ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Предлагаемый цикл лабораторных работ охватывает большую часть изложенного теоретического материала и направлен на его практическое закрепление. Учебные исследовательские задачи, содержащиеся в каждой из лабораторных работ, должны способствовать формированию у обучаемых творческого отношения к защите компьютерной информации, без чего невозможно решение сложных практических задач администрирования операционных систем.

ОБЩИЕ ТРЕБОВАНИЯ

1. Практикум рассчитан, как минимум, на 28 часов лабораторных занятий.
2. Перед проведением лабораторных занятий обучаемые должны прослушать теоретический курс в объеме настоящего пособия либо освоить его самостоятельно. Непосредственно перед проведением каждой работы преподаватель по своему усмотрению организует в устной или письменной форме допуск обучаемых к самостоятельному выполнению работ.
3. Для проведения занятий необходим компьютерный класс из расчета одно автоматизированное рабочее место на базе персонального компьютера на каждого пользователя. Сложные задания по усмотрению преподавателя разрешается проводить подгруппами в составе двух человек.

На каждом компьютере должна быть инсталлирована одна из распространенных версий операционных систем GNU Linux. Желательно использовать дистрибутивы последних версий с последними версиями ядра операционной системы, что обеспечит поддержку наиболее передового аппаратного обеспечения компьютера. Если инсталляция программного обеспечения на жесткий диск невозможна, рекомендуется использовать загружаемые рабочие системы, установленные на компакт-диске (так называемый Live-CD, содержащий полноценную версию ОС).

Перед выполнением конкретного задания на каждом рабочем месте должны быть созданы учетные записи пользователей, предусмотренные заданием на лабораторную работу. При выполнении некоторых заданий обучаемые наделяются правами суперпользователя и создают учетные записи самостоятельно.

В ходе занятий используются встроенные команды интерпретатора **`/bin/bash`** (Bourne Again Shell) и штатные утилиты операционной системы. Для исследования структуры файловых систем автором рекомендуются свободно распространяемая утилита **`extview`** (автор – Желтышева Е.Д.), и сценарий **`extv`** на языке Perl (автор – Пирожкова М.В.). Кроме того, в работах 2 и 4 используются две утилиты, созданные М.Э. Пономаревым.

Формой контроля является письменный отчет и защита работы.

ПАМЯТКА ОБУЧАЕМЫМ

- Все занятия проводятся в самостоятельном режиме в соответствии с пунктами задания. Задания изложены в логической последовательности, и изменять порядок их выполнения не рекомендуется. В случае затруднений обращайтесь к преподавателю.
- Все операции с файлами и каталогами производите только в режиме текстовой консоли или эмуляции текстового терминала. Там, где это предусмотрено заданием, используйте возможности файлового менеджера **Midnight Commander**.
- Обучаемые при проведении лабораторных работ должны работать в нескольких текстовых консолях. Переключая консоль, вы можете поочередно работать с объектами операционной системы от имени нескольких пользователей и администратора системы. Переключение текстовой консоли производится комбинацией клавиш **Alt-Fn**, где **n** – номер консоли.
- Наибольшие затруднения и наиболее частые ошибки у начинающих, а также пользователей, не искушенных в интерфейсе командной строки, вызывает ввод команд. Если образец команды предусмотрен заданием, постарайтесь разобраться с ее синтаксисом и назначением элементов командной строки. Многие команды даны с указанием объектов в угловых скобках, например **<dev>**. Не копируйте команды механически и правильно подставляйте в них нужные параметры. Перед тем как завершить введеную команду клавишей **<Enter>**, проверьте правильность ее написания. В вашем распоряжении краткий справочник по командам, приведенный в приложении, примеры команд, сопровождающие текст пособия, а также электронные руководства **man** и **info**.
- При выполнении ряда заданий вам придется работать с системой, имея полномочия суперпользователя. Будьте особо внимательны при вводе команд. Помните, что операционная система, как правило, не контролирует целесообразность команд суперпользователя и ваше ошибочное или необдуманное действие может привести к краху системы. Ориентируйтесь на характерную форму приглашения к вводу команд (**$** – пользователь, **#** – администратор). Не бравируйте своим умением быстрого ввода команд. Наиболее ответственные команды, которые могут привести к разрушению операционной системы или тотальному стиранию данных с машинных носителей, перед запуском не стесняйтесь показать преподавателю.
- Автор намеренно усложнил отдельные задания. В некоторых случаях обучаемым предлагается выполнить действия, заведомо приводящие к ошибкам. Например, предлагается создать, прочитать или удалить файл из каталога, который для него недоступен. Обучаемый должен обнаружить причину отказа в доступе, отразить ее в отчете, а затем выполнить

задание, изменив права на доступ. Подобные «ошибки» способствуют более прочному закреплению учебного материала.

- Ответы на поставленные вопросы (с указанием пунктов задания) заносите в отчет, который можно вести в произвольной форме в рукописном или электронном виде.
- Защита выполненных работ производится индивидуально при предъявлении отчета. Основная форма защиты – ответы на вопросы тестового задания (в виде программированного контроля). Преподаватель может по своему усмотрению изменять форму защиты.

ЛАБОРАТОРНАЯ РАБОТА № 1
«Исследование файловых объектов с правами пользователя»

Выполнение работы

1. Изучите теоретический материал, изложенный в параграфах 1.3, 3.1 и 3.5 учебного пособия.
2. Зарегистрируйтесь в первой консоли как **root** с паролем, который вам будет сообщен преподавателем.
3. Для исследования файловых объектов вам потребуются еще два пользователя, учетные записи которых предстоит создать. Для этого воспользуйтесь краткой формой команды

```
useradd -m user1
```

Аргумент **-m** предписывает создание домашнего каталога пользователя с его именем. После этого пользователю необходимо присвоить пароль. Это делается командой

```
passwd user1
```

На запрос введите простой пароль 1234567890.

4. Повторите команды п. 3 для создания учетной записи пользователя **user2**. Помните, что оба пользователя по умолчанию являются членами одной группы **users**.
5. Аналогично откройте третий текстовый терминал и зарегистрируйтесь как **user1**.

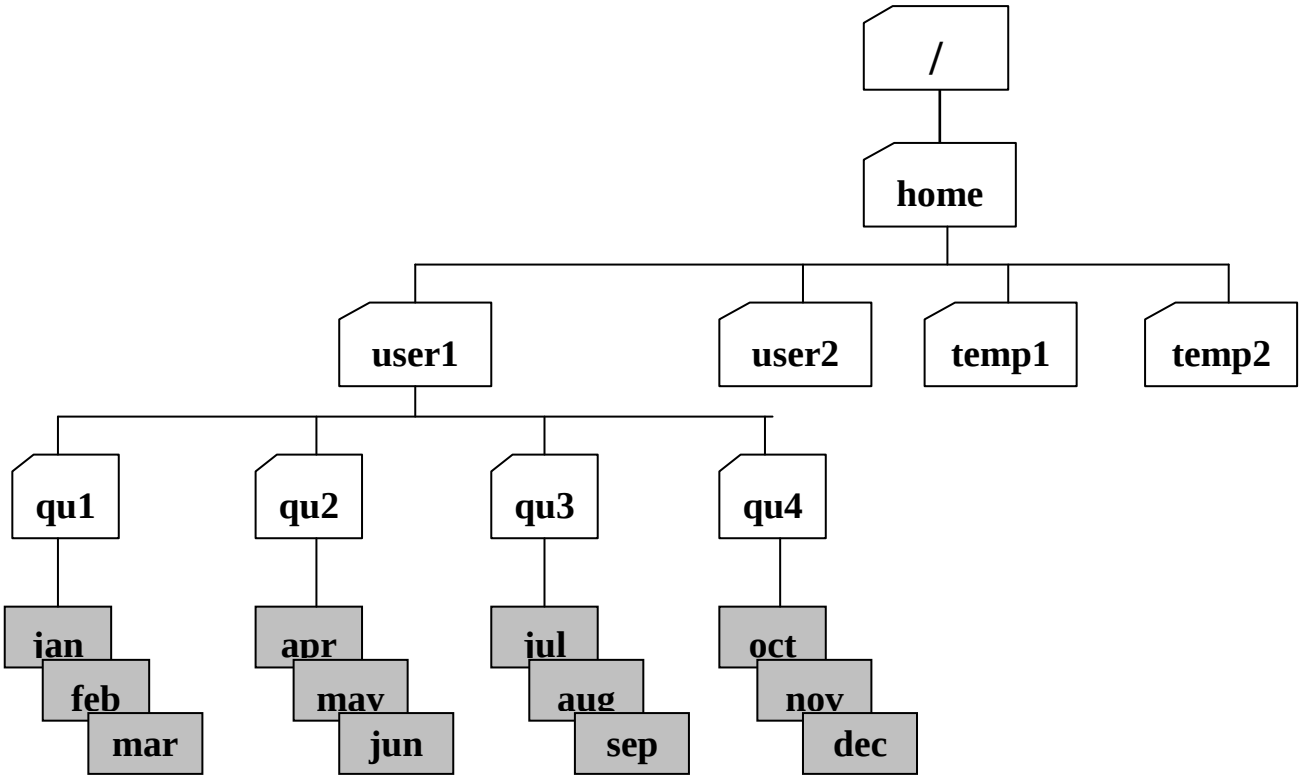

Рис. 1. Пояснение к пп. 10–13

6. С помощью **Alt-F2** откройте второй текстовый терминал и зарегистрируйтесь как **user2** с паролем 1234567890.
7. Нажатием **Alt-F1** вернитесь в первую консоль. Теперь, переключая консоль, вы можете работать с объектами операционной системы от имени двух разных пользователей и администратора системы. Основная часть задания выполняется с правами обычных пользователей. Переходите в первую консоль и используйте права **root** только при выполнении соответствующих пунктов задания. Будьте внимательны при вводе команд! Помните, что операционная система не контролирует действий администратора и ваше ошибочное действие может привести к краху системы. Ориентируйтесь на характерную форму приглашения к вводу команд, в котором отражены ранг и имя пользователя.
8. С правами **user1** с помощью команды **cd /root** попробуйте войти в каталог суперпользователя. Объясните результат. Затем с помощью команды **ls -la /** просмотрите список основных каталогов и укажите, в какие из них вы имеете право войти.
9. Проверьте права доступа к домашним каталогам пользователей **/home/user1** и **/home/user2**: они должны быть установлены в 755.
10. Переключитесь в консоль администратора и создайте два новых временных каталога:

```
mkdir -m 777 /home/temp1
```

и

```
mkdir -m 1777 /home/temp2
```

11. Вернитесь в консоль **user1** и, пользуясь командой **mkdir,** создайте в домашнем каталоге пользователя **/home/user1** четыре каталога с именами **qu1, qu2, qu3, qu4**. При создании каталогов объявите следующие права доступа к ним: (**qu1** - 777, **qu2** - 404, **qu3** - 1333, **qu4** – 505). Пример:

```
cd; mkdir -m 777 qu1
```

С помощью команды **ls /home/user1** убедитесь в том, что каталоги созданы. Какие из предоставленных прав кажутся вам лишенными смысла? Почему?

12. Задайте права доступа к файлам «по умолчанию». Для этого установите маску доступа **umask 022.** Поясните, какие права к вновь создаваемым файлам и каталогам будут предоставляться пользователю, членам его группы и остальным.
13. В каждом из каталогов создайте по три текстовых файла с именами (**jan, feb, mar**), (**apr, may, jun**), (**jul, aug, sep**), (**oct, nov, dec**). В каждый файл запишите календарь на определенный месяц текущего года. Например, команда **cal 1 2011 >jan** создает в текущем каталоге файл **jan** и записывает в него календарь на январь

2011 года. Не забывайте, что использование относительного (короткого) имени файла требует, чтобы вы находились в нужном каталоге. В противном случае следует указывать полный путь к создаваемому файлу. Для навигации по каталогам используйте команды **cd** и **pwd.** В каком случае создание файлов не удалось? Почему?

14. С помощью команды **chmod** установите нужные права доступа в «недоступные» каталоги **qu2**, **qu4** и создайте там указанные файлы. После этого верните каталогам прежние права доступа.
15. С помощью команд **cd** и **ls** войдите в каждый из созданных каталогов и просмотрите список созданных файлов. При просмотре содержимого каталогов используйте два режима: **ls** без аргументов и **ls -l**. В каких случаях не удалось войти в каталог? В каких случаях не удалось посмотреть список файлов? Почему?
16. Прочитайте содержимое одного из файлов в «темном» каталоге (например, **cd /home/user1/qu3; cat aug**). Сделайте выводы.
17. Перейдите во 2-ю консоль и с правами пользователя **user2** войдите в каталог **/home/user1/qu1.** Создайте в каталоге **/home/user2** новый файл **quart1** путем конкатенации нескольких имеющихся:

```
cat jan feb mar >/home/user2/quart1
```

С помощью команды **file** определите тип созданного файла. Попробуйте вывести его на экран командой **cat**. Что представляет собой данный файл?

