2. ФАЙЛОВАЯ СИСТЕМА ОС GNU/LINUX. РАБОТА СФАЙЛАМИ И КАТАЛОГАМИ

2.1 Файловая система EXT2

Файловая система – это метод организации файлов на устройствах хранения данных. В мире UNIX существует несколько видов различных файловых систем со своей структурой. Наиболее известны традиционная файловая система S5 (для System V) и файловая система UFS, разработанная университетом Беркли для своих операционных систем BSD. Многие файловые системы других UNIX-подобных операционных систем используют модифицированные варианты UFS. В частности, для систем на основе ядра Linux была разработана файловая система EXT2 (Second Extended File System), во многом наследующая основные черты UFS. В настоящее время применяется файловая система EXT3, представляющая собой обычную систему EXT2, дополненную функцией журналирования.

Файловая система EXT2 состоит из последовательности секций (т. н. *групп цилиндров*), каждая из которых содержит не только общую информацию, но и информацию о размещении файлов в этой группе.

Файловая система обычно размещается на дисках или других устройствах внешней памяти, имеющих блочную структуру. Выделение места под данные на диске производится порциями, обычно большими, чем один физический блок (сектор). В UNIX такая порция также называется *блоком*. Далее, если не будет оговорено специально, термин *блок* будет использоваться во втором значении. Размер логического блока определяется при форматировании и может быть 1, 2, 4, 8 или 16 физических блоков (секторов).

Кроме блоков, содержащих файлы и каталоги, в файловой системе организовано ещё несколько служебных областей. *Суперблок* – это наиболее важная часть файловой системы, т. к. содержит сведения, необходимые для работы с файловой системой в целом (число блоков, число i-узлов в системе и в каждой группе цилиндров и др.).

Основной структурой в файловых системах для UNIX является *i-узел* (*i-node*). Каждому файлу соответствует один и только один i-узел. В i-узле содержится информация, описывающая файл: размер файла, время создания и модификации, режимы доступа и права доступа к файлу, информация о местоположении блоков, составляющих файл, и пр. I-узлы содержатся в таблицах фиксированного размера, поэтому число i-узлов (а значит, и файлов) ограничено. Нумерация i-узлов (как и блоков) единая во всём разделе. То есть если в каждой группе по 1976 i-узлов, то i-узлы с 1 по 1976 находятся в группе № 0, i-узлы с 1977 по 3952 – в группе № 1 и т. д.

I-узел определяет блоки, в которых содержится файл. При этом номера первых 12 блоков содержатся непосредственно в i-узле, а остальные блоки адресуются косвенно через специальные таблицы. Каждая таблица занимает ровно один блок. Например, в случае блоков размером 1 Кбайт таблица косвенно адресуемых блоков (ТКАБ) 1-го уровня содержит номера следующих 256 блоков файла. Если файл размером больше 268 блоков, то заполняется таблица косвенно адресуемых блоков 2-го уровня. Она хранит номера блоков, в которых содержатся таблицы 1-го уровня косвенности, адресуя 256×256 блоков. Если же размер файла превышает 65804 блоков, то задействуется таблица косвенно адресуемых блоков 3-го уровня. Если бы для размещения файла требовалось целиком заполнить эту таблицу, то файл занимал бы на диске более 16 Гбайт. В традиционной версии EXT2 под размер файла в i-узле отводится 32 бита (макс. размер файла – 4 Гбайт). Существуют 64-битные версии этой файловой системы.

Другая важная часть i-узла – права доступа к файлу. I-узел содержит *идентификатор пользователя-владельца файла* (UID) и *идентификатор группы владельца файла* (GID). Процесс может получить доступ к файлу только в том случае, если хранящееся в слове атрибутов значение позволяет сделать это. Кроме того, в слове атрибутов хранятся некоторые параметры управления запуском исполняемого файла, а также тип i-узла.

I-узел не содержит имени файла. Имя файла задаётся элементом каталога. Элемент каталога в UNIX называется *связью*, он содержит («связывает») только имя файла и его номер i-узла. Первые два элемента каталога имеют специальное назначение: элемент “.” связан с собственным i-узлом каталога, а элемент “..” связан с i-узлом родительского каталога. С каждым несвободным i-узлом должен быть связан, по крайней мере, один элемент какого-либо каталога. Такой элемент называют ещё *«жёсткой» связью*. Иначе говоря, у каждого файла может быть одно и более имён (жёстких связей). В одном из полей i-узла файла запоминается количество жёстких связей. Как только это значение становится равным нулю, i-узел и все соответствующие ему блоки данных помечаются как свободные (файл «удаляется»).

