котором запущена оболочка сессии. Этот атрибут наследуется. Он установлен только для тех программ, которые запущены пользователем во время сессии.

Некоторые системные программы не связаны с каким-либо управляющим терминалом. Они называются *демонами*. Управляющий терминал теряют программы, которые пользователь оставляет работать после окончания сессии.

Программа ps выводит названия управляющих терминалов для процессов в колонке TTY. По умолчанию ps выводит информацию только о тех процессах, которые связаны с ее управляющим терминалом. Для вывода информации о процессах, связанных с другими терминалами, их имена следует перечислить после ключа –t. Информацию о процессах, связанных со всеми терминалами, можно получить с помощью ключа a или –a. Вывести список всех процессов можно с помощью ключей ax. Если у процесса нет управляющего терминала, в колонке TTY выводится знак вопроса.

Приоритет и относительный приоритет. Планирование выполнения процессов в ОС UNIX основано на системе *приоритетов*. Приоритет процесса — переменная величина, которая вычисляется ОС в момент выбора процесса, которому должен быть предоставлен очередной «квант» процессорного времени. Приоритет

изменяется в зависимости от ряда факторов. Среди этих факторов — *относительный приоритет процесса*, время ожидания запуска, текущее состояние процесса и некоторые другие. Единственным фактором, на который может повлиять пользователь, является относительный приоритет (nice). Параметр nice в Linux имеет значения от –20 до +20. Большему значению nice соответствует меньший приоритет. Этот атрибут наследуется, то есть дочерний процесс будет иметь такой же относительный приоритет, что и родительский процесс.

Рядовой пользователь системы может только понижать приоритет своих задач, то есть увеличивать значение параметра nice. Понижать значение nice может только суперпользователь. С повышенным приоритетом работают важнейшие процессы, которые большую часть времени ожидают внешних событий и почти не потребляют процессорное время.

Пример:

```
# ps -eo pid,pri,nice,stat,cmd
  PID PRI  NI STAT CMD
    1  39   0 S    init
    6  59 -20 SW<  [mdrecoveryd]
  432  23  16 SN   xload -nolabel -geometry 32x20+0+0 -bg grey60 -update 5
  476  24   0 R    ps -eo pid,pri,nice,stat,cmd
```

Ключ –e команды ps позволяет вывести информацию обо всех процессах. Имена колонок PRI, NI и STAT обозначают соответственно приоритет, относительный приоритет и состояние процесса. Буква S в колонке STAT обозначает, что процесс находится в состоянии ожидания и не потребляет процессорное время. Меньшее значение в колонке PRI соответствует более высокому приоритету процесса. В колонке STAT у процессов с отрицательными значениями nice выводится символ <. В колонке STAT у процессов с пониженным приоритетом выводится символ N.

Наивысший приоритет имеют пользовательские программы, даже если их относительный приоритет понижен. Системные программы основную часть времени находятся в состоянии ожидания.

Текущий каталог. Атрибутом процесса является его текущий или рабочий каталог. Каждый процесс имеет свой текущий каталог, который он может изменить независимо от других процессов.

Время исполнения. ОС UNIX определяет *пользовательское, системное и реальное* время исполнения процесса. Пользовательское время — это время, затраченное центральным процессором на исполнение кода программы. Системное время — это время, затраченное процессором на обработку системных вызовов, операции ввода/вывода, пересылка данных между ЭВМ и т. д. Сумма пользовательского и системного времени равна полному процессорному времени, затраченному на исполнение программы. Реальное время — это время, прошедшее с момента запуска программы.

У программы ps есть специальные колонки, которые выводятся при ее использовании с ключом –o и которые позволяют вывести степень загрузки процессора (PCPU или %CPU), процессорное время (TIME или CPUTIME), время запуска программы (STIME) и реальное время, прошедшее с момента запуска (ETIME):

```
# ps -o pid,pcpu,time,etime,stime,cmd -p 353
  PID %CPU     TIME  ELAPSED STIME CMD
  353  4.2 00:01:06 00:26:11 15:37 /etc/X11/X
```

В данном примере процесс использовал 1 минуту 6 секунд процессорного времени.

Время исполнения программы можно определить, если запустить ее с помощью команды time. Аргументом является имя программы. В результате выполнения в файл стандартных ошибок выводятся три времени исполнения, а также некоторые другие параметры. В качестве примера приведем определение времени работы редактора jed, с помощью которого пользователь редактировал файл force.p:

```
% /bin/time jed force.p
real 82.85
user 1.14
sys 0.23
```

Переменные окружения. Переменные окружения доступны программам для чтения. Механизм переменных окружения служит для передачи программам параметров, которые не меняются от вызова к вызову и устанавливаются пользователем командами оболочки export или setenv.

Размер программы. Важным параметром процесса является объем занимаемой им памяти. Для размещения процессов в оперативной памяти ЭВМ в ОС UNIX применяются механизмы свопинга и виртуальной памяти. Свопингом называют выгрузку на жесткий диск процесса.

Часть процесса или весь процесс (машинный код и сегмент данных) могут оказаться выгруженными на жесткий диск, не занимая место в оперативной памяти компьютера. Каждый процесс характеризуется значениями виртуального размера и размера резидентной части. Виртуальный размер показывает полный объем памяти, необходимый для размещения всего процесса. В оперативной памяти располагается только резидентная часть, которая содержит часто используемые фрагменты кода программы и сегмента данных. Не использовавшиеся в течение некоторого времени фрагменты могут выгружаться на диск для того, чтобы освободить оперативную память для других процессов.

Объем используемой памяти можно узнать с помощью команды ps и ключа –l или при указании колонок SZ, VSZ, RSS и PMEM или %MEM после ключа –o. В колонке SZ выводится виртуальный размер программы в страницах оперативной памяти (обычно страница имеет объем 2 или 4 Кбайт), а в колонке VSZ — виртуальный размер в Кбайт. В колонке RSS указывается размер резидентной части в Кбайт, а в колонке %MEM — отношение размера резидентной части к общему объему оперативной памяти, в процентах. По команде ps –l из указанных колонок выводится только колонка SZ.

Символ W в колонке STAT обозначает, что у процесса нет резидентных страниц в памяти. После того, как пользователь приступил к вводу команды, операционная система вынуждена подгрузить часть машинного кода и данных интерпретатора с диска в память, выгрузив предварительно на диск части других процессов. Из приведенного ниже листинга видно, что в данном случае для ввода команды в командной строке используются только 480 Кбайт кода и данных, а не 1,3 Мбайт, как это было до выгрузки интерпретатора на диск:

```
# ps -o pid,sz,vsz,rss,pmem,stat,comm -p 334
  PID    SZ   VSZ  RSS %MEM STAT COMMAND
  334   519  2076  480  0.7 S    bash
```

Лимитирующие параметры. Операционная система может накладывать ограничения на работу программ. К таким ограничениям могут относиться, в частности, максимальное процессорное время, максимальный размер создаваемого файла, максимальное количество открытых файлов и некоторые другие. Список лимитирующих параметров и их текущие значения можно узнать с помощью

команды и ulimit –a (bash). Если пользователь установил ограничение, отменить его он уже не сможет. Рядовой пользователь имеет право только уменьшать значения ограничивающих параметров. В начале сессии, как оболочка, так и вызванные из нее программы обычно не имеет ограничений на используемые ресурсы.