18. С помощью команды **chmod** установите права доступа 077 на созданный файл **quart1**. Вновь попробуйте прочесть его. Ответьте, почему владельцу файла запрещается доступ, если файл доступен для всех? Что необходимо сделать, чтобы вернуть владельцу права на доступ?
19. Установите для файла **quatr1** права на доступ 4700. Кому и какие права вы при этом предоставили? Как воспользоваться этими правами? Какие из предоставленных прав не имеют смысла?
20. Перейдите в консоль администратора и передайте право владения на файлы **may** и **aug** пользователю **user2** (команда **chown**). Поочередно из консолей **user1** и **user2** проверьте, как изменились права владения файлами после его передачи. Может ли пользователь **user2** воспользоваться предоставленными правами? (Пользователи **user1** и **user2** при создании учетных записей по умолчанию отнесены к одной группе **users.**)
21. Правами пользователя **user1** из каталогов **/home/temp1** и **/home/temp2** с помощью команды **ln** создайте две «жесткие» ссылки на файл **dec** с именами **dec_h1** и **dec_h2**

```
ln /home/user1/qu4/dec /home/temp1/dec_h1
```

Чем созданные ссылки отличаются от исходного файла?

22. С помощью команды **ln -s** создайте из каталогов **/home/temp1** и **/home/temp2** две символические ссылки на файл **dec** с именами **dec_s1** и **dec_s2**. Чем отличаются созданные ссылки от исходного файла? Попытайтесь прочитать содержимое файлов символических ссылок. Что они собой представляют?
23. Правами пользователя **user2** с помощью команды **cp** создайте в каталогах **/home/temp1** и **/home/temp2** по одной копии файла **dec** с другим именем (**dec_copy1**). Чем отличаются исходный файл и его копия (обратите внимание на то, кто является владельцем исходного файла и его копии)? Чем отличаются права доступа на эти файлы? Вернитесь в консоль **user1.**
24. С правами пользователя **user1** создайте жесткие ссылки из его домашнего каталога на файл **/bin/su** (доступен обычным пользователям только на исполнение) и на файл **/etc/shadow** (для обычных пользователей недоступен). Ответьте, какими правами на объектовый файл нужно обладать, чтобы создать на него жесткую ссылку? Какую выгоду получает обладатель жестких ссылок на недоступные файлы?
25. С правами пользователя **user1** скопируйте в его домашний каталог утилиту **/bin/mount**. Сравните между собой оригинал и копию и укажите все отличия. Сможет ли пользователь применить копию опасной утилиты во вред политике безопасности? Почему?
26. С помощью команды **rm** удалите файл **dec**. Что произошло с «жесткими» и символическими ссылками на данный файл? Что произошло с его копиями? Что нужно сделать для того, чтобы файл перестал существовать (на логическом уровне)?
27. Правами **user1** удалите файлы из каталогов **/home/temp1** и **/home/temp2.** Какие файлы не удалось удалить? Почему? Попробуйте удалить оставшиеся файлы правами пользователя **user2**. Объясните результат.
28. Попробуйте удалить любой из каталогов **qu1, qu2, qu3, qu4** с помощью команды **rmdir** (не удаляя предварительно из них файлов). Объясните результат.
29. Войдите в консоль администратора и с правами **root**, пользуясь командой **chattr,** заблокируйте файл **feb** от любых изменений (предварительно ознакомьтесь с синтаксисом команды). Установите параметр запрета любых операций, кроме добавления данных для файла **mar**. Вернитесь в консоль **user1**. С помощью команды **lsattr -l** проверьте наличие дополнительных атрибутов у файлов.
30. Правами пользователя **user1** добавьте одну строку **finish** в конец файлов **feb** и **mar** . Воспользуйтесь для этого командой

```
echo finish >> file_name
```

Убедитесь в успешном завершении операции, объясните результат.

31. Правами пользователя **user1** с помощью команды **rm -rf** последовательно удалите ранее созданные каталоги **qu2, qu3, qu4** вместе с файлами. Объясните результат.
32. С правами пользователя **user1** создайте в **/home/user1** два каталога **mkdir –m 400 src** и **mkdir –m 200 dst**. Внутри каталога **src** командой **echo 1234567890 > abc** создайте текстовый файл (если для этого прав доступа к каталогу недостаточно, временно измените их). Для созданного файла установите права доступа **chmod 100 abc**.
33. Манипулируя правами доступа на созданные файловые объекты, установите минимально необходимые права, позволяющие копировать и перемещать указанный файл из каталога **src** в каталог **dst**. Зафиксируйте результаты в отчете.
34. Определите, какие минимально необходимые права на файловые объекты нужно иметь, чтобы выполнять копирование и перемещение файла от имени администратора.
35. Понаблюдайте и зафиксируйте в отчете изменения временных отметок файлов и каталогов, происходящие при файловых операциях.
36. Ответьте на указанные преподавателем тестовые вопросы.
37. После успешной защиты лабораторной работы, получив разрешение преподавателя, из консоли суперпользователя удалите все созданные вами файлы. *Соблюдайте осторожность: с правами* **root** *и с помощью утилиты* **rm** *вы можете вызвать крах системы!*
38. После выполнения работы командами **exit** завершите пользовательские сеансы во второй и третьей консолях, а из первой консоли правами **root** выполните команду останова системы **halt**.

Контрольные вопросы

1. Какие дополнительные атрибуты можно присвоить файлу в файловой системе **ext*fs**? Как эти атрибуты влияют на обеспечение конфиденциальности, целостности и доступности информации?
2. Как можно создать текстовый файл без помощи текстового редактора?
3. В чем различие между копиями файла, его «жесткими» и символическими ссылками? Для чего используются символические ссылки?
4. Для каких целей могут использоваться «темные» каталоги?
5. Какие права по отношению к файлам и каталогам вам необходимо иметь для копирования файла? Как изменяются при этом атрибуты копии?
6. Администратор по невнимательности ввел следующую команду:

chmod –R 555 / . Что последует за выполнением этой команды?

ЛАБОРАТОРНАЯ РАБОТА № 2
«Исследование архитектуры файловых систем **ext*fs**»

1. Изучите теоретический материал, изложенный в главе 4, а также в параграфах 3.2 – 3.4 учебного пособия.
2. В Приложении к учебному пособию найдите и прочитайте справочный материал по использованию утилиты **extview**.
3. Зарегистрируйтесь в первой текстовой консоли с правами **root.** Пароль администратора узнайте у преподавателя.
4. С помощью **Alt+F2** откройте второй текстовый терминал и зарегистрируйтесь как пользователь **user1**.
5. Вспомните синтаксис команд **mount** и **umount**.
6. С помощью команды **which <file_name>** убедитесь в наличии на дисковом разделе Linux программных файлов **extview** (или **extv**) и **fillfile**. Если указанные файлы отсутствуют, смонтируйте предоставленный преподавателем сменный носитель к одному из подкаталогов **/mnt** и скопируйте с него поименованные файлы в каталог **/sbin**. Размонтируйте носитель (до размонтирования не извлекать!) и верните его преподавателю.
7. С помощью команды **ls -li /** просмотрите список файлов корневого каталога. В первом столбце указаны индексные дескрипторы файлов (**inode**). Почему номера индексных дескрипторов основных подкаталогов так сильно различаются? Можете ли вы из выведенной на экран информации определить размер логического блока на диске (1, 2 или 4 Кб)?
8. Аналогичным образом посмотрите начало списка файлов одного из каталогов, например:

```
ls -li /bin | more
```

   Сравните номера **inode** файлов, входящих в данный подкаталог. Какие вы можете сделать выводы относительно размещения файловых объектов на дисковом пространстве?
9. С помощью команды **cat** прочитайте файл **/etc/mtab**. Зафиксируйте в отчете смонтированные устройства и файловые системы, а также права на их монтирование. Запомните наименование логического раздела жесткого диска, на котором смонтирована файловая система **ext*fs** (например, **/dev/sda3**). К этому имени файла-устройства (далее оно будет обозначаться как **<dev>**) вы будете обращаться, запуская дисковый редактор **extview**.
10. Перейдите в каталог **/home**. С помощью команды

```
fillfile file_size symbol
```

   создайте файл, размер которого больше размера одного сектора, но

меньше размера одного блока. Например, с помощью команды **fillfile 600 b** вы создадите в текущем каталоге файл с именем **b** объемом 600 байтов, который содержит повторяющиеся символы **b**. С помощью таких файлов с определенным наполнением вам будет удобно наблюдать за дисковым пространством.

11. Командой **ls -li** просмотрите список файлов в текущем каталоге и найдите созданный файл. В первом столбце списка найдите и запишите номер индексного дескриптора файла. С помощью команды

extview -i <inode> <dev>

(вместо **<dev>** укажите нужный Linux-раздел на диске) выведите индексный дескриптор данного файла. По размеру файла и иным признакам убедитесь в том, что вы действительно наблюдаете **inode** созданного вами файла. Письменно ответьте на вопросы:

- К какому типу относится данный файл?
- Каковы права доступа на данный файл у владельца, группы владельца и прочих пользователей?
- Что означает число **block count**?
- Сколько блоков с непосредственной адресацией выделила система под данный файл?
- Запишите шестнадцатеричный номер блока данных этого файла.

12. С помощью команды

extview -b <block_number> <dev> | more

выведите поэкранный дамп блока данных файла. Убедитесь в том, что он действительно заполнен определенными повторяющимися символами.

13. Командой **rm <file_name>** удалите созданный файл. С помощью дискового редактора вновь посмотрите таблицу **inode** и блок данных данного файла. Что с ними произошло?
14. С помощью программы **fillfile** вновь создайте в текущем каталоге файл с другим именем и заполнением. Размер очередного файла должен быть немного меньше предыдущего (например, 500 байтов).
15. С помощью редактора посмотрите таблицу **inode** и блок данных повторно созданного файла. Убедитесь в том, что это действительно он. Остался ли в блоке данных информационный «мусор» от прежнего файла?
16. Если вы обнаружили, что утилита **extview** выводит сведения об уже удаленном файле, это означает, что содержимое дискового кэша, откуда берет информацию утилита **ls**, еще не записано на диск, с которым работает **extview**. Принудительно сохранить на диск вновь созданный файл можно путем перезагрузки системы (**reboot**) либо с помощью дисковой утилиты командой

hdparm –f <dev>

17. Вновь удалите созданный файл. Создайте третий файл с иным именем и размером более одного блока (например, 5000 байтов). Убедитесь в наличии и заполнении файла, зафиксируйте номер последнего блока данных. Вновь удалите файл.
18. Создайте четвертый файл с отличающимся именем и размером менее одного блока (например, 3000 байтов). После того как вы убедитесь в создании очередного файла, посмотрите содержимое последнего блока, оставшегося от третьего файла. Какие вы можете сделать выводы?
19. Создайте пятый файл с размером 10000 байтов. С помощью команды

chattr +s <file_name>

установите для данного файла дополнительный атрибут, обозначающий гарантированное стирание блоков данных при удалении файла. Убедитесь, что индексные данные, выводимые командой **extview**, отображают дополнительные атрибуты файла.

20. Удалите пятый файл, после чего проверьте ранее занятые им **inode** и блоки данных. Сделайте выводы и занесите их в отчет. (Поддержка некоторых дополнительных атрибутов файлов реализована не во всех версиях ОС.)
21. С помощью команды **extview -b 0 <dev> | more** запустите дисковый редактор на поэкранный вывод содержимого суперблока (блок номер 0). Первые 1024 байтов (**400h**) блока отведены для размещения загрузчика (**LILO** или **GRUB**), но могут пустовать, если загрузочная программа размещается в Master Boot Record (MBR). Суперблок начинается со смещения **400h** и имеет размер 1024 байтов. С помощью справочных данных о файловой системе **ext*fs**, приведенных в главе 4 учебного пособия и в Приложении 2, исследуйте содержимое суперблока. Обратите внимание на то, что дамп памяти выводит шестнадцатеричные слова в обратном порядке. Для перевода чисел из шестадцатеричной системы счисления в десятичную из пользовательской консоли запустите калькулятор **bc** и задайте его внутренней командой **ibase=16** шестнадцатеричный ввод чисел. Теперь, вводя шестнадцатеричное число (символы A–F должны быть заглавными!) и нажимая **<Enter>**, получаем его десятичный эквивалент. В результате исследования полей суперблока письменно ответьте на вопросы:

- Сколько файлов может быть создано в данной файловой системе?
- Чему равен размер одного логического блока?
- Каков размер одного индексного дескриптора?
- Какой объем диска зарезервирован для нужд суперпользователя?
- Сколько блоков на момент исследования занято?
- Когда в последний раз проверялась файловая система? Каковы результаты проверки?
- Какой номер у индексного дескриптора файлового журнала?

- Что содержит неиспользуемая часть суперблока?

22. Рассмотрите описатели групп блоков **Group Descriptors**, которые обычно располагаются в первом блоке. Выберите первый из описателей размером 32 байта (две строки по 16 байтов). Найдите в нем и запишите в отчет шестнадцатеричные адреса:

- битовой карты блоков (**block bitmap**),
- битовой карты индексных дескрипторов (**inode bitmap**),
- таблицы индексных дескрипторов (**inode table**).

23. По аналогии прочитайте описатель 10-й группы блоков. Найдите в нем адреса битовых карт блоков и индексных дескрипторов. С помощью **extview** и утилиты **dd** рассмотрите битовые карты и сделайте выводы об алгоритмах использования дискового пространства.
24. Посмотрите шестнадцатеричный дамп одного из индексных дескрипторов в **inode table** 10-й группы блоков (при использовании обычного 128–байтного **inode** записи в блоке идут с интервалом в **80h** байтов). С помощью таблицы, приведенной в главе 4 учебного пособия, определите и отразите в отчете:

- тип этого файла и права доступа к нему,
- идентификатор владельца (выяснить, кому он принадлежит),
- размер файла в байтах,
- число имен этого файла,
- наличие дополнительных атрибутов этого файла,
- число блоков, занимаемых файлом,
- номера блоков данных с непосредственной и косвенной адресацией.