Кроме имён, которые задаются жёсткими связями, пользователь для доступа к файлу может использовать ещё так называемые *символические связи*. *Символическая связь* – специальный файл, содержащий одно из имён файла, на который эта связь ссылается. Для символической связи выделяется отдельный i-узел, поэтому символическая связь может ссылаться на файл, который не существует или расположен на другом носителе.

2.2 Логическая организация файловой системы

Один из самых простых и наиболее элегантных аспектов конструкции системы UNIX в целом – это представление всего в виде файла. Даже устройства, на которых хранятся файлы, представляются как файлы. Таким образом, файловая система UNIX организована в виде древовидной структуры каталогов, внутри которых находятся файлы и подкаталоги. Физические или логические устройства (диски, ленты) в файловой системе UNIX не указываются. При загрузке система определяет, какое запоминающее устройство будет предоставлять корневой каталог. Остальные устройства внешней памяти монтируются (подключаются) как каталоги внутри корневой файловой системы.

В UNIX можно выделить несколько классов файлов:

- *Обычный дисковый файл* – к этому классу относят обычные файлы любого формата с пользовательской информацией, в том числе исполняемые файлы и объектные модули. Структурой этих файлов управляет пользователь.
- *Каталог* – класс файлов, представляющий собой список жёстких связей, сопоставляющих имя тому или иному i-узлу.
- Кроме того, существует несколько классов *специальных файлов* – это файлы символических связей, файлы отображения внешних устройств и др. Обращение к файлу символической связи система переадресует файлу, на который эта связь ссылается. При обращении к файлу устройства система вызывает ту или иную функцию драйвера соответствующего устройства (выполняется ввод-вывод).

Корневой каталог обозначается, в отличие от DOS и Windows, косой чертой (/). Обычно корневой каталог системы GNU/Linux содержит следующие подкаталоги:

- bin содержит основные программы общего назначения;
- boot содержит файлы, использующиеся при загрузке системы;
- dev содержит файлы отображения внешних устройств;
- etc содержит конфигурационные файлы системы;
- home содержит домашние каталоги пользователей;
- lib содержит служебные файлы (системные библиотеки);
- mnt или media содержит каталоги для подключения других файловых систем;
- sbin содержит программы специального назначения;
- tmp – каталог для временных файлов;
- usr содержит прикладные программы (/usr/bin) и их данные;
- var используется системой в процессе работы для различных целей (ведение журналов, временных баз данных, кэширование данных программ и т. д.)

Максимальный размер имени файла в EXT2 – 65535 символов. Строчные и прописные буквы различаются (my.txt и my.TXT – разные имена). Имя файла может содержать любые символы, кроме слэша[2]. Файл, имя которого начинается с точки (.), считается скрытым и не отображается обычными командами просмотра каталогов.

Для пользователей DOS и Windows привычным является выделение в имени файла «расширения». В UNIX имя файла монолитно. Вместо термина «расширение» используется термин «суффикс». Использование или неиспользование суффиксов остаётся на усмотрение пользователя. При необходимости определить тип документа и связанное с ним приложение оболочка и прикладные программы зачастую делают это не по суффиксу, а по содержимому файла (по нескольким первым байтам, если это возможно).

Когда какой-либо процесс пытается получить доступ к файлу, UID и GID процесса сравниваются с UID и GID файла. В зависимости от результатов такого сравнения выбираются ограничения режима доступа. При доступе к файлу через символическую связь права самой связи игнорируются, эффективны лишь права файла, на который она ссылается. UNIX различает три режима доступа к файлу:

Таблица 2.1. Режимы доступа к файлам

Режим	Для обычных файлов	Для каталогов
r	Чтение	Получение списка файлов
w	Запись	Удаление, создание, переименование файлов в каталоге
x	Исполнение	Переход в каталог, получение атрибутов файлов

Процессам, запущенным от имени администратора системы (пользователя root), все файлы доступны для любого типа доступа вне зависимости от атрибутов файла. Администратор может также поменять для любого файла UID владельца, GID владельца и атрибуты доступа. Различаются три класса прав доступа к файлу. Для каждого класса доступа в атрибутах файла хранится свой набор разрешённых режимов доступа.