Состояние процесса

Процесс находится в одном из перечисленных ниже состояний и может быть переведен из одного состояния в другое операционной системой или командами пользователя. Рассмотрим характеристики каждого из состояний и обозначения, которые используют программы ps и top в колонке S или STAT для этих состояний:

19. **Работоспособный (runnable) процесс**. Если процесс в текущий момент выполняет какие-либо действия или стоит в очереди на получение кванта времени на центральном процессоре, он называется *работоспособным* и обозначается символом R.

20. **Ожидающий (спящий, sleeping) процесс**. Это состояние обозначается символом S и возникает после того, как процесс инициирует системную операцию, окончания которой он должен дождаться. К таким операциям относятся ввод/вывод, истечение заданного интервала времени, завершение дочернего процесса и т. д. Операции вывода на медленные носители вроде гибких магнитных дисков и магнитных лент будут приводить к ожиданию программой завершения длительных операций. Системные процессы-демоны большую часть времени проводят в состоянии ожидания и не потребляют процессорное время.

21. **Остановленный (stopped) процесс**. Процесс можно остановить в любое время и после останова продолжить его выполнение. Остановленный процесс не потребляет основные ресурсы компьютера — процессор и оперативную память. Пользователь может остановить процесс, например, чтобы дать возможность другому процессу использовать больший объем оперативной памяти или быстрее завершиться. Операционная система останавливает фоновые процессы в случае, когда они пытаются ввести данные с терминала. Это состояние обозначается символом T.

22. **Завершившийся (зомби, «zombie») процесс**. После завершения процесса информация о нем должна быть удалена операционной системой из таблицы процессов. Если родительский процесс выполняется параллельно с дочерним, не ожидая его завершения, ОС вынуждена хранить в таблице процессов запись о завершившемся процессе, хотя он реально уже не существует. Завершившийся процесс обозначается символом Z.

Кроме перечисленных разные системы могут различать и некоторые другие состояния процессов.

Оперативный и фоновый режимы исполнения процессов

В оперативном режиме оболочка после запуска дочернего процесса переходит в режим ожидания его завершения, а не в режим ввода очередной команды пользователя. При этом дочерние процессы могут вводить данные с терминала.
Если после команды поставить символ &, она будет выполняться в фоновом режиме. После запуска фонового процесса интерпретатор сразу переходит к исполнению

следующей команды или выводит на экран приглашение командной строки, не ожидая завершения фонового дочернего процесса.
Пример выполнения в оперативном и фоновом режимах программы a.out, которая выводит на экран приветствие «Hello, world!»:

% **a.out**

Hello, world!

% **a.out &**

[1] 16380

% *Hello, world!*

[1] Done a.out

Первая команда производит запуск программы в оперативном режиме. Оболочка ожидает завершения оперативной программы и только после этого выводит приглашение командной строки. Вторая команда — запуск программы в фоновом режиме. После запуска программы оболочка выводит на терминал в квадратных скобках *номер фоновой задачи*, а также PID процесса. Оболочки устанавливают собственную нумерацию фоновых задач для удобства управления ими с помощью встроенных команд jobs, fg, bg и kill. Когда фоновая программа завершается, оболочка может оповестить об этом пользователя специальным сообщением, состоящим из трех полей:

1.номер фоновой задачи (в квадратных скобках);

2.причина завершения (например, «Done» — выполнена);

3.командная строки (команда и ее аргументы).

Пример:

ls

a.out* s.c s.o

[1]+ Done a.out

Для того чтобы стандартный вывод и сообщения об ошибках фоновых процессов не попадали на экран терминала, рекомендуется направлять эти потоки в файлы с помощью конструкций >, >>:

ls –al >>out 2>&1 &

Ввод данных с терминала может осуществлять только оперативный процесс. Если фоновая задача пытается ввести данные с терминала, она останавливается и пользователь получает извещение.
Остановленную фоновую задачу можно перевести в оперативный режим командой fg. Эта команда воспринимает в качестве аргумента *номер задачи* (а не PID), который предваряется символом процента:

fg %1

Фоновых задач у одной оболочки может быть много.
Если ввести команду fg без аргументов, в оперативном режиме будет выполняться задача, отмеченная символом + в списке задач, выводимом командой jobs. Если остановленная задача всего одна, именно она и будет переведена в оперативный режим. Приведем пример выполнения команды jobs:

jobs

[1]+ Stopped a.out

[2]- Running ./mdx &

Оперативный процесс можно остановить и перевести в фоновый режим. Для этого надо на клавиатуре управляющего терминала нажать комбинацию клавиш Ctrl+Z. После остановки оперативной задачи оболочка присваивает ей номер и выводит сообщение об остановке:

./mdx

Ctrl+Z

[1]+ Stopped ./mdx

Остановленный процесс можно продолжить либо в оперативном режиме с помощью команды fg, либо в фоновом режиме с помощью команды bg. Команда bg имеет такие же аргументы, как и команда fg:

jobs

[1]+ Stopped ./mdx

bg %1

[1]+ ./mdx &

jobs

[1]+ Running ./mdx &

При выполнении команды bg оболочка выводит номер задачи, командную строку и символ &, показывающий, что команда исполняется в фоновом режиме.

Фоновые процессы, запущенные во время сессии, могут продолжать работу после ее окончания. Пользователь после запуска фонового процесса может ввести команду logout, а работоспособный фоновый процесс продолжит выполнение. У этого процесса не будет управляющего терминала и все файлы, связанные с данным управляющим терминалом, потеряют свой источник данных (клавиатуру) или их получателя (экран). Любая попытка ввода или вывода в эти файлы является недопустимой операцией и приведет к аварийному завершению процесса. К таким процессам нельзя применять команды fg, bg и jobs, а управлять ими можно только с помощью сигналов, которые будут описаны позже.

Для предупреждения аварийного завершения фонового процесса при выводе данных после окончания сессии следует перенаправить стандартные файлы вывода и ошибок в файл одним из двух способов:

1.средствами интерпретатора >, >>, >& и т. д.;

2.командой nohup.

Команда nohup направляет стандартные файлы вывода и ошибок в файл nohup.out, который расположен в текущем каталоге. Если файл уже существует, весь вывод будет добавлен в конец этого файла:

nohup *программа* [аргументы] [<*файл*] &

Имя программы является первым аргументом команды nohup, а аргументы собственно программы указываются после ее имени. Если программа будет вводить

данные из стандартного файла ввода, пользователь должен заранее создать файл с данными на диске и заменить ввод с терминала вводом из этого файла, пользуясь средствами оболочки. Команда nohup не запускает задачу в фоновом режиме. Для этого в командной строке следует использовать символ &.

Сигналы и управление процессами

В ОС UNIX управление процессами осуществляется с помощью *сигналов*. Большинство сигналов имеют стандартизованные номера и мнемонические обозначения. Сигналы могут отличаться номерами, названиями и назначением на ЭВМ с различной архитектурой.
Пользователь может послать любой сигнал с помощью команды kill. Реакцией операционной системы на сигнал может быть:

1. завершение процесса (стандартная реакция на подавляющее большинство сигналов);
2. завершение процесса и запись в файл core образа памяти с программой в целях отладки;
3. останов процесса;
4. продолжение остановленного процесса;
5. игнорировать сигнал и продолжить выполнение процесса.