25. С помощью редактора **extview** и утилиты **dd** просмотрите несколько блоков данных исследуемого файла.
26. По имеющимся у вас данным определите, **inode** какого файла вы рассматривали. Для этого следует определить порядковый номер **inode** в своей группе, а также суммарное число индексных дескрипторов в 9 пропущенных группах. Для нахождения имени (или имен) файла по его номеру используйте возможности утилиты **find** (см. справку о командах). Укажите в отчете полный путь к этому файлу.
27. С помощью дискового редактора посмотрите, что представляет собой файл каталога. Для этого, воспользовавшись командой **ls -li /**, найдите номер **inode** одного из подкаталогов первого уровня, откройте его с помощью **extview**, найдите номер первого непосредственно адресуемого блока и изучите его. Сравните с форматом каталога, приведенным в теоретической части пособия. Сделайте выводы.
28. Если вы исследуете журнализируемую файловую систему, найдите в суперблоке индексный дескриптор файлового журнала и откройте его для просмотра. Обратите внимание на объем и расположение журнала. Изучите содержимое блока косвенной адресации.
29. Аналогичным способом посмотрите, что представляет собой файл сим-

волической ссылки. Обратите внимание на то, где и в каком виде в файле символической ссылки хранится путь к целевому файлу. Если не сумеете найти ответ, найдите и рассмотрите шестнадцатеричный и символьный дампы индексного дескриптора символической ссылки. Сделайте выводы и отразите их в отчете.

30. Проведите наблюдения за временными отметками обычных файлов и каталогов. Для этого можно использовать права одного из пользователей, создав от его имени два каталога **time1** и **time2**. В одном из каталогов с помощью команды **echo** требуется создать несколько текстовых файлов. С помощью утилиты **stat** вывести и записать в табл. 1 все временные отметки созданных объектов. Какую информацию можно извлечь из временных отметок файлов?

Таблица 1

№ п/п	Команда	Файл			Каталог		
		Время A	Время C	Время M	Время A	Время C	Время M
1	**ls**						
2	**ls -l**						
3	**touch**						
4	**stat**						
5	**chmod**						
6	**chown**						
7	**mv**						
8	**cp**						

Контрольные вопросы

1. Как файловая система **ext2fs** операционной системы Linux использует дисковое пространство? Можно ли по индексным дескрипторам файлов и каталогов представить себе их взаимное расположение на жестком диске?
2. Какую роль в монтировании сменных носителей и логических разделов жесткого диска играет файл **fstab**? Чем отличаются автоматическое и «ручное» монтирование?
3. Почему дискету не следует извлекать из дисковода до ее размонтирования? Что произойдет в случае пренебрежения этим правилом?
4. В параметрах файла **fstab** могут быть указаны разрешения монтирования сменных машинных носителей для **user** и **users**. Чем отличаются эти разрешения?
5. Механизмы образования и удаления информационного «мусора» в файловых системах ОС Linux.
6. Экономно ли используется дисковое пространство в ОС Linux? Приведите доказательства своего ответа.
7. Пользователь может читать и записывать информацию на сменные машинные носители, даже если ему запрещено их монтировать. Как можно запретить использование сменных машинных носителей?
8. Какую опасность для системы представляют файлы, не имеющие владельца?
9. Как можно найти файл по номеру его **inode**?

ЛАБОРАТОРНАЯ РАБОТА № 3
«Восстановление данных программными средствами ОС Linux»

1. Изучите теоретический материал, изложенный в параграфах 3.3 – 3.4, 4.3 учебного пособия.
2. Зарегистрируйтесь в первом текстовом терминале с учетной записью **root**.
3. С помощью **Alt+F2** откройте второй текстовый терминал и зарегистрируйтесь как пользователь **user1**.
4. Используя права обычного пользователя, введите команду **fdisk –lu**. Повторите ввод команды с правами суперпользователя. Посмотрите, какие устройства долговременной памяти соединены с аппаратными интерфейсами персонального компьютера, из каких логических разделов состоит дисковая память и какие операционные системы на них установлены.
5. Создайте в файловой системе новый пустой каталог **mkdir /mnt/abcd**, который будет служить точкой монтирования. С помощью команды **locale** установите, какая из кодировок по умолчанию поддерживается загруженной операционной системой (**koi8-r, cp1251** и т. д.).
6. Используя команду

mount –t <type> –o iocharset=koi8-r <dev> <dir>,

где **<type>** – монтируемая файловая система (**ntfs, vfat** и др.),
<dev> – файл блочного устройства, соответствующий монтируемому логическому разделу (например, **/dev/sda3**),
<dir> – созданная точка монтирования (**/mnt/abcd**),
примонтируйте к дереву каталогов Linux файловую систему ОС Windows*. Опция **iocharset=koi8-r** указывает на используемую кодировку, что позволит просматривать имена файлов в неискаженном виде. После монтирования запустите редактор **Midnight Commander** и посмотрите каталоги и файлы примонтированной системы. Если русскоязычные имена файлов отображены неправильно, демонтируйте раздел в соответствии со следующим пунктом и попробуйте задать иную кодировку.

7. Демонтируйте файловую систему Windows* с помощью команды **umount <dev>** или **umount <dir>** (вместо **<dev>** или **<dir>** укажите конкретные значения). К моменту монтирования все каталоги и файлы смонтированной файловой системы должны быть закрыты.
8. Получите у преподавателя поврежденный сменный машинный носитель (дискету, оптический диск, USB-Flash) с информацией, которую необходимо спасти. С помощью команды **dd** поблочно скопируйте часть дисковой памяти в файл **/home/abcd** (этот файл будет создан).

Количество копируемых блоков зависит от состояния поврежденного носителя. Обязательно укажите последнюю опцию команды **conv=noerror**, позволяющую пропускать отсутствующие и поврежденные блоки. Рекомендуемый синтаксис команды

```
dd if=/dev/fd0 of=/home/floppy bs=512 count=10
conv=noerror,fsync.
```

Размер полученного файла должен соответствовать суммарному объему правильно прочитанных секторов дискеты. С помощью редактора (F3) файлового менеджера **Midnight Commander** или команды **mcedit** прочитайте созданный файл-образ машинного носителя. Убедитесь в том, что информация скопирована достоверно (в противном случае попытку создания файл-образа придется повторить).

9. Пользуясь командой **dd** , запишите произвольный текст в первый сектор анализируемой дискеты. Например:

```
echo 1234567890 | dd of=/dev/fd0 bs=1 seek=10 count=10
```

Затем просмотрите первый сектор дискеты командой **cat /dev/fd0 | more** и найдите следы своей записи.

10. С помощью команды

```
dd if=/dev/hda bs=512 count=1|xxd|more
```

выведите на монитор содержимое первого сектора главной загрузочной записи и рассмотрите ее. После того как вы убедились в том, что видите именно Master Boot Record (MBR), скопируйте загрузочную запись в файл. Обычно подобные копии создаются на сменном носителе и предназначены для использования в случае порчи жесткого магнитного диска. Запишите в тетради команду обратного копирования (ее вводить не нужно!) и покажите преподавателю. Будьте внимательны при вводе команд, иначе вы можете сделать жесткий диск недоступным!

11. При наличии у вас собственного носителя на полупроводниковой памяти USB-Flash (кафедра не может обеспечить учебный процесс подобными устройствами) примонтируйте его к файловой системе. После подключения носителя к USB-интерфейсу запускается команда **fdisk –lu**. Допустим, отображается единственный раздел **/dev/sda1**. Он монтируется командой

```
mount –t vfat –o iocharset=koi8-r /dev/sda1 /mnt/usb
```

(кодировка символов может быть иной, а вместо каталога **/mnt/usb** можно указать другой пустой каталог). С помощью команды **cat** или файлового менеджера просмотрите файлы на примонтированном носителе.

12. Командой **umount /mnt/usb** отключите устройство и извлеките его. Путем просмотра файлов в каталоге **/var/log** найдите запись, которая зафиксировала факт подключения мобильного носителя (предполо-

жительно это файл **/var/log/message**).

13. С помощью команды **cat /root/.bash_history** просмотрите текстовый файл истории команд. Найдите в нем команды, введенные вами. Каким образом их лучше удалить? Если вы вводили команды от имени одного из зарегистрированных пользователей, то аналогичный файл истории команд вы найдете в его домашнем каталоге.
14. Вспомните, в каком порядке происходит логическое удаление файлов в файловых системах **ext2fs** и **ext3fs**.
15. Используя утилиту **fillfile**, по методике, изложенной в лабораторной работе № 2, создайте в рабочем каталоге пользователя пять текстовых файлов с именами от “A” до “E” размером от 500 до 10000 байтов. Убедитесь в их создании. С помощью редактора **extview** посмотрите индексный дескриптор и блок данных одного из них.
16. С помощью утилиты **fillfile** в рабочем каталоге пользователя создайте дополнительно три файла с длинными именами (например, для того, чтобы удобнее было отслеживать файловые записи в каталоге).
17. Найдите и просмотрите блок данных, выделенный рабочему каталогу пользователя. Найдите в нем и исследуйте записи обо всех созданных вами файлах.
18. Зафиксируйте в отчете номера **inode** и логических блоков, выделенных каждому созданному файлу. Эти данные вы будете использовать при анализе фрагментов удаленных файлов.
19. С помощью команды **rm –f** удалите все созданные вами файлы. Используя команду **ls –l,** просмотрите рабочий каталог пользователя и убедитесь, что логическое удаление файлов состоялось.
20. Создайте еще семь новых файлов с произвольными именами. Просматривая содержимое рабочего каталога пользователя с помощью команды **ls –li**, найдите вновь созданные файлы и установите, какие из индексных дескрипторов ранее удаленных файлов система передала новым объектам.
21. С помощью редактора **extview** просмотрите файловые записи в блоке данных рабочего каталога пользователя и установите, какие фрагменты удаленных файлов подлежат восстановлению. Обратите особое внимание на записи удаленных файлов с «длинными» именами. Результаты осмотра отразите в отчете.
22. С помощью команды **cat** прочитайте файл **/etc/fstab** и определите, как обозначается файл специального устройства, за которым закреплен логический раздел жесткого диска с файловой системой **ext2fs**.
23. Командой **debugfs** войдите в среду отладчика файловой системы и откройте логический раздел **ext2fs** для чтения (команда **open device**). Вместо **device** укажите файл специального устройства с логическим разделом Linux.
24. Вводя команду отладчика **lsdel**, посмотрите список **inode** удаленных

файлов. Найдите среди них номера файлов, которые были созданы и удалены вами (в файловой системе **ext3fs** отладчик **debugfs** удаленных файлов не обнаружит).

25. Выберите один из обнаруженных индексных дескрипторов удаленных вами файлов и с помощью команды **stat <inode>** установите, что записано в таблице и сохранились ли в ней ссылки на блоки данных. Обратите внимание на атрибуты, указывающие на удаление файла.
26. Если файл подлежит восстановлению, установите в «1» бит соответствующего **inode** в битовой карте индексных дескрипторов. Делается это с помощью внутренней команды отладчика **seti <inode>.**
27. С помощью команды отладчика **mi <inode>** сбросьте время удаления файла и установите в «1» число ссылок на файл.
28. Попытайтесь восстановить символьное имя удаленного файла. Это делается с помощью команды отладчика **ncheck <inode>.**
29. Командой **close** закройте логический раздел **ext2fs** на жестком диске и затем, вводя **quit,** выйдите из среды отладчика **debugfs**.
30. Выведите листинг рабочего каталога пользователя и проверьте, удалось ли вам восстановить удаленный файл. Сделайте выводы и отразите их в отчете.
31. Повторите операцию восстановления для другого удаленного файла.
32. По материалам Приложения изучите возможности утилиты **extview** по восстановлению удаленных файлов.
33. Предложите собственную методику восстановления данных удаленного файла, если его индексный дескриптор уже оказался занятым.
34. Воспользовавшись утилитой **shred,** удалите один из файлов, созданных в последнюю очередь. Используя вышеизложенную методику, попытайтесь найти составные части удаленного файла. Сделайте выводы и отразите их в отчете.

Контрольные вопросы

1. С какого времени файл можно считать логически удаленным? Каковы признаки логического удаления файла?
2. В какой последовательности «исчезают» компоненты файла при его логическом удалении в файловых системах **ext2fs** и **ext3fs**?
3. Какие методики восстановления логически удаленных файлов вам известны? В чем они заключаются?
4. Каким образом можно найти блоки данных, ранее принадлежавшие удаленному файлу (предполагаем, что файл был текстовым и его содержание приблизительно известно). Какие команды для этого следует использовать?
5. Каким образом можно «спасти» остатки логически удаленного файла, если его индексный дескриптор уже занят?
6. Как скопировать данные с поврежденного машинного носителя?

7. Какие требования предъявляются к гарантированному удалению конфиденциальной информации?
8. Какие программные средства гарантированного удаления данных имеются в современных операционных системах Linux?
9. Восстановите поврежденный или логически удаленный файл, предложенный преподавателем.