[2] Символы \ | $! & % * ( ) { } ~ ` ' " ; < > ? и пробел при использовании их в командной строке оболочки имеют специальное назначение, в таком случае их надо «экранировать» обратной косой чертой (\) или заключать имя файла в кавычки.

1. *Владелец файла* ($UID_{процесса}$=$UID_{файла}$). Процессы владельца файла получают доступ к файлу в соответствии с заданными для него ограничениями. Владелец файла не имеет права сменить UID владельца. Владелец может сменить GID владельца файла только на идентификатор той группы, к которой сам принадлежит. Владелец файла может менять атрибуты доступа для всех трёх классов доступа.
2. *Группа владельца файла*. Если UID процесса не совпадает с UID файла, но идентификаторы группы совпадают, процесс имеет право на доступ с заданными для этого класса ограничениями. Пользователи, входящие в эту категорию, не могут изменить UID владельца, GID владельца и атрибуты файла.
3. *Все остальные*. Если UID процесса, пытающегося получить доступ к файлу, не совпадает с UID файла и не равен 0, GID процесса и файла также различаются, то применяются соответствующие ограничения типа доступа. Пользователи, входящие в эту категорию, не могут изменить UID владельца, GID владельца и атрибуты файла.

Процесс не получает доступ к файлу, если система отказывает в доступе *«x»* в одном из родительских для данного файла каталогов.

Атрибуты доступа обозначаются следующим образом. Если для определённого класса какой-либо из режимов доступа запрещён, ставится прочерк.

r **w** **x**	**r** **w** **x**	**r** **w** **x**
права доступа для владельца	права доступа для группы	права доступа для остальных

Строка «`rw-r--r--`» означает, что файл не является исполняемым, доступен владельцу для чтения и записи, остальные (в т. ч. группа владельца) имеют доступ только для чтения.

Строка «`r-xr-xr-x`» для обычного файла означает, что этот файл исполняемый и может быть запущен на исполнение любым пользователем, но не может быть модифицирован.

Строка «`rwxr-x--x`» для каталога означает, что владелец имеет право просматривать каталог, создавать, удалять и переименовывать файлы в каталоге, а также получать доступ к файлам в этом каталоге; пользователи из группы владельца файла могут просматривать каталог и получать доступ к файлам в этом каталоге; все остальные не имеют права просматривать каталог, но могут сделать его текущим и получать доступ к файлам в этом каталоге.

Если пользователю для каталога разрешён тип доступа «r» и «x», пользователь имеет право осуществлять в нём поиск файла (например, при помощи команды **`find`**), в противном случае, поиск невозможен.

2.3 Команды для работы с файлами и каталогами

`pwd` – вывод имени текущего каталога.

`cd` – смена текущего каталога. При использовании без аргументов команда переходит в домашний каталог пользователя (обычно /home/*имя_пользователя*). Альтернативное обозначение домашнего каталога пользователя ~ (тильда). Примеры:

- **`cd`** или **`cd ~`** – переход в домашний каталог пользователя;
- **`cd /`** – переход в корневой каталог;
- **`cd ..`** – переход в родительский каталог.

`ls` – просмотр содержимого каталога. Будучи запущенной без параметров, команда выдаёт лишь список файлов в текущем каталоге без каких-либо атрибутов. Ключ *-l* обеспечивает вывод основных атрибутов файла.

```
$ ls -l
итого 65180
drwxr-xr-x 2 pupkin pupkin     4096 Май 9 14:16 bmk/
-rw-r--r-- 1 pupkin pupkin  1033427 Фев 22 2017 etc.tgz
lrwxrwxrwx 1 pupkin pupkin       13 Сен 1 21:36 home -> /home/pupkin/
-rw-r--r-- 1 pupkin pupkin 65626430 Фев 22 2017 home.rar
```

В первой строке выводится число блоков, занятых файлами из этого каталога. Затем идут строки с подробной информацией о файле. Первый знак описывает тип файла (табл. 2.2). Затем следуют атрибуты доступа. Следующий столбец содержит количество жёстких связей для данного файла. Третий столбец содержит имя владельца файла. Четвёртый столбец содержит группу владельца. Пятый столбец содержит размер файла. В разных системах по умолчанию размер файла может выводиться в блоках (тогда следует добавить ключ *-h*) или байтах. Затем следует время создания или последней модификации файла. Последний столбец содержит имя файла. Для символических связей в последнем столбце (после знака –>) также указывается имя файла, на который данная связь ссылается.