Рассмотрим наиболее важные сигналы. С полным списком сигналов с помощью команды kill –l.

Сигналы управления процессом. Сигналы из этой группы предназначены для экстренного завершения или временной приостановки процесса. Некоторые сигналы посылаются ОС процессу, когда пользователь нажимает определенные комбинации клавиш на клавиатуре. Ряд сигналов могут быть посланы только из процесса (пользователем или системным администратором) для экстренного завершения программы.
Следующие сигналы экстренно завершают программу:

- SIGINT (2) — прервать программу. Посылается при нажатии клавиш Ctrl+C;
- SIGTERM (15) — завершить процесс. Используется для предупреждения программы о скором ее завершении. Большинство процессов перехватывают этот сигнал для того, чтобы выполнить перед завершением подготовительные действия (закрыть файлы, удалить временные файлы, послать сообщения процессам-партнерам и т. д.). Этот сигнал посылает команда kill по умолчанию, если ей не были указаны номер или имя сигнала;
- SIGKILL (9) — безусловное завершение процесса. Самый «сильный» из всех сигналов завершения. Его нельзя игнорировать или перехватить, операционная система завершает процесс без предварительного уведомления.

Стандартная реакция процессов (завершение) на все сигналы, кроме SIGKILL, может быть изменена.
Следующие сигналы служат для остановки и продолжения процессов:

- SIGSTOP — остановить процесс. Посылается при нажатии клавиш Ctrl+Z;

- SIGCONT — продолжить остановленный процесс. Посылается остановленной группе процессов при вводе команд fg и bg или программным путем.

Познакомимся теперь с пользовательским интерфейсом отправки сигналов процессам. Основным средством отправки сигналов является команда kill:
kill -l
kill [-номер] PID...
kill [-название] PID...
kill [-номер] %*задача*...
kill [-название] %*задача*...
Первый вариант предназначен для вывода списка номеров и названий сигналов, которые определены в данной версии ОС. Остальные четыре варианта команды посылают сигнал другому процессу. Если номер или название сигнала не указаны, посылается сигнал SIGTERM.
Оперативный процесс можно завершить нажатием клавиш Ctrl+C. Если программа игнорирует сигналы, посылаемые при нажатии этих клавиш, можно попробовать остановить оперативный процесс нажатием клавиш Ctrl+Z, а затем завершить его сигналом SIGKILL.

Приоритет процесса

Команда nice может понижать приоритет, как исполняемой команды, так и самой оболочки:
nice [–число] [*команда* [аргументы...]]
Команда renice позволяет уменьшить приоритет существующих процессов. Ее формат:
renice число [PID...] [-u UID] [-g GID]
В отличие от команды nice здесь указывается новое значение приоритета, а не его изменение. Обычный пользователь может только понизить приоритет, указав положительное число, большее значения текущего приоритета процесса

Команды ps и top

Для получения информации о процессах используются в основном две команды: ps и top. Команда ps обсуждалась ранее. Команда top сочетает в себе средства наблюдения над процессами и управления ими. Она выводит на экран и периодически обновляет список наиболее активных процессов. По умолчанию активность определяется значением текущей загрузки процессора В простейшем варианте программа вызывается без аргументов. Выход из нее производится нажатием клавиши Q.
Экран top делится на три части:

1. в верхней части выводится обобщенная информация о процессах и работе системы;
2. средняя часть используется для диалога с пользователем;
3. нижняя часть экрана используется для вывода отсортированного по определенному критерию списка процессов.

В нижней части экрана выводится таблица, напоминающая по форме вывод утилиты ps. Строки, отражающие состояние процессов, сортируются по определенному

критерию. По умолчанию сортировка происходит по текущему значению загрузки процессора.

Программа top позволяет управлять процессами в интерактивном режиме. Для этого служат команды, связанные с определенными клавишами. После нажатия клавиши никаких других клавиш нажимать не надо. После ввода команды, требующей аргументов, top выводит в строке диалога приглашение для ввода аргумента. Текст в этой строке можно редактировать обычным способом, а завершить ввод необходимо клавишей Enter.

Названия и значение колонок, выводимых разными версиями программы top:

- PID — идентификатор процесса;
- USER — регистрационное имя пользователя-владельца процесса;
- UID — реальный идентификатор пользователя;
- PRI и NI — приоритет и относительный приоритет процесса;
- SIZE — полный объем памяти, занимаемой процессом;
- RES RSS — размер резидентной части процесса;
- SHARE — размер используемых процессом разделяемых библиотек;
- STAT — состояние процесса;
- LIB — объем библиотечных подпрограмм;
- %CPU — текущее значение загрузки процессора;
- %MEM — часть оперативной памяти, занятая резидентной частью процесса;
- TIME — процессорное время, затраченное на выполнение процесса;
- COMMAND — команда запуска процесса;
- PPID — идентификатор родительского процесса;
- TSIZE — объем машинного кода программы;
- DSIZE — объем данных (области данных и стека) программы;
- SWAP — объем части процесса, выгруженной в область свопинга.

Задания для практической работы

1. Определите, сколько процессов в данный момент запущено на ЭВМ. Сколько из них работоспособны? Сколько процессов принадлежит вам?
2. Посмотрите иерархию ваших процессов. Сколько ваших оболочек являются оболочками сессии? Сколько сессий открыто всеми пользователями ЭВМ в данный момент?
3. Запустите какую-либо программу в фоновом режиме (лучше всего для этого подходят графические программы, работающие с системой X Window). Попробуйте «убить» ее разными сигналами. Какие из сигналов приводят к завершению данной задачи?
4. Проследите за работой наиболее активных, то есть потребляющих наибольшее процессорное время, процессов на данной ЭВМ. Есть ли среди них процессы с пониженным приоритетом?
5. Напишите на языке Фортран или С программу с бесконечным циклом. Запустите ее с помощью nohup. Измените стартовый приоритет nice, затем поменяйте его во время выполнения программы. Перевидет процесс в состояние «остановлен», затем

возобновите его выполнение. Исследуйте реакцию процесса на другие сигналы (по указанию преподавателя). Компилятор Фортрана вызывается командой:

f77 –o имя_исполняемого_файла имя_исходного_файла

Тема 4. X Window

Как устроена система X Window

X Window представляет собой стандарт для графического пользовательского интерфейса, реализованный для всех UNIX-систем, а также для ряда других ОС. Система X Window представляет собой набор программ и библиотек. Она является сетевой. В X Window программа X-сервер выполняется на локальном компьютере, а графические программы (X-клиенты) могут выполняться как на локальном, так и на удаленных компьютерах.

X Window базируется на следующих составных частях:

23. *X-сервер* — программа, написанная с учетом архитектуры того компьютера, на котором работает пользователь и его видеоподсистемы. X-сервер «умеет» отображать графические объекты на мониторе пользователя. При запуске X Window вначале стартует X-сервер, а затем запускаются X-клиенты.;

24. *X-клиент* — прикладная программа, предназначенная для работы в графической среде. Команды рисования графических объектов на экране монитора передаются от X-клиента к X-серверу с помощью специального протокола передачи данных;

25. *X-терминал* — устройство, на котором выполняется X-сервер. В качестве X-терминала может использоваться графическая рабочая станция UNIX, на которой X-сервер выполняется как одна из нескольких прикладных программ. Это может быть и специализированный компьютер, на котором работают только X-сервер и программы, запущенные на выполнение в среде X Window.