ЛАБОРАТОРНАЯ РАБОТА № 4
«Реализация политики разграничения доступа средствами ОС Linux»

1. Зарегистрируйтесь в системе с правами суперпользователя.
2. С помощью команды **`cat /etc/group`** откройте для просмотра файл групп. *Файл представляет собой таблицу, каждая строка которой является отдельной учетной записью, состоящей из 4 полей, разделенных двоеточиями. Первое и третье поле соответственно – имя и номер (GID) группы, второе – обычно отсутствующий групповой пароль, четвертое – имена пользователей, включенных в данную группу дополнительно. Они обладают групповыми правами на файлы владельцев, входящих в данную основную группу.*
3. С помощью команды **`cat /etc/passwd`** откройте для просмотра файл учетных записей. *Файл представляет собой таблицу, каждая строка которой является отдельной учетной записью, состоящей из 7 полей. Обратите внимание на характерные разделители полей (регистрационное имя, признак пароля, идентификатор пользователя, идентификатор группы, дополнительная информация, домашний каталог, имя командного процессора) в виде двоеточия. Символ **x** в поле признака пароля указывает на то, что хэшированный пароль находится в другом файле –* **`/etc/shadow`**.
4. Аналогично изучите содержимое «теневого» файла паролей **`/etc/shadow`**. *Он также представляет собой таблицу, каждая строка которой состоит из 9 полей, разделенных двоеточиями (регистрационное имя, хэшированный пароль, контрольные сроки в днях, среди которых:*

- *число дней с 01.01.70 до дня последнего изменения пароля,*
- *минимальное число дней действия пароля со дня его последнего изменения,*
- *максимальное число дней действия пароля,*
- *число дней до устаревания пароля, за которые система начнет выдавать предупреждения,*
- *число дней со времени обязательной смены пароля до блокировки учетной записи,*
- *день блокировки учетной записи).*

Последнее, девятое, поле зарезервировано и не используется.

5. Запустите утилиту **`pwck`** и проверьте с ее помощью файлы учетных записей **`/etc/passwd`** и **`/etc/shadow`** на отсутствие ошибок. Разберитесь в обнаруженных ошибках и, если они представляют угрозу, покажите их преподавателю. Самостоятельное редактирование парольных файлов, не предусмотренное настоящим заданием, запрещено.
6. С помощью команды **`groupadd omega`** создайте новую группу без указания ее числового идентификатора. Откройте для просмотра файл

групп и найдите в нем созданную запись. Обратите внимание на номер созданной группы и запишите его.

7. С помощью команды **useradd –m john** зарегистрируйте в системе нового пользователя. Аргумент **–m** указывает на необходимость создания домашнего каталога пользователя с его именем. После успешной регистрации, не назначая пользователю пароль, посмотрите содержимое текстовых файлов **/etc/passwd** и **/etc/shadow**. Обратите внимание на заполнение второго поля в созданной учетной записи, которая будет располагаться в конце каждого из файлов. Проверьте наличие и содержимое домашнего каталога пользователя, включая «скрытые» файлы. В какую группу по умолчанию включен зарегистрированный пользователь?
8. Командой **passwd john** присвойте новому пользователю какой-либо простой пароль (например, 12345). Программа напомнит вам о том, что пароль слишком простой, но администратор может с этим мнением не считаться. После завершения процедуры регистрации вновь откройте теневой файл паролей **/etc/shadow**. Что изменилось в учетной записи этого пользователя?
9. С помощью **Alt+F2** перейдите во вторую консоль и зарегистрируйтесь с именем и паролем вновь созданного пользователя. Чем отличаются символы в приглашениях ввода команд для администратора и обычного пользователя?
10. С правами пользователя **john** создайте в его домашнем каталоге текстовый файл с содержимым, позволяющим его идентифицировать (например, **echo 'my name' john > /home/john/j**). Затем установите права доступа 0750 на каталог **/home/john** и 0640 на созданный текстовый файл.
11. Аналогично предыдущим пунктам создайте учетную запись второго пользователя **useradd –m –g omega braun**. Присвойте ему такой же пароль. Затем посмотрите содержимое теневого файла **/etc/shadow**. Сравните учетные записи созданных пользователей и попытайтесь объяснить, почему при одинаковых паролях их хэшированные функции различаются?
12. С помощью **Alt+F1** перейдите в первую консоль и правами **root** создайте командой **touch /etc/nologin** пустой файл, запрещающий регистрацию в системе новых пользователей. Перейдите в третью консоль (**Alt+F3**) и попытайтесь зарегистрироваться там. Убедитесь в бесполезности таких попыток.
13. Из первой консоли удалите правами **root** созданный файл **/etc/nologin**, переключитесь в третью консоль и повторите попытку входа в систему пользователем **braun**. На этот раз попытка входа должна закончиться удачно.
14. С правами пользователя **braun** создайте в его домашнем каталоге тек-

стовый файл с содержимым, позволяющим его идентифицировать (например, **echo my name braun > /home/braun/b**). Затем установите права доступа 0750 на каталог **/home/braun** и 0640 на созданный текстовый файл.

15. С правами пользователя **braun** попытайтесь войти в домашний каталог пользователя **john** и прочитать его файл. Сделайте выводы.
16. С помощью **Alt+F2** перейдите во вторую консоль и с правами пользователя **john** попытайтесь войти в домашний каталог пользователя **braun** и прочитать его файл. Сделайте выводы.
17. Перейдите в первую консоль и с правами администратора командой **usermod –G omega john** включите названного пользователя в дополнительную группу. Откройте для просмотра файл **/etc/group** и зафиксируйте изменения.
18. Правами пользователей **john** и **braun** из соответствующих консолей повторите попытки доступа к чужим каталогам и файлам. Объясните произошедшие изменения.
19. С помощью команд **who** и **w** посмотрите список пользователей, которые сейчас работают в системе. Какую информацию вам удалось из этого извлечь? Чем отличаются результаты, выведенные командами **who** и **w**?
20. Не выходя из консоли администратора, запустите редактор **mcedit** с именем файла учетных записей **/etc/passwd** в качестве аргумента. В учетной записи пользователя **john** вместо символа **x** во втором поле поставьте символ восклицательного знака. Сохраните (F2) изменения в файле и выйдите из редактора.
21. Перейдите (**Alt+F2**) во вторую консоль и командой **exit** завершите сеанс работы пользователя **john**. Попытайтесь вновь зарегистрироваться в системе с этим именем. Почему вам это не удалось?
22. Вернитесь в консоль администратора (**Alt+F1**) и с помощью редактора **mcedit** разблокируйте учетную запись пользователя **john**. Вновь из второй консоли войдите в систему с его именем.
23. Вернитесь в консоль администратора (**Alt+F1**) и попытайтесь удалить учетную запись пользователя **john** командой **userdel –r john** (аргумент **–r** позволяет удалить с учетной записью и домашний каталог пользователя **/home/john**). Почему система не позволяет удалить эту учетную запись?
24. С помощью команды **ps –elf** выведите на экран список процессов, исполняющихся в системе. В последних строках файла найдите процесс командной оболочки, закрепленной за пользователем **john**. Прочитайте во втором столбце числовой идентификатор этого процесса (PID) и «убейте» его командой **kill -9 PID**. Перейдите во вторую консоль и убедитесь в том, что сеанс пользователя **john** завершен. Теперь учетную запись можно удалить (но сейчас это лучше не делать).

25. Попытайтесь присвоить пользователю **braun** нулевой идентификатор (это можно сделать командой **usermod –u 0 braun**). Почему администратору в такой попытке было отказано?
26. Вновь с правами администратора запустите **mcedit** и откройте в нем файл **/etc/passwd**. Найдите учетную запись пользователя **braun** и замените его числовой идентификатор **UID** на **0**. Сохраните изменения в файле (F2) и завершите редактирование.
27. Перейдите в третью консоль и попытайтесь прочитать теневой файл паролей (**cat /etc/shadow**). Почему пользователю в этом было отказано? Командой **exit** завершите сеанс пользователя **braun**, а затем вновь зарегистрируйтесь с этим именем. Почему изменился символ в строке приглашения? Вновь попытайтесь прочитать файл паролей. Почему на этот раз все получилось? Какую информацию о пользователях на этот раз выдает команда **who**?
28. Из третьей консоли запустите редактор и верните в исходное состояние учетную запись пользователя **braun**. Сделайте выводы о роли учетных записей пользователей.
29. Из консоли администратора просмотрите текстовые файлы в каталоге **/var/log** и найдите записи аудита, в которых зафиксированы входы в систему администратора и пользователей. Имеется ли доступ к файлам аудита у обычных пользователей?
30. Вам необходимо создать учетные записи и определить права доступа для десяти (10) сотрудников: **w_gromov, n_kalinina, e_ivanova, r_klinova, b_rebrov, k_beglov, i_frolov, d_lavrov, m_kruglov, t_uporov**, работающих в одном подразделении и занятых созданием и редактированием текстовых документов различного уровня конфиденциальности. Разграничение доступа к информации должно быть произведено на основании следующих требований:

- допуск к секретным сведениям имеют четыре пользователя: **w_gromov, n_kalinina, b_rebrov, k_beglov;**
- три пользователя: **n_kalinina, b_rebrov, k_beglov** работают над созданием секретных документов, каждый по своему профилю. Их домашние каталоги и файлы должны быть полностью недоступными как друг для друга, так и для всех остальных, исключая **w_gromov;**
- три пользователя: **i_frolov, d_lavrov, e_ivanova** имеют допуск к конфиденциальной информации и работают над документами с соответствующим грифом. Они имеют право читать файлы с конфиденциальной информацией, созданные своими коллегами, без права их модификации;
- все секретоносители имеют право знакомиться с конфиденциальными файлами;
- три пользователя: **r_klinova, m_kruglov, t_uporov** могут ра-

ботать только с открытой информацией. Их файлы должны быть доступны для чтения каждому сотруднику подразделения без права модификации;

- **w_gromov** является редактором подразделения и имеет право читать и копировать файлы всех сотрудников и всех уровней конфиденциальности. Завершенные документы копируются пользователем **w_gromov** в его домашний каталог, который должен быть недоступен для всех остальных сотрудников подразделения;
- всем сотрудникам, включая редактора, запрещается вносить изменения и удалять документы других пользователей, независимо от уровня конфиденциальности.

31. Укажите в отчете, какие коллизии вы усматриваете в сформулированных требованиях? Как реализовать указанные требования таким образом, чтобы пользователи не могли по своему усмотрению изменять установленный порядок?
32. С помощью команды **groupadd** создайте четыре пользовательских группы: **alfa, beta, nabla, sigma.** Формат команды **groupadd -g GID group_name.** Идентификатор группы **GID** можно назначать произвольно, начиная с номера 100 **(**например**, groupadd –g 101 alfa).**
33. Создайте учетные записи для вышеуказанных десяти новых пользователей. Регистрационные данные (кроме паролей и групп) сведены в табл. 2. Пароли назначайте произвольно, длиной не менее 8 символов, не забывая фиксировать их в черновике отчета. Для пользователей **e_ivanova, r_klinova** задайте одинаковые пароли. Распределите сотрудников по группам таким образом, чтобы удовлетворить вышеперечисленным требованиям. Изобразите в отчете наглядную схему, поясняющую разграничение доступа сотрудников подразделения к компьютерной информации. Заполните требуемыми правами доступа ячейки табл. 3.

Таблица 2

Пользователь	**UID**	**Пароль**	**Группа**	**Домашний каталог**	**Дата удаления учетной записи**
i_frolov	2001			/home/i_frolov	T+ 10 дней
m_kruglov	2002			/home/m_kruglov	T+ 30 дней
b_rebrov	2003			/home/b_rebrov	T+ 12 дней
d_lavrov	2004			/home/d_lavrov	T+ 60 дней
e_ivanova	2005			/home/e_ivanova	T+ 30 дней
t_uporov	2006			/home/t_uporov	T+ 15 дней
k_beglov	2007			/home/k_beglov	T+ 45 дней
n_kalinina	2008			/home/n_kalinina	T+ 30 дней
r_klinova	2009			/home/r_klinova	T+ 90 дней
w_gromov	2010			/home/w_gromov	не удалять

Таблица 3

Пользователь	Права доступа к объектам ФС					
	Файл редактора	Файлы «С»			Файлы «К»	Файлы «Н»
		С1	С2	С3		
i_frolov						
m_kruglov						
b_rebrov						
d_lavrov						
e_ivanova						
t_uporov						
k_beglov						
n_kalinina						
r_klinova						
w_gromov						

34. Пять первых пользователей (**w_gromov, n_kalinina, e_ivanova, r_klinova, b_rebrov)** зарегистрируйте с помощью команды **useradd.** Например, **useradd –u 501 –g sigma –d /home/n_kalinina –p v5g7K2S4 –e 2010-01-07 n_kalinina**. Параметр **-m** обеспечивает создание домашнего каталога пользователя, если он еще не существует. Прочие параметры команды можно не указывать. Идентификаторы пользователей **UID** назначать, начиная с 2001. Дату удаления учетной записи пользователя вводить в формате ГГГГ-ММ-ДД.
35. Пять последних пользователей зарегистрируйте с помощью командного файла **adduser**, которая запрашивает значения в интерактивном режиме. При вводе данных ориентируйтесь на подсказки системы [в квадратных скобках]. Все параметры, кроме имени пользователя, его идентификатора, имени группы, пароля и домашнего каталога, можно игнорировать. Для ввода параметра по умолчанию вводить **Enter**. В некоторых дистрибутивах ОС Linux командный файл **adduser** отсутствует, и в этом случае регистрацию остальных пользователей следует произвести аналогично предыдущему пункту. При использовании **adduser** символ подчеркивания в имени пользователя может не восприниматься. Если это так, замените этот символ в именах пользователей на символ дефиса.
36. Зарегистрировавшись администратором во вспомогательной консоли, отслеживайте из нее изменения, происходящие в файлах **/etc/passwd** и **/etc/shadow** по мере создания новых учетных записей.
37. Обратите внимание на то, что утилита **useradd** записывает пароли в файл **/etc/shadow** в открытом виде. Поскольку система воспринимает эту запись в качестве хэш-функции пароля, у пользователя с подобной учетной записью нет никаких шансов войти в систему. Правами администратора установите пользователям требуемые пароли с помощью команды **passwd.**