Таблица 2.2. Расшифровка класса файла

–	Обычный файл
d	Каталог
l	Символическая ссылка
b	Файл блочного устройства

c	Файл символьного устройства

du – оценка пространства, занимаемого файлом или каталогом. Команда **du** выдаёт отчёт об использовании дискового пространства заданными файлами, а также каждым каталогом иерархии подкаталогов каждого указанного каталога. Опция *-s* заставляет du выводить только суммарный размер для указанных аргументов, не расписывая их по подкаталогам. Опция *-h* определяет вывод значений в наиболее удобных единицах.

```
$ du -hs mplayer
25M     mplayer
$ du -h mplayer
968K    mplayer/skins
14M     mplayer/codecs
25M     mplayer
```

quota отображает лимит дискового пространства для пользователя, а также текущий объём, занимаемый файлами данного пользователя.

cp – копирование файлов. Для рекурсивного копирования каталогов следует использовать ключ *-R*. Если указано несколько файлов или каталогов, последнее имя является именем каталога назначения. Пример:

```
cp a1.txt a2.txt a3.txt /home/pupkin
cp *.txt /home/pupkin
cp –R /home/pupkin /mnt/floppy
```

rm – удаление файлов. Для рекурсивного удаления используется ключ *-R*.

```
rm a1.txt a2.txt a3.txt
rm –i *.txt
```

mv – перемещение (переименование) файла.

mkdir – создание каталога.

rmdir – удаление каталога. Удаляет подкаталог, имя которого задаётся аргументом. Удаляемый каталог должен быть пустым.

chown – смена владельца файла. Синтаксис:

chown [*ключи*] *пользователь*[**:***группа*] *файлы*

Например:

```
chown –R pupkin:pupkin /home/pupkin
```

Приведённая команда рекурсивно (ключ *-R*) устанавливает для всех файлов и подкаталогов папки /home/pupkin владельца pupkin и группу владельца pupkin.

chgrp – смена группы владельца файла. Синтаксис:

chgrp [*ключи*] *группа файлы*

chmod – смена атрибутов доступа файла. Синтаксис:

chmod [*ключи*] *атрибуты файлы*

Атрибуты могут задаваться в символьном или цифровом виде. Формат элемента символьного определения атрибутов доступа:

[u g o a][+ – =][rwx]

Первая часть элемента определяет класс доступа: **u** – владелец (user), **g** – группа (group), **o** – остальные (other), **a** – все (all, эквивалентно комбинации **ugo**). Если первая часть отсутствует, используется **a**, но с учётом значения **umask** (см. ниже). Вторая часть определяет, как меняется режим доступа: **+** – добавляется режим доступа, **–** – удаляется режим доступа, **=** – режим доступа устанавливается точно, как описано третьей частью. Третья часть определяет, с какими видами доступа производится операция. В команде можно использовать несколько таких элементов, разделяя их запятыми без пробелов.

Примеры:

- **chmod a+rw my.txt** – разрешить всем любой доступ к файлу my.txt;
- **chmod ug=r,o–rwx my.txt** – владельцу и группе устанавливается доступ только для чтения, всем остальным доступ запрещается;
- **chmod –R +r ./** – разрешить всем (с учётом **umask**) чтение из файлов и подкаталогов текущего каталога (рекурсивная операция: ключ *-R*).

Атрибуты можно задать в виде восьмеричного числа. Биты этого числа сопоставляются режимам доступа следующим образом:

8	7	6	5	4	3	2	1	0
$\mathbf{r_u}$	$\mathbf{w_u}$	$\mathbf{x_u}$	$\mathbf{r_g}$	$\mathbf{w_g}$	$\mathbf{x_g}$	$\mathbf{r_o}$	$\mathbf{w_o}$	$\mathbf{x_o}$

Примеры:

- **chmod 0666 my.txt** – файлу my.txt назначаются атрибуты «rw-rw-rw-»;
- **chmod 0751 mydir** – каталогу mydir назначаются атрибуты «rwxr-x--x»;
- **chmod 0600 private.txt** – файлу private.txt назначаются атрибуты «rw-------».