X-сервер и X-клиенты обмениваются сообщениями, посылая друг другу и принимая пакеты с данными и графическими командами. Протокол передачи сообщений X Window может использовать сетевые протоколы прикладного уровня, такие как TCP/IP. Благодаря этому X-сервер и X-клиенты могут работать на разных компьютерах, связанных компьютерной сетью. В таком режиме работы изображение формируется программой, выполняющейся на одном компьютере, а отображается на другом. X-сервер может работать и на одном компьютере с X-клиентом.

Диспетчеры окон

При работе в X Window пользователь обычно работает с несколькими окнами прикладных программ. Система должна иметь набор средств, с

помощью которых пользователь может изменять размеры окон, их положение на экране, решать другие задачи по управлению окнами. Это обеспечивает специальный клиент — *диспетчер окон* — программа, которая указывает X-серверу, где должны располагаться окна приложений, управляет перемещением окон по рабочему экрану, изменением их размеров и т. д.
Если запустить X Window без диспетчера окон, система будет работать, возможно, будут открыты окна каких-то программ, но эти окна не будут иметь никакого оформления, управлять ими будет невозможно. В любой момент времени в системе может работать только один диспетчер окон. В UNIX имеется большое число диспетчеров окон (*оконных менеджеров*). Среди них:

- **fvwm** — один из наиболее распространенных диспетчеров окон.
- **fvwm95** — базируется на fvwm2, привнося в него «аромат» Microsoft Windows 95/98.
- **mwm** — входит в состав коммерческого пакета Motif, имеется бесплатная версия.
- **icewm** — разрабатывался с целью увеличить скорость работы и гибкость настройки.
- **twm** — имеет панель инструментов, конфигурируемые окна, использует фокус и т. д.

Имеются также графические рабочие среды, которые включают в свой состав не только диспетчеры окон, но и различные дополнительные средства — наборы элементов для украшения рабочего стола, наборы утилит и программ для работы с текстами и графикой, администрирования, работы в сети и другие. Наиболее распространены следующие:

- **KDE** — одна из наиболее дружественных графических сред, используемая преимущественно в ОС Linux. Имеется панель инструментов. Поддерживается технология drag and drop, имеются ярлыки, есть возможность гибкой настройки рабочей области с помощью программы KDE Control Center. Файлы определенных типов можно ассоциировать с соответствующими приложениями. Менеджер файлов может использоваться в качестве обозревателя для просмотра страниц WWW. Имеется набор приложений и утилит. В среду KDE интегрирован диспетчер окон kwm;

- **GNOME** — среда для ОС Linux, использующая оконный менеджер enlightenment. Имеется набор приложений, некоторые из них совместимы с KDE.
- **CDE** — является коммерческим продуктом, который создавался консорциумом фирм Sun, IBM, Hewlett Packard и Novell. Применяется в ОС Sun Solaris и других операционных системах указанных производителей UNIX-систем.

Запуск и останов X Window

Старт X Window производится командой:
startx
startx — это командный файл, из которого запускается на исполнение бинарный файл xinit, последний запускает необходимый X-сервер.
В случае успешного старта X Window происходит последовательное выполнение действий, необходимых для инициализации графического режима и запуска начального набора X-клиентов. В результате появляется графический экран, вид которого зависит от используемого диспетчера окон, общесистемных и пользовательских настроек.
Описание настроек и начального набора X-клиентов содержится в соответствующих конфигурационных файлах. При запуске система считывает конфигурационные файлы.
Имеется несколько способов завершения сессии работы в X Window. Если на поверхности виртуального рабочего стола находится панель инструментов, следует найти на этой панели кнопку, дающую выход к команде завершения X-сессии. Меню может активизироваться и при нажатии одной из кнопок мыши, например, правой. «Грубый» и быстрый способ выхода из X Window — применение комбинации клавиш Ctrl+Alt+Backspace.
Сессия работы пользователя в системе X Window, таким образом, начинается после запуска X-сервера или ввода пользовательского идентификатора и пароля в специальную форму диспетчера дисплея системы X Window и заканчивается после завершения работы X-сервера. Сеанс работы в ОС UNIX при этом может закончиться (при наличии диспетчера дисплея) или продолжаться.

Основные элементы графического интерфейса X Window

Познакомимся с основными элементами графического интерфейса X Window.
Окно. При работе в X Window каждая запущенная на выполнение программа открывает одно или несколько *окон*. В правой или левой

части окна, а также снизу может находиться *полоса прокрутки*. Назначение полосы прокрутки заключается в том, чтобы дать пользователю возможность просмотра той части текста, которая не умещается в области окна. Перемещая мышью движок на полосе прокрутки, можно просмотреть весь файл целиком. Нижняя полоса прокрутки позволяет просматривать строки, длина которых превышает горизонтальный размер окна. Полоса прокрутки может начинаться и оканчиваться кнопками с изображением стрелок, направленных вверх и вниз (для вертикальной полосы) или влево и вправо (для горизонтальной полосы). Способы прокрутки:

- установить указатель на стрелку, соответствующую требуемому направлению прокрутки и нажать на левую кнопку мыши. Стрелок в полосе прокрутки может и не быть, направление прокрутки (вперед или назад) в этом случае определяется тем, какая кнопка мыши нажата — левая или правая;
- установить указатель над ползунком или под ним – для вертикальной полосы и слева или справа от него для горизонтальной полосы и нажать левую кнопку мыши;
- установить указатель на ползунке, нажать левую кнопку мыши и передвинуть ползунок. Его перемещение будет сопровождаться прокруткой текста.

Для изменения размера окна используются *манипуляторы*, расположенные в каждом из четырех углов рамки окна. Для изменения размера следует установить указатель мыши на манипулятор, при этом вид указателя меняется, затем нажать левую кнопку и, не отпуская ее, растянуть окно.

Меню. *Меню* представляет собой перечень часто используемых команд, которые позволяют пользователю управлять окнами или прикладными программами.

В меню некоторые команды могут отображаться «бледной» окраской. Это означает, что такие команды доступны только при выполнении определенных условий. Так, например, команда «Восстановить» (Restore) может исполняться только, если окно свернуто в пиктограмму. Работа с меню производится следующим образом. Указатель мыши размещается над меню, затем нажимается левая кнопка мыши и, при нажатой кнопке, указатель проводится по пунктам меню. Активизация требуемого пункта меню (запуск соответствующей программы) происходит при освобождении кнопки.

Элементы управления. При работе в графической среде приходится иметь дело с различными элементами управления, в числе которых:

- *кнопки*, позволяющие исполнять команды или задавать те или иные установки (режимы работы программы);

- *текстовые поля*, представляющие собой специальную область для ввода текстовой информации;
- *списки*, представляющие собой перечень вариантов действий, из которого можно выбрать только один;
- *«ползунки»*, специальные элементы интерфейса, позволяющие производить пошаговый выбор значения из некоторого предопределенного диапазона.

Операции по управлению окнами включают в себя такие действия, как выбор активного окна (перенос фокуса), перемещение окна по поверхности виртуального рабочего стола, изменение размера окна и его удаление с поверхности рабочего стола.