38. Регистрируясь в системе от имени пользователей **w_gromov**, **b_rebrov**, **d_lavrov** и **m_kruglov**, создайте их правами и в их домашних каталогах по одному текстовому файлу. Путем пробного доступа к этим файлам на чтение и запись от имени иных пользователей проверьте, удалось ли вам реализовать требуемую политику безопасности.
39. Правами администратора с помощью команды **usermod** по своему усмотрению измените основную группу пользователю **d_lavrov**. Проверьте, как изменились его права доступа к файлам иных пользователей.
40. Из первой консоли с помощью команды **su** измените права администратора на права пользователя **w_gromov**. Почему система не запрашивает пароль? С помощью команды **exit** верните себе права администратора. Был ли запрошен пароль? Почему?
41. Пользователь **d_lavrov** уволен за дисциплинарный проступок. С помощью команды **userdel d_lavrov** удалите его учетную запись. Кто теперь стал владельцем его домашнего каталога?
42. Зарегистрируйте вместо уволенного пользователя нового сотрудника **f_mironov** с предоставлением ему аналогичных прав (пароль должен быть новым!). Приобрел ли новый пользователь права на каталог ранее удаленного сотрудника? Для того чтобы подобное не происходило, при удалении учетных записей администратору необходимо вначале скопировать в недоступную для других директорию файлы пользователя, представляющие ценность для организации, а затем удалить учетную запись пользователя вместе с каталогом.
43. Пользователь **r_klinova** убыла в командировку сроком на две недели. Заблокируйте ее учетную запись любым из известных вам способов. Попытайтесь зарегистрироваться во второй консоли с правами **r_klinova** и убедитесь в том, что для этого пользователя система недоступна.
44. Зарегистрируйтесь во второй консоли с правами пользователя **k_beglov,** вызовите команду **passwd** и измените свой пароль. В качестве нового пароля введите **qwerty**.
45. Перейдите в консоль администратора и назначьте пользователю **k_beglov** новый пароль **zxcvbnm.** Затем с помощью команды **chage** (change aging – изменить информацию об устаревании) установите для этого пользователя минимальное время действия паролей, равное 5 дням. С какой целью устанавливается минимальный срок действия пароля?
46. Вспомните теоретический материал, связанный с файлом **/etc/sudoers.** Отредактируйте его таким образом, чтобы предоставить следующим пользователям дополнительные права за счет использования команды **sudo**:

- пользователю **e_ivanova** – право монтировать файловые системы типа **iso9660** на оптических дисках на своем компьютере,

- пользователю **b_rebrov** – право изменения владельца файлов на всех компьютерах локальной сети,
- пользователю **f_mironov** – право изменения учетных записей пользователей на своем компьютере без обязательного ввода пароля.

47. Из второй консоли с правами пользователя **f_mironov** создайте файл **cal 2011 > /home/f_mironov/cal2011.** С помощью команды **su** переключите консоль на пользователя **b_rebrov** и с помощью временно предоставленных ему привилегий передайте права на созданный **f_mironov** файл другому владельцу **n_kalinina.** Каким еще путем можно предоставить подобные права пользователям, не передавая им «опасных» полномочий администратора?
48. Просмотрите с правами администратора системные журналы в каталоге /**var/log** и убедитесь, что система зафиксировала факты присвоения полномочий администратора.

Временная нейтрализация парольной защиты

49. Отработайте вариант временной нейтрализации пароля администратора (например, для случая его утраты). Получите у преподавателя загрузочный оптический диск с ядром и минимальным набором утилит ОС Linux. Загрузите компьютер со сменного носителя. При необходимости измените порядок загрузки с использованием настроек в Setup BIOS. По завершении загрузки вы должны увидеть символ **#** и приглашение для ввода команды.
50. Введите команду **fdisk –lu** для отображения информации о разделах фиксированных и пристыкованных машинных носителях и установленных на них файловых системах. Найдите название файла блочного устройства, на котором установлен корневой раздел Linux (например, **/dev/sda6**).
51. Примонтируйте файловую систему Linux (предположительно это ФС **ext2fs** или **ext3fs**) на разделе HDD с помощью команды **mount –t ext2 /dev/hda6 /mnt**.
52. В случае успешного монтирования откройте файл учетных записей в смонтированной системе с помощью команды **vi /mnt/etc/passwd**. Текстовый редактор **vi** отобразит требуемый файл с мигающим курсором под первым символом учетной записи **root**. С помощью клавиш управления курсором переместите его под первый символ справа от двоеточия после слова **root**. Затем с помощью клавиши **x** требуется удалить все символы между первым и вторым двоеточием. В результате начало первой строки редактируемого файла должно выглядеть так: **root::0:0:**
53. Переключитесь в режим ввода команд редактора с помощью клавиши с двоеточием. После того как двоеточие и мигающий курсор после него появятся в последней строке, введите команду на сохранение изменений

и выход из редактора **wq <Enter>**. Если вы ошиблись и удалили лишние символы, проще перейти в режим команд и выйти из редактора без сохранения изменений с помощью **q! <Enter>** (затем следует повторить попытку редактирования).

54. При наличии в операционной системе встроенного редактора **Midnight Commander** процесс модификации учетной записи можно сделать более удобным. Файл открывается командой **mcedit /etc/passwd**, а результаты сохраняются функциональной клавишей F2.
55. Введите команду **reboot** для перезагрузки компьютера и извлеките компакт-диск. После загрузки Linux с жесткого диска на запрос об авторизации введите имя **root** и система зарегистрирует вас своим администратором без пароля. После завершения доступа требуется либо сменить пароль администратора с помощью команды **passwd,** либо восстановить удаленный признак пароля (по умолчанию это символ **x** между первым и вторым двоеточием в учетной записи **root** файла **/etc/passwd**). Редактирование файла паролей можно произвести встроенным редактором (F4) файлового менеджера **Midnight Commander**. Можно не использовать целиком файловый менеджер, ограничившись запуском его редактора **mcedit /etc/passwd**. Сохраните изменения (F2) и закройте программу.
56. Найдите в каталоге **/var/log** файл аудита, в котором зафиксирован вход в систему администратора без пароля.

Контрольные вопросы

1. Достаточно ли трех базовых прав доступа к файлам для реализации в ОС Linux требуемой политики безопасности?
2. Какие изъяны вы усматриваете в использованных утилитах регистрации учетных записей пользователей?
3. Может ли пользователь закрыть для себя доступ к собственному файлу? Каким образом? Почему система не соотносит владельца файла ни с его группой, ни со всеми остальными?
4. Существуют ли способы ограничения доступа суперпользователя к некоторым конфиденциальным файлам?
5. Какие дополнительные права администратор может предоставить пользователям с помощью утилиты **sudo** и регистрационного файла **/etc/sudoers**?

ЛАБОРАТОРНАЯ РАБОТА № 5
«Исследование процессов в ОС Linux»

Подготовка к работе

1. Зарегистрируйтесь в первом текстовом терминале с учетной записью **root.**
2. Комбинацией клавиш **Alt+F2** откройте второй текстовый терминал и зарегистрируйтесь как пользователь **user1** с паролем **1234567890**. В случае, если такой учетной записи нет, создайте ее по методике лабораторной работы 4.
3. Смонтируйте предоставленный преподавателем носитель к одному из подкаталогов каталога **/mnt**. Скопируйте с носителя один исполняемый файл с именем **signorer** в каталог **/bin.** Размонтируйте носитель и верните его преподавателю.

Наблюдение за файловой системой /proc

4. Все наблюдения за содержимым этой директории следует производить с правами администратора. Поскольку суперпользователь обладает правами на запись практически любого файла в таблице процессов, ему следует быть весьма осторожным в работе с ними.
5. Командой **cd /proc** перейдите в директорию **/proc.**
6. С помощью команды **ls -l|more** выведите на экран содержимое файловой системы **/proc**. Обратите внимание на то, что в строках листинга отображаются странные каталоги и файлы, имеющие нулевой размер. Большая часть каталогов вместо имен в последнем столбце содержит номера. Для каждого активного процесса в каталоге **/proc** существует подкаталог, имя которого совпадает с идентификатором этого процесса **PID**. При создании процесса каталог с его номером автоматически генерируется и удаляется при завершении процесса.
7. Командой **ls -li|more** выведите также информацию об индексных дескрипторах отображаемых файлов и каталогов. С помощью редактора **extview** убедитесь, что эти номера либо не существуют, либо принадлежат другим файлам. В дальнейшем для удобства просмотра информации можете использовать **Midnight Commander.**
8. Обратите внимание на время модификации псевдофайлов. Оно равно времени запуска соответствующей команды.
9. Откройте несколько нумерованных подкаталогов подряд. Обратите внимание на то, что все подкаталоги содержат одни и те же «файлы». С помощью команды **ps –ef** найдите **PID** командного интерпретатора, с которым администратор работает из первой консоли, и в дальнейшем наблюдайте за каталогом с этим номером.
10. Прочитайте информацию, относящуюся к одному из процессов (например, **cat /proc/103/status**). Чем может быть полезна такая

информация для администратора и пользователя? Представляет ли эта информация интерес для информационного нарушителя?

11. Открывая файлы для просмотра, вы можете получить необходимую справочную информацию о системе. Именно отсюда ее берут многочисленные утилиты:

- открывая командой **cat** файл **version**, прочитайте версию ядра ОС, которая сейчас запущена, а также сведения об его компиляции. Примерно такую же информацию предоставляет утилита **uname -r**,
- командой **ls -l kcore** можно вывести информацию об объеме ОЗУ в байтах. Не следует читать файл **kcore** – это эквивалентно попытке чтения всей оперативной памяти. Если в этом чтении есть потребность, следует узнать конкретный адрес размещения интересующих данных и читать их из специального файла **/dev/mem** с помощью утилиты **dd**,
- запустив команду **cat cpuinfo**, получите информацию о центральном процессоре (или процессорах),
- открывая для чтения файл **mounts**, можно получить динамическую информацию о смонтированных устройствах. Сравните ее с содержимым каталога **/etc/mtab**,
- файл **devices** содержит список старших номеров зафиксированных системой символьных и блочных устройств,
- файл **meminfo** хранит сведения об использовании системной памяти. Получите аналогичную информацию с помощью команды **free**.

12. Зайдите в каталог, номер которого соответствует **PID** командного интерпретатора, который в данный момент обслуживает вашу консоль. Прочтите значения установленных переменных окружения оболочки из псевдофайла **environ**. Командой

```
HOME=$HOME:/tmp
```

временно измените свой домашний каталог. Пронаблюдайте соответствующие изменения в файле **environ**. Проверьте изменение переменной окружения командой **echo $HOME**. Введите эту команду еще раз. Какие изменения произошли?

Если сброс переменной при ее чтении не происходит, посмотрите, как она изменится после завершения процесса. Для оболочки – это команда **exit** и повторная регистрация в этой же консоли (проверьте, как меняется **PID** оболочки).

Просмотр и анализ информации о процессах

13. Из консоли пользователя командой **ps -efl | more** выведите расширенный поэкранный список исполняемых процессов (перечень параметров для расширенного вывода информации можно уточнить с помощью электронного справочника **man ps)**. Разберитесь с выводимой информацией. Определите процессы:

- по типу: системные, демоны, пользовательские (тип процесса определяется по косвенным признакам, в частности по имени процесса в квадратных скобках, связи процессов с определенными владельцами и терминалами и др.);
- по состоянию **S:** (исполняющиеся – **R** или **O**, ожидающие записи на диск – **D**, ожидающие событий – **S**, приостановленные – **T**, зомби – **Z**),
- по текущему динамическому приоритету **PRI** (наименьшее значение у высокоприоритетных процессов),
- по относительному приоритету **NI.**

14. Комбинацией клавиш **Alt+F3** откройте третий текстовый терминал и зарегистрируйтесь в нем как суперпользователь. В этой консоли запустите утилиту **top** для текущего контроля процессов. Утилита позволяет отобразить наиболее активные процессы (сколько их помещается на экран) с достаточно полной информацией о них (для пользователя утилита представляет ограниченный набор выводимых параметров).
15. Из первой консоли создайте процесс **od /dev/zero > /dev/null.** В соответствии с введенной командой утилита **od** читает и выводит непрерывный поток байтов из «рога изобилия» в нулевое устройство. Переключившись в третью консоль, с помощью команды **top** просмотрите список наиболее активных процессов. Найдите и идентифицируйте запущенный процесс, найдите по идентификатору **PPID** его «родителя», определите его приоритет (возможно, это – величина переменная), долю загрузки центрального процессора **%CPU** и оперативной памяти **%MEM.**
16. Поочередно из первой и второй консолей с правами администратора и пользователя с помощью команды **od /dev/zero > /dev/null &** создайте по 2-3 одинаковых фоновых процесса.
17. По мере создания новых процессов отслеживайте в третьей консоли их текущий приоритет, загрузку процессора и памяти. Имеются ли различия в приоритете процессов, выполняемых от имени администратора и пользователя?
18. С консоли пользователя **user1** измените приоритет одного из принадлежащих ему процессов. Для этого воспользуйтесь командой **renice -10 PID**. Изменился ли относительный приоритет процесса?
19. Повторите предыдущий пункт, используя права **root.**
20. Переключитесь в консоль пользователя и измените приоритет одного из принадлежащих ему процессов командой **renice +5 PID.** Произошло ли изменение приоритета?
21. Проконтролируйте из третьей консоли изменение приоритетов запущенных процессов.
22. Из консоли пользователя восстановите приоритет ранее замедленного процесса командой **renice –5 PID.** Произошло ли восстановление прежнего приоритета? Почему?
23. С разрешения преподвателя завершите созданные вами процессы.