При создании нового файла система назначает ему атрибуты доступа в соответствии с маской, задаваемой встроенной командой **umask**. При вызове без параметров команда выводит текущую маску. Маска может быть задана в символьном виде (как для команды **chmod**) либо в виде восьмеричного числа. В последнем случае устанавливаются биты, которых не должно быть в атрибутах создаваемого файла.

```
$ umask
0002
$ umask –S
u=rwx,g=rwx,o=rx
$ touch 1.txt
$ ls –l 1.txt
-rw-rw-r-- 1  pupkin  pupkin     0  Сен 17 21:33  1.txt
$ umask 0022
$ ls > 2.txt
$ ls –l 2.txt
-rw-r--r-- 1  pupkin  pupkin   223  Сен 17 21:34  2.txt
```

cat – вывод файла на стандартный вывод. Если указано несколько файлов, они выводятся последовательно. Если в командной строке файлы не указаны, данные берутся со стандартного ввода.

more выдаёт содержимое указанных файлов поэкранно. После заполнения экрана программа приостанавливается и ожидает нажатия клавиши: [пробел] – показать следующий экран, [Enter] – показать следующую строку, [q] – выход из программы и др.

less – улучшенный вариант **more**. Допускает прокрутку не только вперёд, но и назад. Другие команды: [<] – начало файла, [>] – конец файла, [/] – поиск вперёд по шаблону, [?] – поиск назад по шаблону, [n] – продолжить поиск, [N] – продолжить поиск в обратном направлении, [h] – вывести подсказку.

grep – поиск по шаблону. Если файл не указан, поиск осуществляется в данных, поступающих на стандартный ввод программы. Строки, содержащие указанный шаблон, выводятся на экран. Синтаксис команды:

grep [*ключи*] *шаблон* [*файлы*]

Например, чтобы вывести все строки, содержащие «student» в файле /etc/passwd, следует выполнить следующую команду:

```
$ grep student /etc/passwd
student:x:500:500::/home/student:/bin/bash
```

Ключ –E позволяет задавать шаблон в виде расширенных регулярных выражений. В данном случае поиск осуществлялся в данных, поступивших на стандартный ввод. Ввод данных завершается при нажатии на [Ctrl]+[d]. *Разберитесь, почему шаблону соответствуют только две строки?*

```
$ grep -E '[^[:alnum:]]{2,3}ex+' >1.txt
23ex+
--exxx
???ex
#exx
...e
^D
$ cat 1.txt
--exxx
???ex
```

find – поиск файлов по различным критериям. Синтаксис:

find [*путь поиска*] [*выражение*]

Параметр *путь поиска* задаёт начальную точку дерева каталогов, с которой осуществляется поиск (т. е. поиск будет проходить только в указанных подкаталогах). *Выражение* составляется из различных опций, определяющих критерий поиска:

- -name *шаблон* – поиск файла по шаблону имени файла;
- -user *пользователь* – поиск файлов, принадлежащих указанному пользователю;
- -size *тип число* – поиск по размеру и т. д.

По умолчанию имена файлов, удовлетворяющих критериям поиска, выводятся на экран. Можно также указать опции для осуществления определённых операций над найденными файлами. Для получения более подробной информации используйте **man**.

Примеры:

- **`find / -name '*conf*'`** – поиск всех файлов и каталогов, в названиях которых встречается последовательность символов conf;
- **`find /usr -user pupkin`** – поиск всех файлов в подкаталогах каталога /usr, владельцем которых является пользователь pupkin.

2.4 Файловый менеджер GNU Midnight Commander

Удобный интерфейс для навигации по файловой системе и управления файлами и каталогами предоставляет файловый менеджер GNU Midnight Commander[3] (запускается командой **`mc`** или **`midc`**). Его интерфейс аналогичен программам типа FAR, Norton Commander и т. п. (рис. 2.1). Перемещение по папкам осуществляется стрелочками [↓], [↑] и кнопкой [Enter]. Выбор активной панели – [Tab]. Клавиши [F3], [F4], [F5], [F6], [F8] используются для просмотра, редактирования, копирования, переименования и удаления выбранного файла. [F7] – создание подкаталога, [F9] – вход в меню (рис. 2.1, поз. 1), [F10] – выход из Midnight Commander.

Файлы разных классов подсвечиваются различным цветом, кроме того, Midnight Commander добавляет перед именем специального файла определённый символ (который не является частью имени), характеризующий его класс: каталог – имя белого цвета, начинается со слэша (/) (рис. 2.1, поз. 2); исполняемый файл – имя зелёного цвета, начинается с астериска (*) (рис. 2.1, поз. 5); символическая связь – начинается с тильды (~), если ссылается на каталог, или с «эт» (@), если ссылается на обычный файл (рис. 2.1, поз. 3); обычный файл – имя серого цвета. Если имя файла слишком длинное, на панели выводится начало и конец имени файла, соединённые тильдой (~) (рис. 2.1, поз. 4), а полное имя может быть отображено в информационном поле (рис. 2.1, поз. 6).