Для перемещения окна по поверхности рабочего стола указатель мыши устанавливается на полосу заголовка окна, затем нажимается левая кнопка мыши, которая удерживается нажатой и окно переносится в новое положение. Другой способ заключается в том, что надо одновременно нажать клавишу Alt и левую кнопку мыши, предварительно поместив указатель во внутренней части окна. После этого, удерживая клавишу и кнопку в нажатом состоянии, можно переместить окно в любое положение.

Если открыто несколько окон и они расположены одно поверх другого, перенести наверх одно из окон можно с помощью двойного щелчка мыши по области окна. Двойной щелчок в области заголовка увеличивает или уменьшает размер окна.

Фокус. *Фокус* — это активное окно, в которое направляется ввод с клавиатуры. Фокус определяется положением курсора, а установка фокуса может производиться как в стиле X Window, так и в стиле Microsoft Window. В первом случае для установки фокуса достаточно перевести указатель мыши в область окна приложения и оно станет активным, а во втором случае переключение фокуса осуществляется щелчком мыши. Полоса заголовка окна, в котором установлен фокус, выделяется более насыщенным и ярким цветом.

Виртуальный рабочий стол. Важнейшей концепцией современной графической среды является концепция *виртуального рабочего стола* (virtual desktop). Компьютер может использоваться для решения множества задач одновременно. Окна активных приложений могут быть размещены на поверхности виртуального рабочего стола. Во многих диспетчерах окон имеется несколько экранов — рабочих столов, между которыми легко переключиться в процессе работы. Все экраны могут отображаться на одном и том же мониторе.

На виртуальном рабочем столе может размещаться панель инструментов, содержащая часто используемые команды, команды

запуска различных приложений и элементы управления рабочим столом.

Пользователь в любой момент времени работает лишь с одним–двумя приложениями, а остальные могут в это время находиться или в состоянии выполнения или в ожидании ввода данных. В этом случае имеет смысл не держать окна всех приложений на одном экране, а отделить активные приложения от временно неактивных, что делает работу более удобной. При работе в среде X Window это можно сделать двумя одним из двух способов:

1. перевести ненужные окна в пиктограммы;
2. использовать виртуальный рабочий стол, когда имеется несколько различных экранов, и в каждый из них можно переключиться с помощью соответствующей команды. Диспетчеры окон часто имеют специальную панель — *пейджер*, имеющую вид небольшой диаграммы с условным изображением всех экранов и открытых в каждом из них окон приложений.

Ярлыки. Во многих графических средах, таких как CDE или KDE, на поверхность виртуального рабочего стола могут быть вынесены *ярлыки*, дающие быстрый доступ к часто используемым файлам, каталогам, программам или адресам в Интернете.

Пиктограммы. *Пиктограммы* представляют собой условные графические изображения, заменяющие окна выполняющихся программ. Пример использования пиктограмм — работа с большим числом одновременно запущенных приложений, часть из которых используется редко, но окна, соответствующие этим приложениям, загромождают рабочее пространство экрана. В этом случае часть из них переводится в иконки, которые можно в любой момент развернуть в обычные окна.

X-атрибуты. Вид окна определяется набором параметров, которые называются *X-атрибутами*. Одним из X-атрибутов является *геометрия* окна. Геометрия определяется следующими величинами:

1. горизонтальный размер окна. Обычно указывается в пикселах. Реже размеры окна задаются в символах, эта единица измерения используется для задания геометрии окна редактора emacs и эмулятора X-терминала xterm;
2. вертикальный размер окна;
3. расстояние по горизонтали от левого края экрана до левой границы окна (в пикселах). Отрицательное смещение отсчитывается от правой границы экрана;

4. расстояние по вертикали от верхней границы экрана до верхней границы окна. Отрицательное смещение отсчитывается от нижней границы экрана.

Геометрия окна задается в следующем формате:
HSIZExVSIZE[+ или -]HSHIFT[+ или -]VSHIFT
Здесь HSIZE и VSIZE — горизонтальный и вертикальный размеры окна (между ними записывается символ x), а HSHIFT и VSHIFT — горизонтальное и вертикальное смещения соответственно. В примере:
503x73-78+10
окно имеет протяженность по горизонтали 503 пикселов, по вертикали 73 пиксела и отстоит от нижней границы экрана на 78 пикселов, а от левой границы на 10 пикселов.
Атрибутами окна являются также *основной цвет* (цвет символов вывода текстовой информации) и *цвет фона*. Цвета могут быть заданы с помощью английских наименований, взятых из базы данных цветов X-сервера, например, yellow, blue и т. д.
Графическому приложению сопоставляется *дисплей*, который задается тремя параметрами:

1. *имя компьютера*, на котором выполняется X-сервер. На изолированных компьютерах клиенты и сервер выполняются на одном компьютере, а в сетевой конфигурации могут и на разных. В первом случае имя компьютера не требуется, а во втором указывается сетевое имя (может быть указан IP-адрес);
2. *номер X-сервера*, выполняющегося на данном компьютере. X-серверов может быть несколько, хотя обычно имеется лишь один. Нумерация ведется от 0;
3. *номер графического экрана*. Один X-сервер может одновременно управлять выводом изображений на несколько графических экранов (мониторов). Нумерация экранов ведется от 0.

Дисплей задается текстовой строкой:
имя_компьютера:номер_сервера.номер_экрана
Пример:
bluebird.niicha.vo.ru:0.0
В этом примере bluebird.niicha.vo.ru — сетевое имя компьютера, затем следует разделитель — двоеточие, далее идут номер X-сервера (0), разделитель (точка) и номер графического экрана (0). Установить номер дисплея можно, задав переменную окружения DISPLAY. В bash это делается с помощью команды:
```
# export DISPLAY=bluebird.niicha.vo.ru:0.0
```
Проверить, установлено ли правильное значение переменной DISPLAY можно с помощью команды:
```
# echo $DISPLAY
```

Вид окна каждого приложения может быть настроен пользователем. Для настройки используются ключи, с которыми могут вызываться X-клиенты:

- –geometry — геометрия окна;
- –display — дисплей;
- –fg — основной цвет;
- –bg — цвет фона;
- –font — размер шрифта;
- –fn *шрифт* — шрифт.

Примеры использования:

```
xterm –geometry 80x24+10+100
xterm –display bluebird.niicha.spb.ru:0.0
xterm –fg yellow
xterm –bg blue
xterm –font 9x15
xterm –fn –Adobe-Times-Bold-R-Normal—14-100-100-100-P-76-ISO8859-1
```

Настройка X Window

Настройка включает в себя определение вида окон, приложений, стартующих при запуске X Window, используемые шрифты и т. д. Вид приложения может задаваться с помощью ключей при его запуске, в конфигурационных файлах, в стартовом файле X-сессии. Синтаксис записей, определяющих значения ресурсов следующий:

```
приложение.элемент_графического_интерфейса. … .ресурс: значение
```

Здесь вместо многоточия может быть указана цепочка элементов, связанных отношениями иерархического подчинения. Например, запись:

```
xterm*background: Green
```

устанавливает зеленый цвет фона для окна программы xterm. Если вместо точки в качестве разделителя используется звездочка, она играет роль «метасимвола», обозначая все элементы, использующие данный ресурс (в нашем примере это background). Можно опустить и имя программы-клиента. Строка:

```
*background: white
```

устанавливает белый цвет фона для всех клиентов и элементов, имеющих дело с ресурсом background. Регистр букв в этих записях имеет значение.