Управление процессами

24. С правами пользователя создайте в своей директории сценарий с именем **abcd.** Сценарий можно создать с помощью команды **cat:**

```
cat >abcd
#! /bin/bash
while : rem обратите внимание на пробел перед двоеточием!
# примечание вводить не нужно
do
echo HELLO!
done
Ctrl+d
```

25. Используя команду **chmod**, присвойте пользователю права на чтение и исполнение данного сценария. Запустите сценарий на исполнение (на экран должны непрерывно выводиться приветствия **HELLO!**).
26. Перейдите в третью консоль, с помощью команды **top** просмотрите список процессов и найдите в нем «зависший» процесс, запущенный пользователем (на самом деле это только имитация «зависания», которое пользователь легко может прекратить сам). Прочитайте идентификатор процесса **PID**.
27. Нажатием **Ctrl+C** из второй консоли остановите процесс. Как изменилось при этом состояние процесса?
28. Повторно запустите из второй консоли процесс, перейдите в первую консоль и отправьте «зависшему» процессу сигнал на останов (командой **kill** или внутренней командой **k** из работающей утилиты **top**).
29. Запустите ранее скопированную в каталог **/bin** утилиту **signorer**. Отправьте ей из этой же консоли несколько прерывающих сигналов в виде комбинаций клавиш (**Ctrl-C, Ctrl-\, Ctrl-Z**). Понаблюдайте за реакцией процесса. Как вы полагаете, по какой причине этот процесс не удается остановить?
30. Перейдите в другую консоль и отправьте «непослушному» процессу сигнал **kill -20 PID.** Как реагирует процесс на данный сигнал? Посмотрите в электронном справочнике, что означает данный сигнал.
31. С помощью команды **kill -9 PID** отправьте этому процессу сигнал принудительного завершения. С другой консоли проконтролируйте выполнение команды. Остановился ли процесс? Остался ли он в списке процессов? Какая программа на самом деле перехватывает и исполняет команду **kill -9 PID**?
32. С помощью команды **echo $PATH** поочередно из консоли администратора и пользователя **user1** выведите список директорий, в которых

производится поиск исполняемых файлов, заданных только по имени. В чем заключается различие выведенных списков? Почему в списке **PATH** администратора отсутствует текущий каталог (**.**)? Почему в списке **PATH** пользователя отсутствует каталог **/sbin**? Имеет ли пользователь возможность изменить порядок проверки каталогов для администратора?

33. Перейдите в первую консоль и повторите запуск утилит с правами суперпользователя.
34. Убедитесь в том, что пользователю разрешен запуск указанных утилит. Объясните, почему пользователь не может запустить утилиты с некоторыми «критичными» параметрами? Где, по вашему мнению, расположен механизм контроля за ходом исполнения таких команд (в ядре операционной системы, в командной оболочке, в самой утилите?). Ответ обоснуйте.
35. С правами пользователя скопируйте в свой рабочий каталог один из исполняемых файлов с параметром **SUID** каталога **/bin** (исполняемые файлы выделены цветом и символом *, а параметр **SUID** отмечен символом **s** в правах владельца на исполнение). Как изменились права доступа к файлу после его копирования?
36. Из второй консоли с правами пользователя скопируйте в свой домашний каталог утилиту, которую разрешено запускать только администратору (например, **chattr**). Копирование производите с параметрами, гарантирующими переход копии во владение пользователю. Убедитесь, что пользователь имеет на скопированный файл все необходимые права. Попробуйте использовать свою копию утилиты по ее назначению (в случае копирования утилиты **chattr** установите дополнительный атрибут **+i** одному из своих файлов). Сделайте выводы.

Работа с консолями

В тексте задания, хотя это не вполне верно, под консолью и терминалом будет пониматься один и тот же объект.

37. С помощью команды **useradd –m <user_name>** создайте в системе учетные записи трех новых пользователей: **alisa**, **berta** и **wanda**. Присвойте им одинаковые пароли **12345** и войдите под их именами в систему из виртуальных консолей **/dev/tty2 (Alt+F2)**, **/dev/tty3 (Alt+F3)** и **/dev/tty4 (Alt+F4)**.
38. Поочередно запустите из консолей каждого из пользователей несколько команд: **tty; mesg; ls –l $tty; id –G** . С их помощью определите:

- как правильно именуются файлы специальных устройств, связанные с консолями,
- доступны ли консоли для вывода информации для членов специальной «консольной» группы,

- какие права доступа определены на консоли для владельцев, членов их группы и иных пользователей,
- включены ли пользователи в какие–либо общие группы. В какие именно?

39. Используя в каждой из консолей команды **whoami**, **who** и **w**, определите, зависит ли вывод информации от имени ее инициатора (исключая первую команду). Насколько подробна и достаточна информация, выводимая каждой из утилит?
40. Перейдите в консоль пользователя **berta** и заблокируйте ее на запись командой **mesg n**. Командой **ls –l $tty** выведите права доступа пользователей к этой консоли. В соответствующем столбце должно отобразиться что–то похожее на **crw-------** .
41. Перейдите в консоль пользователя **alisa**. Определите рабочую консоль пользователя **berta** и командой **write** отправьте ей произвольное сообщение. Проверьте, дошло ли отправленное сообщение до адресата. Почему?
42. С правами одного из обычных пользователей отправьте иным пользователям «широковещательное» сообщение с помощью команды **wall**. Получено ли сообщение? Почему? Может ли пользователь создать помеху для администратора, забрасывая его потоком сообщений? Можно ли сделать это скрытно? Повторите отправку сообщений с помощью команды **wall**, используя права **root**.
43. Используя права одного из обычных пользователей, попытайтесь полностью открыть для других (на чтение и запись) его терминал. Затем попробуйте из консоли другого пользователя записать что–либо в доступную консоль, а также прочитать из нее вводимую информацию. Объясните полученные результаты.
44. Используя права одного из обычных пользователей, попытайтесь полностью заблокировать его консоль (например, в целях имитации отказа системы).
45. С правами суперпользователя заблокируйте программный интерпретатор пользователя командой

skill STOP tty3,

после чего переключитесь в консоль «заблокированного» пользователя, убедитесь, что он беспомощен в отношении вводимой и выводимой информации.

46. От имени суперпользователя отправьте «заблокированному» пользователю сообщение с помощью команды

echo Ваша консоль заблокирована > /dev/tty3

Попробуйте от имени пользователя откликнуться на полученное сообщение.

47. Из консоли суперпользователя разблокируйте пользовательскую консоль командой

skill CONT tty3

Проверьте результат выполнения этой команды.

Работа с каналами

48. Создайте неименованный канал (конвейер). Для этого можно использовать команду

od /dev/zero | tr '\0' '1'| more

Перейдя в другую консоль, с помощью команды **top** убедитесь, что были созданы 3 процесса: программы восьмеричного дампа **od**, программы перекодировки и транслитерации **tr**, заменяющей нули на единицы, и программы поэкранного вывода **more**. Все процессы запускаются и действуют одновременно. Различаются ли приорететы запущенных процессов?

49. Остановите запущенный процесс комбинацией клавиш **<Ctrl-C>.**

50. Изучите порядок создания и функционирования именованного канала. Для этого:

- правами **user1** создайте в каталоге хранения временных файлов именованный канал **mkfifo /tmp/fifo,**
- убедитесь в его создании и наличии прав на чтение из канала и запись в него **ls -l /tmp/fifo,**
- переключитесь во вторую консоль и введите команду чтения из канала со стандартного вывода (экрана) **cat < /tmp/fifo,**
- переключитесь в первую консоль и введите команду записи в канал со стандартного ввода (клавиатуры) **cat > /tmp/fifo,**
- наберите в первой консоли произвольный текст и нажмите **<Enter>** (буферизованный ввод),
- переключитесь во вторую консоль и прочитайте введенный текст. Повторите процедуру несколько раз,
- из первой консоли (где производится ввод в канал) комбинацией клавиш **<Ctrl+D>** введите команду на закрытие канала **FIFO**,
- удалите именованный канал командой **rm /tmp/fifo** .

Исследование опасных команд

51. Правами пользователя создайте в каталоге **/tmp** файл неограниченного размера, замаскированный под имя временного файла:

yes 12345 > /tmp/file2AF5DS & ,

где после префикса **file** в имени файла должна стоять шестибайтная случайная последовательность символов. Дождитесь сообщения систе-

мы о нехватке памяти и с помощью команды **ls -l** выведите информацию о размере созданного файла. С помощью утилиты **cat** посмотрите содержание созданного файла и убедитесь, что он заполнен строками из цифр 12345.

52. С помощью утилиты **df** оцените расход дискового пространства.
53. Перейдите в консоль другого пользователя и командой

```
echo privet! > /tmp/a
```

попробуйте создать небольшой текстовый файл. Чем завершилась попытка?

54. Попытайтесь создать такой же файл **/tmp/a** с правами администратора. Сделайте выводы об опасности подобных атак. Командой **rm** удалите созданный файл **/tmp/file2AF5DS.**
55. С помощью команды **ulimit –f 100** из консоли пользователя установите лимит на размер создаваемых файлов (в блоках).
56. Повторно правами пользователя попытайтесь создать в каталоге **/tmp** файл неограниченного размера:

```
yes 12345 > /tmp/file2 &
```

57. Дождитесь сообщения о создании файла и командой **ls –l /tmp** проверьте его объем. Ограничения на иные системные ресурсы устанавливаются аналогично.
58. Произведите атаку на истощение доступного ресурса индексных дескрипторов. Для этого от имени пользователя создайте в каталоге **/tmp** командный файл **abcd** со следующим содержанием:

```
cat >abcd
```

```
#! /bin/bash
while :
do
mkdir 1
cd 1
touch 2
done
```

```
Ctrl+d
```

59. Командой **chmod** присвойте владельцу указанного файла права на чтение и исполнение. Запустите файл из командной строки. Перейдите в каталог администратора (или другого пользователя) и ожидайте результат. Объясните его. Удалите созданный командный файл.
60. Произведите атаку путем запуска большого числа процессов. Для этого от имени пользователя создайте в каталоге **/tmp** командный файл **abcd** следующего содержания:

```
cat >abcd
#! /bin/bash
export RUN=$((RUN+1))
echo $RUN…
$0
Ctrl+d
```

Командой **chmod** присвойте владельцу указанного файла права на чтение и исполнение.

61. Перейдите в консоль администратора и введите команду **ps -ef** (но пока не запускайте ее!).
62. Вернитесь в консоль пользователя и запустите созданный командный файл. Вновь перейдите в консоль администратора. Запустите команду **ps -ef** и наблюдайте пользовательские процессы. Попробуйте остановить этот процесс правами администратора.
63. Система должна остановить этот процесс и вывести в консоль пользователя сообщение о большом числе созданных файлов.
64. Сделайте выводы о степени опасности перечисленных атак.

Контрольные вопросы

1. Чем отличаются системные процессы, «демоны» и пользовательские процессы?
2. Какую информацию можно извлечь из каталога **/proc**?
3. Как система распределяет процессорное время между конкурирующими процессами? Как на это влияет приоритет процессов?
4. Почему пользователю не разрешается повышать приоритет процессов? Усматриваете ли вы какую-либо опасность в возможности понижения приоритета пользовательских процессов?
5. В консоли, с которой вы работаете, произошло «зависание», вызванное неправильным исполнением одного из ваших процессов. Каким образом можно повлиять на возникшую ситуацию?
6. Какие угрозы безопасности связаны с использованием эффективного идентификатора **SUID**? Почему администратор должен учитывать и контролиролировать такие файлы?
7. В чем заключается различие между именованным и неименованным каналами? Для чего они используются?
8. Как можно заблокировать и разблокировать пользовательскую консоль?
9. Какие пользовательские атаки на исчерпание ресурсов системы вам известны?

ЛАБОРАТОРНАЯ РАБОТА № 6
«Исследование сетевых возможностей ОС Linux»

Лабораторная работа должна выполняться в проводной локальной вычислительной системе звездообразной топологии с концентратором (хабом). Убедитесь в наличии в исследуемой операционной системе Linux штатных утилит **ifconfig, netstat, nmap (netmap), nc (netcat)** и **tcpdump**. При отсутствии нужной утилиты самостоятельно либо с помощью преподавателя установите (скопируйте) ее.

1. Внимательно прочитайте задание и справку по синтаксису основных сетевых команд Linux.
2. Зарегистрируйтесь в системе в первой консоли с правами администратора.
3. Зарегистрируйтесь во второй консоли с правами пользователя.
4. Из консоли администратора с помощью команды **ifconfig -a** выведите на экран данные о текущем состоянии всех сетевых интерфейсов компьютера. Какую информацию из прочитанного вывода вы извлекли? Запомните, как обозначается основной Ethernet-адаптер (он может обозначаться **eth0, eth1, eth2**), и в дальнейшем используйте в сетевых командах это имя. Далее в тексте задания он упоминается как **eth0**.
5. Выведите информацию о сетевых интерфейсах с помощью команды **netstat –ai**, сравните возможности двух использованных утилит.
6. Активизируйте отключенную по умолчанию сетевую службу **telnet server**. Для этого запустите **Midnight Commander**, найдите конфигурационный файл **/etc/inetd.conf** и в режиме редактирования (F4) удалите символ комментария **#** перед строкой **telnet stream tcp nowait**, после чего сохраните изменения в файле. Если в системе используется демон **xinetd**, то активизация протокола производится в конфигурационном файле **/etc/xinet.d/telnet**, строка **disable = no**. Затем следует перезапустить систему.
7. Командой

```
ps –ef | more
```

выведите список процессов и убедитесь, что сетевой процесс **inetd** работает. Иначе его нужно запустить вручную командой **inetd**. Если исследуемая версия ОС не содержит сервера **telnet** (по причине его явной уязвимости некоторые версии Linux не предусматривают использования этого протокола), соответствующие пункты задания выполните с защищенной программной оболочкой Secure Shell (**SSH**).