Midnight Commander поддерживает работу с архивами, как с каталогами, т. е. для просмотра архива надо в него «войти» [Enter]. Извлечение файлов из архива достигается «копированием» [F5]. Стандартные для Linux типы архивов: .tgz (.tar.gz), .tar.bz2, а также мультиплатформенные .zip, .rar, .arj, .lhz и др. при условии, что соответствующие команды установлены в системе.

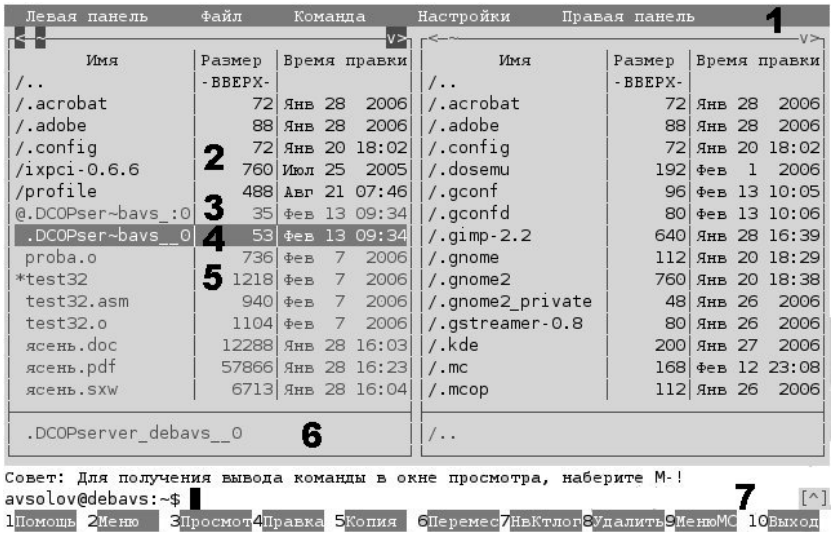

Рис. 2.1. Интерфейс программы Midnight Commander

Клавиша [Esc] в текстовых терминалах используется для задания управляющих последовательностей. Например, если вы работаете с терминалом, который не поддерживает функциональные клавиши [F1], [F2], …, то соответствующие команды для Midnight Commander можно подать при помощи [Esc]: [Esc],[1][4] соответствует [F1], [Esc],[2] соответствует [F2], … [Esc],[0] соответствует [F10]. Чтобы выйти из диалоговых окон **`mc`**, приходится нажимать [Esc] дважды.

Множество функций в Midnight Commander доступно через комбинации клавиш, начинающихся с клавиши [Meta]. На клавиатуре IBM PC-совместимого компьютера её роль выполняет [Alt]. Т. е. для вызова функции меню «Файл», «Quick CD» (смена каталога) в подсказке написано «M-c», что соответствует комбинации [Alt]+[c]. Для выполнения команды с отображением её стандартного вывода в окне просмотра («Файл», «Filtered View») указана комбинация «M-!», что соответствует [Alt]+[Shift]+[1].

Команды можно набирать в командной строке (рис. 2.1, поз. 7). Чтобы вставить в эту командную строку имя выделенного на панели файла, надо нажать [Esc],[Enter]. Midnight Commander имеет собственные средства хранения истории команд, которая вызывается комбинацией [Alt]+[h]. Выполнив команду, Midnight Commander может вывести приглашение нажать любую клавишу, прежде чем вновь отобразить панели, тем самым закрыв

[3] В системах BSD популярен файловый менеджер Demos Commander (**`deco`**), который обладает сходными возможностями.

[4] Это означает, что клавиши нажимаются последовательно: сначала [Esc], затем [1].

вывод только что завершившейся команды. Однако эта возможность определяется типом терминала и настройкой заданной через меню «Настройки», «Конфигурация», пункт «Пауза после выполнения…».

```
[pupkin@somehost homedir]$ ls
Desktop      tmp          my.txt
Для продолжения нажмите любую клавишу...
```

Другой способ скрыть панели и просмотреть вывод команд – при помощи комбинации [Ctrl]+[o]. Обратное включение панели – также [Ctrl]+[o]. В режиме, когда панели скрыты, пользователю доступна командная строка оболочки, в которой действуют стандартные для неё комбинации клавиш (история – [↑] [↓], автозавершение – [Tab] и др.). Если пользователь начнёт ввод команды в этой командной строке, то запуск команд из командной строки режима панелей становится недоступен – Midnight Commander в таком случае выдаст предупреждение (рис. 2.2). Проблема устраняется, если переключится на второй командный интерпретатор (скрыть панели при помощи [Ctrl]+[o]) и завершить начатую там команду, нажав [Enter].