При указании цветов следует иметь в виду, что в системе X Window используется RGB-представление, в котором каждому цвету сопоставляются 3 16-битных значения. Эти значения характеризуют интенсивности красной, зеленой и синей составляющих

соответственно. Некоторым цветам сопоставлены имена (буквенные идентификаторы, например, SeaGrey). Выяснить имена, которые понимает ваш X-сервер, можно с помощью утилиты showrgb. Пример вывода showrgb:

```
255 250 250     snow
248 248 255     ghost white
248 248 255     GhostWhite
245 245 245     white smoke
245 245 245     WhiteSmoke
220 220 220     gainsboro
……
```

Конфигурация X Window описывается в специальных файлах. В системе имеются общесистемные конфигурационные файлы, которые создаются и изменяются суперпользователем. В них описывается конфигурация видеоподсистемы, используемые в X Window видеорежимы и другая важная информация. Имеются локальные, то есть, принадлежащие конкретному пользователю, конфигурационные файлы, в которых описывается внешний вид, стиль графической среды для данного пользователя. Локальный конфигурационный файл имеет больший приоритет, чем системный. При запуске X Window сначала пытается прочесть локальный, пользовательский файл конфигурации и лишь в том случае, когда это сделать не удается, считывает системный. Если отсутствуют оба файла, выбираются настройки по умолчанию. Локальные файлы находятся в домашнем каталоге пользователя. Файл, описывающий программы, стартующие автоматически при запуске X Window, а также некоторые конфигурационные параметры (например, настройки клавиатуры), называется .xinitrc. Последняя команда в .xinitrc должна запускать диспетчер окон.

Пример файла .xinitrc приводится ниже. Это простейший файл, в котором содержится лишь вызов одного из диспетчеров окон. Можно изменить вторую строку этого командного файла, подставив туда вызов любого другого диспетчера окон из числа имеющихся в системе, например, twm, afterstep и т. д.:

```
#!/bin/sh
exec /usr/X11R6/bin/twm
```

В более сложном варианте данного файла содержатся дополнительные строки. Программа xsetroot позволяет настроить вид главного окна в X Window. В приведенном ниже примере задается фон, который имеет

сплошную темно-синюю окраску. Далее содержится вызов X-клиентов, которые запускаются при старте X Window. Это xterm — эмулятор X-терминала и xclock — часы. Все X-клиенты должны запускаться в фоновом режиме (символ & в конце командной строки), в противном случае процесс запуска X Window приостановится, и система будет ожидать завершения выполнения программы-клиента. Только диспетчер окон запускается в интерактивном режиме! Попробуйте поменять используемый диспетчер окон, стартовый набор X-клиентов, их число, расположение на экране и внешний вид. Попробуйте изменить цвет фона, для чего вам придется обратиться к описанию программы xsetroot.

```
#!/bin/sh
# Установка раскладки клавиатуры
xmodmap -e "keycode 22=BackSpace"
# Установка окраски поверхности виртуального
# рабочего стола: сплошная темно-синяя
xsetroot -solid DarkBlue
# X клиенты, запускаемые при старте X Window:
xterm -g 80x20+150+8 &        # эмулятор X терминала
xclock -g +815+0 -digital &   # запуск часов
# Запуск диспетчера окон twm
exec twm
```

Кроме файла .xinitrc в X Window используются и другие конфигурационные файлы, создавать и модифицировать которые может сам пользователь. Так, раскладка клавиатуры может быть описана в отдельном файле .Xmodmap.

В специальных конфигурационных файлах ресурсов хранится информация о тех параметрах, которые используются диспетчером окон для определения вида окон, информация об используемых шрифтах и т. д. Так, например, файлы .twmrc или .fvwmrc считываются соответствующим диспетчером окон и содержат информацию, необходимую для его работы (или system.fvwmrc). Используемое в этих файлах имя ресурса состоит из имени диспетчера окон и собственно имени ресурса (например, twm*background).

В файле .Xclients содержатся перечень и настройки тех X-клиентов, которые запускаются каждый раз в начале пользовательского сеанса работы в системе X Window. Этот файл располагается в домашнем

каталоге пользователя. Синтаксис файла совпадает с синтаксисом файла .xinitrc:

```
xv -rmode -noresetroot -quit /usr/local/backgrounds/slate.xpm &
kfm &
kcontrol -init &
kbgndwm &
krootwm &
kpanel &
kvt &
kwm
```

В данном примере стартовый набор X-клиентов включает графическую программу xv, файловый менеджер для KDE, панель управления и некоторые другие программы для этой графической системы и диспетчер окон kwm.
Клиенты, перечисленные в файле .xinitrc, не запускаются, если в домашнем каталоге пользователя найден файл .Xclients.

X-клиенты

Познакомимся с некоторыми X-клиентами. Эти программы с большой вероятностью имеются в X Window на любом UNIX-компьютере.
Один из способов запуска X-клиентов — из эмулятора X-терминала xterm с помощью командной строки. Лучше всего запуск производить в фоновом режиме, так как при запуске программы в интерактивном режиме она монопольно захватывает xterm и с ним нельзя будет работать. В случае необходимости придется запускать новый терминал, что потребует дополнительных затрат как оперативной памяти, так и процессорного времени. Другой способ — использование меню.
X-клиенты могут работать на том же компьютере, что и пользователь, а могут быть запущены на удаленном компьютере. В последнем случае вначале необходимо войти в сеанс работы с удаленным UNIX-компьютером, а затем, перед запуском клиента задать дисплей, с которым будет работать данная программа, причем обязательно следует указать сетевой адрес компьютера с X-сервером.

Диспетчер файлов

В состав многих графических сред (таких как CDE, KDE и GNOME) входит *диспетчер файлов* (file manager), который позволяет

перемещаться по файловой системе UNIX и выполнять различные операции с файлами и каталогами. Файлы и каталоги в диспетчере файлов отображаются в виде пиктограмм. Манипуляции с ними производятся с помощью мыши. Современные графические среды поддерживают технологию drag and drop. В этом случае для того, чтобы, например, распечатать файл, достаточно перетащить соответствующую ему пиктограмму на пиктограмму принтера, расположенную на виртуальном рабочем столе. Копирование или перенос файлов выполняется аналогичным образом.

Часто диспетчер файлов позволяет ассоциировать (связывать) файлы определенного типа с определенными прикладными программами, так, что щелчок по пиктограмме файла приведет к запуску ассоциированного приложения, загрузке в него данного файла и его обработке. Диспетчеры файлов позволяют выполнять не только простые операции с файлами, но и менять права доступа к ним, а также выполнять другие сложные действия.

При работе с диспетчером файлов важно освоить следующие навыки:

1. *выбор и выделение* одного или нескольких файлов, что позволяет применять одну операцию сразу к нескольким файлам;
2. *перетаскивание объекта* (dragging and dropping) — позволяет с помощью мыши и перетаскивания пиктограмм объектов и размещения их друг на друге выполнять различные действия, не набирая сложные или длинные команды;
3. *использование всплывающих меню*. Каждая пиктограмма в диспетчере файлов имеет свое собственное всплывающее меню, которое дает быстрый доступ к различным операциям с данным объектом.