8. Отключите сетевой адаптер командой **ifconfig eth0 down** (см. справку по сетевым командам). Присвойте сетевому интерфейсу временный MAC-адрес **A0:B1:C2:D3:E4:N**, где **N** – двузначный номер

компьютера в классе (при использовании в ЛВС одинаковых аппаратных или сетевых адресов возможны коллизии). Подключите адаптер к сети. Убедитесь в том, что его аппаратный адрес изменен.

9. Назначьте основному сетевому интерфейсу компьютера временный IP-адрес и маску подсети. Для этого введите команду

`ifconfig eth0 192.168.0.N netmask 255.255.255.0` ,

где **`N`** – номер компьютера. Повторным вводом команды **`ifconfig eth0`** убедитесь в том, что запись введенной информации произведена. Присвоенные сетевые адреса будут действовать до перезагрузки компьютера.

10. Проверьте работоспособность петли обратной связи, послав на свой же компьютер эхо-запрос **`ping 127.0.0.1`**. Убедившись, что отклики поступают, остановите зондирование комбинацией клавиш **`Ctrl-C`**.
11. Присвойте сетевому адаптеру дополнительный IP-адрес **`192.168.0.20+N,`** где **`N`** – номер компьютера. Проверьте прохождение ICMP-пакетов между сетевыми адресами на локальном компьютере.
12. Организуйте сеанс **`telnet`** на собственном компьютере, используя для этого интерфейс обратной петли или дополнительный IP-адрес. Для этого перейдите в консоль пользователя, наберите команду **`telnet 127.0.0.1`** (или **`telnet localhost`**) и после сообщения об успешном соединении введите **`login`** и пароль администратора. Почему вам было отказано в доступе? Почему соединение было закрыто? Можно ли считать эти меры надежной защитой, пресекающей передачу опасной информации по каналу связи в открытом виде?
13. Еще раз установите сеанс **`telnet`** через петлю обратной связи, используя на этот раз учетную запись обычного пользователя. После установления сеанса просмотрите список каталогов и файлов в нескольких директориях, список процессов и убедитесь, что в «удаленном» режиме доступа вы можете выполнять все команды, которые доступны пользователю, зарегистрированному на удаленном узле.
14. Перейдите в консоль администратора и с помощью команды **`w`** или **`who`** посмотрите, сколько сейчас пользователей в системе, кто они и с каких терминалов работают. Обратите внимание на то, как обозначаются локальный и удаленный терминалы.
15. С помощью команды **`netstat -a`** проконтролируйте список запущенных сервисов и их состояние. Найдите сеанс **`telnet`**.
16. Вернитесь в консоль пользователя и завершите локальный сеанс **`telnet`** командой **`exit`**. Получите сообщение о закрытии сетевого соединения. Проверьте эту информацию с помощью команды **`netstat`**.
17. Попробуйте войти на один из компьютеров сети в сеансе **`telnet`** (учетные записи пользователей на всех компьютерах должны быть одинаковы). Что потенциально опасного вы можете сделать на удаленном

компьютере? Приобретите на удаленном хосте права **root**. Проверив свои возможности по манипуляции удаленным компьютером, завершите сеанс командой **exit.**

18. Поскольку на вашем компьютере **telnet**-сервер активизирован, он тоже может стать объектом доступа. Периодически с помощью команд **netstat –a, ps –ef** или **w** проверяйте, не зафиксировали ли они подозрительные соединения, процессы или удаленных пользователей.
19. Войдите на произвольный узел по протоколу Secure Shell (команда **ssh** с указанием IP-адреса хоста, после запроса необходимо ввести пароль администратора). Проверьте свои возможности по манипуляции удаленным компьютером.
20. С помощью команды **arp –a** посмотрите таблицу соответствия сетевых и аппаратных адресов. Где расположен ARP-кэш и почему он сейчас пуст?
21. С помощью утилиты **ping** исследуйте локальную сеть, к которой подключен ваш компьютер в диапазоне адресов, идентифицирующих конкретный компьютер в сети, от 1 до 40 (**192.168.0.1/40**). Объясните, в чем уязвимость и неудобство такого метода сканирования.
22. Проверьте, обновилась ли после сканирования динамическая ARP-таблица. Если она содержит нужную вам информацию о сети, ее можно сохранить в файле командой **arp –a > /home/arp1** (через несколько минут информация о сетевых узлах будет изменена, и если тот или иной сетевой узел не проявляет активности, данные о нем в кэше будут утрачены). С помощью команды

arp –s <IP-адрес> <MAC-адрес>

создайте статическую **arp**-таблицу. Выясните местоположение этой таблицы.

23. Ознакомьтесь с синтаксисом команды **nmap** (netmap – карта сети). С помощью утилиты **nmap** исследуйте локальную сеть, к которой подключен ваш компьютер. Адреса в диапазоне можно вводить с использованием символов-джокеров и через дефис. Что вы можете сказать о полученной информации?
24. Исследуйте различные виды сканирования с помощью утилиты **nmap.** На этот раз в качестве объекта выберите один из компьютеров. Типы сканирования перечислены в справке по команде **nmap.** Какую дополнительную информацию о локальной сети вы получили? Каким образом эта утилита определяет тип операционной системы, управляющей сетевым узлом?
25. Отключите ответы своего сетевого адаптера на ARP-запросы других хостов командой

ifconfig eth0 –arp

С помощью команды **ifconfig eth0** убедитесь, что настройка выполнена.

26. Выждите несколько минут для сброса ARP-таблиц на компьютерах локальной сети и с одного из компьютеров сети с помощью утилиты **ping** постарайтесь обнаружить отклик своего компьютера. Достаточно ли надежно защищает компьютер от сканирования данная мера? Насколько нарушается при этом возможность работы в сети? Включите **arp**-отклик командой **ifconfig eth0 arp**.
27. Отключите сетевой интерфейс на своем компьютере с помощью команды **ifconfig eth0 down**. Повторите попытку обнаружения своего компьютера с одного из соседних узлов. Можете ли вы сами при этом проявлять какую-либо сетевую активность? Сделайте выводы. Вновь включите сетевой адаптер с помощью команды **ifconfig eth0 up**.
28. Переведите сетевой адаптер своего компьютера в режим перехвата всех пакетов с помощью команды **ifconfig eth0 promisc**. С помощью команды **ifconfig eth0** убедитесь, что настройка выполнена. При этом сетевой адаптер превращается в устройство подслушивания, но одновременно он становится очень уязвимым к сетевым атакам на отказ в обслуживании.
29. Ознакомьтесь с синтаксисом команды **tcpdump**.
30. С помощью утилиты **tcpdump** перехватите и прочитайте сетевые пакеты:

- отправленные из одного определенного адреса,
- являющиеся результатом сетевого обмена между двумя хостами,
- только Ethernet и IP-заголовки пакетов, направленных в адрес любого из компьютеров сети,
- сеансы **telnet** и **ssh** в локальной сети,
- ICMP-запросы в адрес вашего хоста,
- иные пакеты по указанию преподавателя.

31. Запишите несколько перехваченных пакетов в файл и просмотрите их в шестнадцатеричном коде. Найдите характерные поля и идентификаторы в заголовках канального, сетевого и транспортного уровней. Определите аппаратные и сетевые адреса, номера портов, иные характерные признаки, идентифицирующие сетевые протоколы.
32. Выясните, можно ли использовать **tcpdump** на сетевом интерфейсе, за которым не закреплен ни один IP-адрес.
33. Используя виртуальные сетевые адреса на локальном компьютере, организуйте сеанс сетевого копирования и канал наблюдения за ним. Для этого потребуется работа с трех консолей с правами администратора.
34. В первой консоли подготовьте (но пока не вводите!) команду для копирования (передачи) небольшого текстового файла, например файла паролей:

```
cat /etc/passwd | nc –w 2 192.168.0.22 3333
```

35. Во второй консоли подготовьте команду для приема копируемого файла:

```
nc –l –p 3333 > /home/password1
```

36. В третьей консоли подготовьте команду для перехвата копируемой информации:

```
tcpdump –i lo –xx –vv –s 100 > /home/password2
```

37. После проверки синтаксиса команд произведите их поочередный запуск: вначале **tcpdump** из третьей консоли, затем команду приема данных из второй консоли и, наконец, команду передачи данных. Дождитесь завершения команд в первой и второй консолях и затем комбинацией клавиш **Ctrl+C** остановите сеанс прослушивания **tcpdump**.
38. С помощью команды **cat** или **mcedit** просмотрите результаты копирования и перехвата. Обратите внимание на то, что **tcpdump** перехватил, по меньшей мере, семь пакетов, из которых три первых и три последних предназначались для установления и завершения TCP-сеанса. По этой причине большой объем копируемых данных должен представлять собой один непрерывный поток. Для этого рекомендуется использовать утилиту блочного копирования **dd** или утилиту **tar** (см. Справочник по командам Linux). Сравните между собой содержимое скопированного и перехваченного файлов.
39. На двух произвольно выбранных компьютерах в ЛВС с моноканалом произведите копирование большого массива данных. Предварительно рекомендуется произвести контроль установленных по умолчанию параметров фиксированного жесткого диска. В отношении HDD с IDE-интерфейсом для этого рекомендуется использовать команду **hdparm**. Для хронометража процедуры копирования рекомендуется выполнить совместно с утилитой **time**. Если результаты копирования некуда записывать, перенаправьте вывод в нулевое устройство.
40. На первом компьютере следует запустить команду

```
time dd if=/dev/hda count=10000|nc –w 2 192.168.0.22
3333
```

41. На втором компьютере следует запустить команду

```
time nc –l –p 3333 | dd of=/dev/null
```

42. На любом из компьютеров для контроля запустите программу **tcpdump**.

```
tcpdump –i lo –xx –vv –c 10 –s 100 > /home/hda
```

43. Оцените скорость копирования.
44. Выполните файловое копирование с помощью команд **tar** и **nc**. Команда **tar** используется для создания в качестве объекта копирования одного большого (здесь уместнее сказать – длинного) файла.

```
tar –czvf /home | nc –w 2 192.168.0.22 3333
```

Контрольные вопросы

1. Как закрепить за одним сетевым адаптером несколько IP–адресов?
2. Как программным путем изменить аппаратный адрес сетевой карты?
3. Можно ли перехватывать трафик без установленного IP–адреса?
4. Для чего нужно отключать ARP–отклик?
5. Где находится ARP–кэш? Как долго хранятся в нем данные?
6. Перечислите известные вам виды сетевого сканирования.
7. Запишите команду перехвата шести **`icmp`**–пакетов, исходящих из узла с IP–адресом 192.168.0.3 .
8. Как производится сетевое копирование данных?

ЛАБОРАТОРНАЯ РАБОТА № 7
«Исследование беспроводной сети WiFi под управлением ОС Linux»

1. Загрузите операционную систему Linux с компакт–диска (Live-CD). Загрузку произведите в текстовом режиме. Зарегистрируйтесь в двух первых консолях с правами **root**, пароль администратора отображается в подсказке.
2. С помощью команды **ifconfig –a|more** проверьте доступные сетевые интерфейсы. Обратите внимание на беспроводный адаптер, обозначенный как **ath0** (или **ath1**).

```
ath0      Link encap:Ethernet  HWaddr 00:1e:58:a1:fd:bd
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth0      Link encap:Ethernet  HWaddr 00:02:e3:32:db:1b
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:1782 (1.7 KiB)
          Interrupt:19

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

3. Аналогично посмотрите более точную информацию о состоянии беспроводного адаптера с помощью команды **iwconfig ath0**. В каком состоянии находится этот адаптер?

```
ath0      IEEE 802.11g  ESSID:""  Nickname:""
          Mode:Managed  Channel:0  Access Point: Not-Associated
          Bit Rate:0 kb/s   Tx-Power:15 dBm   Sensitivity=1/1
          Retry:off   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=0/70  Signal level=-94 dBm  Noise level=-94 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

4. Произведите конфигурацию виртуальных беспроводных интерфейсов. Виртуальные устройства следует создавать в заданной последовательности:

- с помощью команды **wlanconfig ath0 destroy** удалите виртуальную точку доступа,

- создайте новые виртуальные интерфейсы:

wlanconfig ath create wlandev wifi0 wlanmode adhoc

Слово **adhoc** означает, что адаптер будет работать в режиме одного из приемопередатчиков в одноранговой сети,

wlanconfig ath create wlandev wifi0 wlanmode monitor

Этот адаптер будет работать в режиме перехвата всех пакетов на заданном канале (**monitor**).