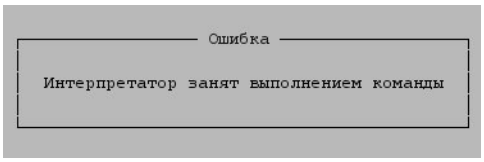

Рис. 2.2. Попытка запустить команду из режима панелей,
если ввод команды во второй оболочке не завершён

Midnight Commander не обновляет информацию на панелях автоматически, если содержимое текущего каталога было изменено в процессе выполнения такой команды. Чтобы перечитать каталог, надо нажать [Ctrl]+[r].

2.5 Контроль доступа к файлам на основе POSIX ACL

Кроме традиционной схемы управления доступом к файлам на основе трёх классов доступа (владелец, группа владельца, остальные) в UNIX может быть использована схема на основе Access Control Lists (ACL) – списков контроля доступа. Стандартизацией этой схемы занимаются группы POSIX 1003.1e и 1003.2c. Поддержка POSIX ACL внедрена в Linux, начиная с ядра 2.5.46. Существуют патчи для ядер 2.4.x, дополняющие Linux этой функциональностью. В отличие от традиционной схемы управления доступом, где режимы доступа задаются только для трёх классов, в ACL каждый элемент списка содержит набор прав доступа. Количество элементов в ACL ограничено только технической реализаций конкретной ФС (в EXT2/EXT3 размер ACL для каждого файла не должен превышать 1 блок (1–8 Кб), в других ФС – до 64 Кб). В ACL могут быть элементы следующих типов:

- *владелец* (только один элемент, содержит права доступа владельца из традиционной схемы),
- *названный пользователь* (множество элементов),
- *группа-владелец* (только один элемент, содержит права доступа группы владельца из традиционной схемы),
- *названная группа* (множество элементов),
- *маска* (только один элемент),
- *прочие* (только один элемент, содержит права доступа прочих из традиционной схемы).

Расширением традиционной схемы являются элементы «названный пользователь» и «названная группа», которые дополняют класс доступа «группа владельца»[5]. При использовании ACL традиционный класс доступа «группа владельца» отражает максимально допустимый набор режимов доступа для элементов ACL типов «названный пользователь», «группа-владелец» и «названная группа». Такой максимально допустимый набор задаётся элементом «маска». Если какой-либо из элементов этого класса содержит режим доступа, не указанный в маске, этот режим игнорируется (не предоставляется). Маска не действует на другие классы доступа («владелец» и «прочие»).

Алгоритм контроля доступа POSIX ACL:

1. Если UID процесса совпадает с UID файла, применяется элемент ACL «владелец».
2. Если UID процесса совпадает с одним из UID элементов «названный пользователь», применяется этот элемент с учётом маски.
3. Если GID процесса совпадает с GID файла, применяется элемент ACL «группа-владелец» с учётом маски.
4. Если GID процесса совпадает с одним из GID элементов «названная группа», применяется этот элемент с учётом маски.
5. В противном случае применяется элемент ACL «прочие».

[5] Обратите внимание, что в класс доступа «группа владельца» в терминологии управления доступом POSIX ACL включаются элементы «названные пользователи» и «названные группы», которые в терминах учётных записей UNIX могут не входить в группу владельца файла.

Выбранный по указанному алгоритму элемент ACL определяет, разрешить запрашиваемый доступ к файлу или запретить. Таким образом, элементы ACL можно использовать как для явного разрешения доступа выбранным пользователям или группам, так и для явного запрещения.

Следующие схемы поясняют действие ACL.

Минимальный набор элементов ACL, полное соответствие традиционной схеме:

ACL	традиционная схема

`user::rw-`
`group::r--`
`other::---`
	`rw- r-- ---`

Расширенный набор элементов ACL и его сопоставление традиционной схеме:

ACL	традиционная схема

`user::rw-`
`user:pupkin:rw-`
`group::r--`
`mask::rw-`
`other::---`
	`rw- `**`rw-`**` ---`

Если в примере справа добавить в ACL ещё один элемент:

`user:ivanov:rwx`,

то эффективными правами для него будут только «r» и «w», т. к. «x» для данного класса запрещён элементом «маска».