Для того, чтобы, работая с диспетчером файлов, выделить файл или каталог, достаточно щелкнуть по его пиктограмме, при это название объекта выделяется. Выделить несколько объектов можно, нажав левую кнопку мыши. Затем следует, не отпуская ее, очертить прямоугольную область, включающую в свой состав пиктограммы требуемых объектов. Еще один способ — нажатие клавиши Ctrl при выборе каждого объекта. Отменить выделение можно, щелкнув один раз в пустой области внутри окна диспетчера файлов. Отметив группу файлов можно удалить их сразу, переместить из одного каталога в другой или вынести несколько объектов на поверхность рабочего стола.

Для того, чтобы переименовать файл или каталог, можно выделить объект, набрать новое имя и нажать клавишу Enter. Такой метод переименования действует, например, в диспетчере окон enlightenment. Справочную информацию, относящуюся к определенному объекту, иногда можно получить, отметив этот объект и нажав клавишу F1.

Открыть файл или каталог можно с помощью двойного щелчка по пиктограмме объекта. Другой способ — использование всплывающего меню данного объекта.
Для того, чтобы «перетащить» файл или каталог, необходимо поместить указатель на его пиктограмме, нажать среднюю кнопку мыши и, удерживая ее в нажатом состоянии, перенести объект в нужное место. Для того, чтобы отменить перетаскивание, следует нажать клавишу Esc перед освобождением кнопки мыши. Если выделена группа пиктограмм, перетащить ее можно, перетаскивая одну пиктограмму из группы.
Если пиктограмма файла переносится на пиктограмму каталога или во второе открытое окно диспетчера файлов, это означает перенос файла в соответствующий каталог.
Перенос файла на поверхность виртуального рабочего стола может приводить к созданию ярлыка для данного файла. Подобно рабочему столу Microsoft Windows, на рабочем столе многих графических систем UNIX имеется «мусорная корзина» (Trash). Перенос файла на изображение мусорной корзины приведет к удалению файла (но это удаление еще не окончательное, окончательно файл будет удален при очистке корзины). При переносе файла на пиктограмму программы-приложения эта программа будет запущена, при этом файл будет использоваться в качестве аргумента. В том случае, когда программа не может открыть переносимый на ее пиктограмму объект, пиктограмма объекта вернется на свое исходное положение в диспетчере файлов или на виртуальном рабочем столе.
Если при переносе пиктограммы объекта будет нажата клавиша Ctrl, файл будет не переноситься, а копироваться. Нажатая при переносе иконки клавиша Shift приводит к созданию символической ссылки на файл из того каталога, в который производится перенос.
Всплывающее меню открывается при нажатии правой кнопки мыши на пиктограмме объекта. Выбрать из появившегося меню необходимую команду можно, щелкнув по ней мышкой.

Другие клиенты

xmodmap представляет собой утилиту, позволяющую изменить назначение клавиш клавиатуры и кнопок мыши при работе в системе X Window:

xmodmap [ключи] [*имя_файла*]

Файл *имя_файла* содержит выражения xmodmap и находится обычно в домашнем каталоге пользователя. Его имя .Xmodmap или .xmodmaprc.

X-клиенты используют специальные таблицы для преобразования кодов, вырабатываемых при нажатии клавиш на клавиатуре, в символы. Утилита xmodmap используется для редактирования и отображения этих таблиц. Каждой клавише клавиатуры сопоставляется уникальный код, который вырабатывается при её нажатии. X-сервер работает со специальной внутренней таблицей, которая сопоставляет коды клавиш символам, имеющимся на клавишах (*раскладка клавиатуры*). С помощью этой таблицы X-клиенты интерпретируют принимаемые от клавиатуры коды как соответствующие символы. В качестве примера можно поменять местами клавиши Backspace и Delete:

```
# xmodmap –e “keysym Delete = Backspace”
```

Некоторые пользователи предпочитают использовать клавишу Backspace в качестве клавиши Delete. Это можно сделать с помощью команд:

```
# xmodmap –e "keysym BackSpace = Delete"
# echo "XTerm*ttyModes: erase ^?" | xrdb -merge
```

Вторая из этих команд позволяет использовать данную установку во всех окнах эмуляторов X-терминала.

xsetroot — утилита, которая позволяет задать параметры главного окна:

```
# xsetroot [ключи]
```

Вызов без ключей или с ключом –def восстанавливает значения параметров фона, принятые по умолчанию. Некоторые ключи xsetroot:

- –cursor *cursorfile maskfile* — задать форму курсора, когда он находится вне окна приложения. Изображение курсора содержится в файле *cursorfile*, который представляет собой битовую карту. Файл *maskfile* является маской и также представляет собой битовую карту;
- –cursor_name *имя_курсора* — установить один из стандартных курсоров;
- –bitmap *файл* — определить «обои», которые представляют собой образец в формате битовой карты, покрывающий поверхность рабочего стола. Такой образец можно создать с помощью утилиты bitmap;
- –fg *цвет* — установить указанный цвет в качестве основного. Данный ключ действует только в сочетании с ключами –cursor, –bitmap и некоторыми другими;
- –bg *цвет* — установить указанный цвет в качестве цвета главного окна;
- –solid *цвет* — установить сплошную окраску главного окна заданным цветом.

xterm — эмулятор X-терминала. Эмулятор X-терминала позволяет при работе в среде X Window использовать команды текстового режима. Запустить терминал можно с помощью панели инструментов, если таковая имеется и если на ней находится пиктограмма с изображением терминала, а можно и из командной строки, например, из другого X-

терминала. В последнем случае запускать X-терминал следует в фоновом режиме для того, чтобы в дальнейшем можно было работать и с тем X-терминалом, из которого был запущен второй экземпляр. Завершается его работа или в результате закрытия окна, или с помощью команды exit. Команда запуска X-терминала может находиться в меню диспетчера файлов. Из X-терминала можно запускать на выполнение как программы с традиционным текстовым интерфейсом, так и графические программы X-клиенты.
Ниже приведены некоторые ключи программы xterm:

- –bd *цвет* — установить цвет рамки окна;
- –bg *цвет* — установить цвет фона (по умолчанию белый);
- –bw *число* — задать ширину рамки окна в пикселах;
- –display *дисплей* — задать дисплей;
- –fg *цвет* — задать основной цвет;
- –fn *шрифт* — задать шрифт (по умолчанию fixed);
- –geometry *геометрия* — задать геометрию окна;
- –title *строка* — вывести указанную строку в качестве заголовка окна.

При работе с X-терминалом можно использовать четыре меню — main, vt, font и tek. Каждое из них вызывается с помощью определенной комбинации клавиш.
Меню xterm появляется при одновременном нажатии клавиши Ctrl и левой кнопки мыши, когда указатель мыши расположен в окне X-терминала. Это меню содержит пункты, одинаковые как в режиме эмуляции терминала VT102, так и в режиме эмуляции терминала Tektronix:

- Continue — продолжение выполнения процесса, приостановленного командной последовательностью Ctrl+Z;
- Kill — посылает процессу, запущенному из X-терминала, сигналы SIGCONT, SIGTSTP, SIGINT, SIGHUP, SIGTERM и SIGKILL.