Для каталогов в ACL могут быть ещё включены элементы с пометкой default (назначаемые по умолчанию). Эти элементы не участвуют в алгоритме контроля доступа, а используются для назначения ACL дочерним элементам данного каталога при их создании.

Для манипулирования ACL используются команды:

- **`getfacl`** *`имя_файла`* – получить ACL указанного файла;
- **`setfacl -m`** *`acl_spec имя_файла`* – изменить ACL указанного файла;
- **`setfacl -x`** *`acl_spec имя_файла`* – удалить какой-либо элемент ACL.

Элементы ACL (*acl_spec*) задаются в следующем виде:

- **user:**[*uid*]**:**rwx – элемент «названный пользователь» или (если не указан uid) «владелец»,
- **group:**[*gid*]**:**rwx – элемент «названная группа» или (если не указан gid) «группа-владелец»,
- **mask**[**:**]**:**rwx – элемент «маска»,
- **other**[**:**]**:**rwx – элемент «прочие».

Примеры:

```
$ umask 0027
$ mkdir dir
$ ls -dl dir
drwxr-x---  2 student student 6 Sep  6 22:23 dir/
$ getfacl --omit-header dir
user::rwx
group::r-x
other::---
```

Этот пример показывает, что с каждым файлом всегда можно сопоставить минимальный ACL из трёх элементов.

```
$ setfacl -m user:netuser:rwx dir
$ getfacl --omit-header dir
user::rwx
user:netuser:rwx
group::r-x
mask::rwx
other::---
$ ls -dl dir
drwxrwx---+ 2 student student 6 Sep  6 22:23 dir/
```

В последнем примере разрешается полный доступ пользователю netuser. Элемент «маска» отражает все возможные режимы доступа для класса «группа владельца». При появлении расширенных элементов в списке ACL стандартные команды начинают показывать именно маску для класса доступа «группа владельца», кроме того, некоторые системы добавляют знак “+” в конце поля прав для индикации наличия расширенных элементов ACL.

Контрольные вопросы и задания

1. Просмотрите содержимое вашего каталога. Создайте внутри ещё один с любым именем. Скопируйте туда какие-нибудь текстовые файлы из общих папок и опробуйте команды перемещения и удаления.

Просмотрите файлы командами **cat**, **more**, **less** и сравните результаты. Просмотрите информацию из справочного руководства по используемым командам.

2. Скрытые файлы (имя начинается с точки) при просмотре командой **ls –l** не выдаются на экран. Выясните, как можно просмотреть всё содержимое каталога.
3. Создайте пробный файл любого формата с именем test_file и поэкспериментируйте на нём с применением команды **chmod**. Каков результат команд:

```
chmod u+w,og+rx test_file
chmod 744 test_file
chmod 640 test_file
```

4. При помощи команды **grep** найдите в файле /etc/passwd всех пользователей с домашним каталогом в папке /home.
5. Объясните, что делают команды, и что будет результатом их выполнения:

```
find /tmp –type d
find /var –name log -ls
find /usr/src/linux –follow –name '*.c' –exec grep –l \
      foo '{}' ';'
```

6. С помощью команды **find** найдите в каталоге /usr все файлы, размер которых не превышает 4 Кб. На экран выведите имя файла и его размер в байтах.
7. Ознакомьтесь с работой файлового менеджера Midnight Commander. Выясните, как в нём выполняются операции копирования, перемещения файлов, создания каталогов, изменения атрибутов файлов. Изучите встроенный редактор.
8. Выясните, на какой элемент ACL влияет команда **chmod** при изменении прав доступа для класса «группа владельца» (например, **chmod g+w dir**). Сравните результат этой команды при применении к файлу с минимальным ACL и при применении к файлу с расширенными элементами ACL.
9. Выясните, какой параметр требуется указать в командной строке **setfacl**, чтобы она не изменяла элемент «маска» при добавлении новых элементов в ACL.
10. В приведённом справа примере выделите элементы, для которых эффективный режим доступа будет отличаться от указанного в элементе. Как в этом случае будут отображаться атрибуты доступа командой **ls**? Какой тип доступа могут получить пользователи petrov и ivanov, входящие в группу users?

```
# owner: student
# group: student
user::rwx
user:petrov:rwx
group::rwx
group:users:---
mask:r--
other:rwx
```