и некоторые другие. Имеется пункт SecureKeyboard, использование которого позволяет включить режим, в котором весь ввод с клавиатуры при работе с X-терминалом будет направляться только в xterm. Имеется команда распечатки содержимого окна, а с помощью пункта Quit можно завершить работу с X-терминалом.
Меню vt позволяет установить режимы работы эмулятора VT102 и вызывается одновременным нажатием средней кнопки мыши (или правой и левой одновременно) и клавиши Ctrl. Указатель мыши при этом должен быть расположен в окне X-терминала.
Меню font позволяет установить шрифт, который используется для отображения текста в окне X-терминала. Оно вызывается одновременным нажатием правой кнопки мыши и клавиши Ctrl.

Запуск удаленных X-клиентов, когда X-сервер работает на одном компьютере, а приложение на другом, соединенном с первым компьютерной сетью, значительно ослабляет безопасность системы, потому что X-протокол не защищен от перехвата сообщений. Усилить защиту можно, ограничив круг удаленных компьютеров, которым разрешено использовать ваш X-сервер:

```
# xhost [[+–] имя]
```

В качестве аргумента указывается сетевое имя компьютера или регистрационное имя конкретного пользователя, знак «плюс» означает разрешение, а «минус» — отказ в использовании X-сервера. Задание имени компьютера разрешает использование данного X-сервера всем пользователям указанного компьютера.

xman является графическим интерфейсом для справочной системы man.

xclock — часы.

xfontsel позволяет справиться с проблемой выбора шрифтов. В X Window для обозначения различных шрифтов используются длинные названия. Эта система называется *логическим описанием шрифтов X Window* (X Logical Font Description). Утилита xfontsel позволяет определить название шрифта по образцу текста, в котором используется данный шрифт.

Имя шрифта в XLFD состоит из символов:

- – — разделитель полей;
- * и ? — метасимвол.

а также полей, содержащих целочисленные значения или буквенно-цифровые идентификаторы. Пробел является значимым символом.

Пример имени шрифта:

```
-misc-fixed-medium-r-normal—20-200-75-75-c-100-iso8859-15-*5
```

Назначение полей имени в порядке их следования (слева направо):

1. FOUNDRY — имя организации, которая поставляет файлы данного шрифта или последней модифицировала этот шрифт;
2. FAMILY_NAME — гарнитура, то есть название семейства шрифтов, объединенных общностью рисунка и которые являются вариациями одного базового типографского стиля Примеры названий гарнитур: Helvetica, Times Roman;
3. WEIGHT_NAME — насыщенность шрифта;
4. SLANT — начертание символов: Roman — прямой (R), Italic — курсивный (I), Oblique — наклонный прямой (O), Reverse Italic — обратный курсив (RI), Reverse Oblique — обратный наклонный (RO);
5. SETWIDTH_NAME — пропорции в начертании букв: Normal — нормальное начертание, Condensed — сжатое начертание, Narrow — узкое начертание, Double Wide — широкое начертание.

6. ADD_STYLE_NAME — дополнительная информация об используемом шрифте;
7. PIXEL_SIZE — размер графического изображения символа в пикселах;
8. POINT_SIZE — кегль (размер шрифта);
9. RESOLUTION_X и RESOLUTION_Y — горизонтальное и вертикальное разрешения. Разрешение измеряется в точках на дюйм (dpi — dots per inch);
10. SPACING — строка, которая определяет шпацию для образования пробелов между словами в строке. Кодируется следующим образом: P — пропорциональный шрифт, M — равномерный шрифт, C — клеточный шрифт;
11. AVERAGE_WIDTH — арифметическое среднее ширин всех символов данного шрифта;
12. CHARSET_REGISTRY — зарегистрированное X-консорциумом наименование кодировки;
13. CHARSET_ENCODING — наименование кодировки. Зарегистрированное X-консорциумом наименование кодировки может отличаться от общепринятого. Так например, кодировка ISO Latin-1 зарегистрирована под названием ISO8859-1.

Примеры наименований шрифтов в системе XLFD:

-Adobe-Courier-Medium-R-Normal--10-100-75-75-M-60-ISO8859-1

-Adobe-Times-Bold-R-Normal--14-100-100-100-P-76-ISO8859-1

Самостоятельно определите характеристики каждого из этих шрифтов.

Каждому из полей в логическом описании шрифта отвечает своя кнопка, щелчок по которой приводит к появлению меню с перечнем вариантов для данного поля. Звездочка в том или ином поле играет ту же роль, что и соответствующий метасимвол в именах файлов.
Если щелкнуть мышью по кнопке select, наименование выбранного шрифта в дальнейшем можно будет использовать в других программах для модификации соответствующих настроек.
Пример задания шрифта в конфигурационном файле:
Xterm*Font: -adobe-helvetica-bold-o-normal-8-8-75-75-p-50-hp-roman8
xpaint представляет собой программу для подготовки графических файлов. Поддерживает одновременную работу с несколькими графическими файлами и различными графическими форматами.
xv представляет собой приложение для просмотра графических файлов с широкими возможностями для их редактирования. Поддерживается работа с графическими форматами GIF, JPEG, TIFF, PBM, PGM, PPM, X11 bitmap, Utah Raster Toolkit RLE, PDS/VICAR, Sun Rasterfile, BMP, PCX, IRIS RGB, XPM, Targa, XWD, PostScript и PM.

ghostview — программа просмотра файлов в формате PostScript. Эта программа имеет удобный интерфейс и разнообразные возможности для настройки изображения, параметров страницы и т. д.
xdvi — просмотр файлов в формате .DVI. Такие файлы создаются издательской системой TeX.
xfd — выводит все символы шрифта, имя которого задается в качестве аргумента, например:

```
# xfd –fn -Adobe-Courier-Bold-R-Normal--10-100-75-75-M-60-ISO8859-1
```

xload — выводит график средней загрузки системы.

xlock — вывод заставки, для завершения работы которой придется ввести пароль. Таким образом можно защитить свой сеанс работы в X Window во время вынужденного отсутствия на рабочем месте. После ключа –mode указывается имя заставки (ant, bat, blank, blot, bouboule, bounce, braid, bug, clock, demon, eyes и другие).

xcalc — программа-калькулятор, которая позволяет проводить простые вычисления.
xedit представляет собой простой текстовый редактор для X Window.

Задания для практической работы

1. Запустите на выполнение программу xterm или любой другой X-клиент, так, чтобы его окно имело заданную форму и положение на экране.
2. Запустите на выполнение программу xterm или любой другой X-клиент, так, чтобы фон его окна имел заданную окраску.
3. Сконфигурируйте свой стартовый набор X-клиентов таким образом, чтобы в начале X-сессии запускалось 4 экземпляра часов в аналоговом или цифровом режиме. Размещаться они должны в углах экрана.
4. При работе в сети запустите любой X-клиент на удаленном компьютере так, чтобы его окно было выведено на ваш компьютер.
5. Определите доступный в вашей конфигурации системы X Window набор шрифтов и попробуйте «на лету» изменить шрифты, используемые программой emacs.
6. Измените свои конфигурационные файлы таким образом, чтобы в начале X-сессии запускался указанный диспетчер окон.
7. Измените свои конфигурационные файлы таким образом, чтобы указатель мыши на поверхности главного окна приобретал указанную форму из стандартного набора.

Литература

С.Немнюгин, *М.Чаунин*, *А.Комолкин* Эффективная работа: UNIX. "Питер", Санкт-Петербург, 2001 г., 688 с.