Созданные устройства автоматически получат порядковые номера **ath0** и **ath1**. Информацию об их состоянии можно получить с помощью команды **iwconfig**:

```
ath0      IEEE 802.11g  ESSID:""  Nickname:""
          Mode:Ad-Hoc  Channel:0  Cell: Not-Associated
          Bit Rate:0 kb/s   Tx-Power:15 dBm   Sensitivity=1/1
          Retry:off   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=0/70  Signal level=-94 dBm  Noise level=-94 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0

ath1      IEEE 802.11g  ESSID:""  Nickname:""
          Mode:Monitor  Channel:0  Access Point: Not-Associated
          Bit Rate:0 kb/s   Tx-Power:15 dBm   Sensitivity=1/1
          Retry:off   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=0/70  Signal level=-94 dBm  Noise level=-94 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

5. У созданных виртуальных устройств пока не установлены рабочие частоты и имя сети, а также иные параметры. Для установки некоторых параметров воспользуйтесь командой

iwconfig ath0 channel 1 essid "abcd"

После ввода команды состояние виртуального адаптера должно отображаться следующим образом:

```
ath0      IEEE 802.11g  ESSID:"abcd"  Nickname:""
          Mode:Ad-Hoc  Frequency:2.412 GHz  Cell: 02:1E:58:A1:FD:B9
          Bit Rate:0 kb/s   Tx-Power:15 dBm   Sensitivity=1/1
          Retry:off   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=0/70  Signal level=-93 dBm  Noise level=-93 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

6. Следующим шагом будет задание IP-адресов. Для этого используйте уже испытанную утилиту **ifconfig**:

ifconfig ath0 192.168.0.1

ifconfig ath1 up

Монитору нет необходимости иметь сетевой адрес, но активизировать его нужно. Результат, выведенный командой **ifconfig -a** после ввода параметров, отображается следующим образом:

```
ath0      Link encap:Ethernet  HWaddr 00:1e:58:a1:fd:bd
          inet addr:192.168.0.1  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

ath1      Link encap:UNSPEC  HWaddr 06-1E-58-A1-FD-BD-00-00-00-00-00-00-00-
00-00-00
          inet addr:192.168.0.1  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2011 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:295763 (288.8 KiB)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:632 (632.0 B)  TX bytes:632 (632.0 B)
```

7. Теперь попытайтесь проверить сетевую активность иных беспроводных устройств в локальной сети. Для этого рекомендуется воспользоваться командой

iwlist ath1 scan

Методом проб и ошибок нетрудно убедиться, что в режиме сканирования радиодиапазона может работать либо монитор, либо одноранговая точка. Заставить работать в данном режиме виртуальную точку доступа не удастся. Результаты выполнения команды **iwlist** могут быть такими:

```
ath1      Scan completed :
          Cell 01 - Address: 02:1E:58:A1:FD:B9
                    ESSID:"abcd"
                    Mode:Ad-Hoc
                    Frequency:2.412 GHz (Channel 1)
                    Quality=50/70  Signal level=-45 dBm  Noise level=-95
dBm
                    Encryption key:off
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 6 Mb/s; 9 Mb/s
                              11 Mb/s; 12 Mb/s; 18 Mb/s; 24 Mb/s; 36 Mb/s
                              48 Mb/s; 54 Mb/s
                    Extra:bcn_int=100
```

```
                    Ex-
tra:wme_ie=dd180050f2020101830002a3400027a4000042435e0062322f00
          Cell 02 - Address: 06:1E:58:A1:FD:B9
                    ESSID:"abcd"
                    Mode:Master
                    Frequency:2.412 GHz (Channel 1)
                    Quality=53/70  Signal level=-42 dBm  Noise level=-95
dBm
                    Encryption key:off
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 6 Mb/s
                              9 Mb/s; 12 Mb/s; 18 Mb/s; 24 Mb/s; 36 Mb/s
                              48 Mb/s; 54 Mb/s
                    Extra:bcn_int=100
                    Ex-
tra:wme_ie=dd180050f2020101830002a3400027a4000042435e0062322f00
                    Extra:ath_ie=dd0900037f01010024ff7f
```

8. Следующей тактической задачей является перехват заголовков пакетов, для чего можно использовать утилиту **tcpdump**. При отсутствии внешней сетевой активности ее можно создать с помощью команды **ping**, адресованной в один из внешних или внутренних IP-адресов.

```
tcpdump –i ath1 –xx –vv –c 5
```

9. Произведите разбор данных, выведенных в заголовке и шестнадцатеричном дампе команды **tcpdump**.
10. Отключите WEP-ключ шифрования для двух виртуальных точек **ad-hoc**. Для этого используйте команду

```
iwconfig ath0 key off
```

11. Подготовьте передачу незашифрованного трафика между двумя точками в одноранговой сети. Для этого наберите соответствующие команды:

- на клиентском (приемном) узле

```
nc –l –p 2222 > /dev/null ,
```

чтобы не записывать ненужную информацию на диск.

- на серверном (передающем) узле

```
tar –czvt /etc/* |nc –w 2 192.168.0.1 2222
```

Разберитесь с синтаксисом и смыслом введенных команд.

12. На виртуальном мониторе запустите команду **tcpdump** в режиме перехвата содержимого передаваемых пакетов. После этого запустить процесс передачи незашифрованных данных вначале на клиентском, а затем на серверном узле. По информации, выводимой анализатором пакетов, убедитесь в перехвате открытой информации.
13. Произведите настройку криптоалгоритма WEP на клиентском и серверном узлах. Для этого воспользуйтесь командой

```
iwconfig ath0 key 0123-4567-89AB-CDEF
```

14. Повторите передачу, прием и перехват зашифрованных данных.
15. С помощью утилиты **airdump-ng**, задавая номер канала, имя файла

для выводной информации и адрес узла, произведите взлом WEP–пароля. В зависимости от выбранного пароля требуется перехватить не менее 100 Мб данных.

16. Сделайте выводы относительно надежности используемого механизма шифрования.

Контрольные вопросы

1. В каких режимах может работать беспроводный сетевой адаптер?
2. В чем состоит функциональное назначение точки доступа?
3. В чем заключаются преимущества и недостатки беспроводной одноранговой сети?
4. Как можно получить параметры открытой беспроводной сети?
5. Какими командами устанавливаются параметры беспроводного сетевого интерфейса?
6. В чем заключается процесс «взлома» криптозащиты беспроводной локальной сети?
7. Каким образом можно перехватить и отобразить данные, передаваемые в беспроводной локальной сети?

ЛАБОРАТОРНАЯ РАБОТА № 8
«Наблюдение и аудит в ОС Linux»

1. Зарегистрируйтесь в системе в консольном режиме с правами **root**. Намеренно сделайте несколько ошибок при вводе пароля, чтобы «отметиться» в журналах аудита.
2. С помощью команды **md5sum** вычислите и запишите контрольную сумму для одного из файлов в каталоге **/home/user1/qu1**. С помощью команды **echo** добавьте один символ в этот файл:

```
echo a >>/home/user1/qu1/jan
```

Вновь вычислите контрольную сумму файла и сравните два результата.

3. С помощью команды **md5sum** вычислите и запишите в файл контрольную сумму всех файлов в каталоге **/bin**.
4. С помощью команды **find** найдите в корневом каталоге файлы:

- имеющие атрибуты **SUID (find / -perm +4000);**
- имеющие атрибуты **SGID (find / -perm +2000);**
- файлы, которые разрешено модифицировать всем:

```
find / -type f -perm +2 ;
```

- файлы, не имеющие владельца **(find / -nouser**);
- файлы и каталоги, имеющие UID незарегистрированных в системе пользователей.

Объясните, какой интерес могут представлять для администратора указанные категории файлов?

5. Путем просмотра процессов убедитесь в том, что системный сервис **syslogd** запущен. В целях контроля его действий найдите в журналах аудита записи о попытках своего входа в систему (это может быть файл **secure** или **auth** в каталоге **/var/log**).
6. С помощью команды **grep** найдите уязвимые учетные записи в файле паролей:

- имеющие числовые идентификаторы суперпользователя и его группы: **UID**=0 и **GID**=0;
- имеющие право на вход в систему без ввода пароля (отсутствующее второе поле в виде **::**).

Если у вас есть проблемы с вводом команды и установкой соответствующих фильтров, вы можете просто внимательно просмотреть содержимое файла паролей **/etc/passwd**.

7. С помощью команды **id user_name** посмотрите список основной и дополнительных групп пользователей. Найдите дополнительные группы **floppy**, **cdrom** и **plugdev**, дающие право использовать сменные машинные носители **/etc/cdrom**, **/etc/fd0** и т.д. для бесконтрольного блочного копирования данных.

Задание 1

Просматривая файл истории командной строки одного из пользователей, администратор обнаружил следующий фрагмент (см. ниже). Требуется исследовать приведенную последовательность команд, выявить намерения пользователя и установить, удалось ли ему нарушить установленную по умолчанию политику безопасности. Наиболее важные команды сопроводите своими комментариями.

```
su
su
su
su root
whereis su
pwd
cp /din/su
cp /bin/su
cp --help
cp --help | more
cp /bin/su /home/petrov
ls -l /bin
ls -l /bin | more
chmod 777 /bin/su
ls -l /bin
cp /bin/chmod /home/petrov
chmod 777 /bin/su
/home/petrov/chmod 777 /bin/su
ls -l /bin | more
ls -l
/home/petrov/chmod 4777 chmod
ls -l
/home/petrov/chmod 777 /bin/su
ls -l
chown root chmod
fdformat --help
fdformat /dev/fd0
mke2fs /dev/fd0
cat /etc/fstab
man mount
ls /mnt
mount -t ext2 /dev/fd0 /mnt/floppy
ls -l /mnt/floppy
cd /mnt/floppy
echo #! /bin/bash > ls
echo chmod 777 /bin/su >> ls
cat ls
```

```
chmod 777 ls
umount /dev/fd0
cd
umount /dev/fdo
mount -t ext2 /dev/fd0 /bin
ls -l /
mount -t ext2 /dev/fd0 /mnt/floppy
/mnt/floppy/ls
```

Задание 2

Проанализируйте содержимое файла **/etc/passwd** и сделайте выводы в отношении потенциальных угроз безопасности. Проставьте в необходимых местах свои комментарии.

```
abcd:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/bin/bash
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody::0:99:Nobody:/:/bin/bash
vcsa:x:69:69:virtual console memory own-
er:/dev:/sbin/nologin
rpm:x:37:37::/var/lib/rpm:/bin/bash
xfs:x:43:43:X Font Server:/etc/X11/fs:/sbin/nologin
rpc:x:32:32:Portmapper RPC user:/:/sbin/nologin
mailnull:x:47:47::/var/spool/mqueue:/sbin/nologin
smmsp:x:51:51::/var/spool/mqueue:/sbin/nologin
gdm:x:42:42::/var/gdm:/sbin/nologin
nscd:x:28:28:NSCD Daemon:/:/sbin/nologin
ntp:x:38:38::/etc/ntp:/sbin/nologin
pcap:x:77:77::/var/arpwatch:/sbin/nologin
root:x:501:501::/home/ivanov:/bin/vi
gromov:x:502:1::/home/gromov:/bin/bash
user1:x:503:504::/home/user1:/bin/bash
user2:!!:504:505::/home/user2:/bin/bash
```

Задание 3

Почти в каждой книге по безопасности операционных систем UNIX/Linux отмечается опасность запуска пользователями специально подготовленных файлов (исполняемых или командных) с установленным атрибутом **SUID**. Указывается на необходимость установки в файле **/etc/fstab** запрета на запуск файлов со сменных машинных носителей (noexec), а тем более – с установленным атрибутом SUID (nosuid). Предлагается проверить реальность такой угрозы. Дискету или носитель USB-Flash с «опасным» сценарием подготовьте с правами администратора, отредактируйте файл **/etc/fstab**, а затем проверьте наличие угрозы с консоли пользователя. На машинном носителе с помощью утилиты **mke2fs** должна быть установлена файловая система **ext2fs**, в противном случае не удастся скопировать на нее файл с нужными атрибутами. По результатам выполнения задания сделайте выводы.

Задание 4

Предлагается выяснить, существует ли надежный способ исключить возможность копирования пользователем конфиденциальной информации с жесткого магнитного диска на сменный машинный носитель. При этом учтите, что запрет на монтирование дисков не исключает возможности блочного копирования компьютерной информации.

Задание 5

У вас возникли подозрения, что в ваше отсутствие злоумышленники получили физический доступ к компьютеру на вашем рабочем месте. На единственном жестком диске компьютера установлена операционная система Linux с офисными приложениями и конфиденциальной информацией. В составе компьютера имеются устройства чтения и записи на ГМД и CD-R(W), а также интерфейсы USB. Системный блок компьютера был опечатан, и печати остались неповрежденными. Можно ли установить факт и характер несанкционированного доступа в операционной среде компьютера? Если да, то каким образом?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. UNIX: руководство системного администратора. Для профессионалов. /Э.Немет, Г.Снайдер, С.Сибасс, Т.Хейн. 3-е изд. СПб.: Питер; Киев: Издательская группа BHV, 2003. 925 с.
2. Баррет Д.Дж. Linux: основные команды. Карманный справочник: пер. с англ. /Д.Дж. Баррет. М.: КУДИС–ПРЕСС, 2008. 288 с.
3. Бэндл Д. Защита и безопасность в сетях Linux. Для профессионалов / Д. Бэндл. СПб.: Питер, 2002. 480 с.
4. Киркланд Д. Linux. Устранение неполадок / Д. Киркланд, К. Тинкер, Г. Тинкер; пер. с англ. А.А.Слинкина. М.: НТ Пресс, 2009. 490 с.
5. Костромин В.А. ОС Linux на вашем персональном компьютере / В.А. Костромин (электронный вариант книги).
6. Кэрриэ Б. Криминалистический анализ файловых систем / Б. Кэрриэ. СПб.: Питер, 2007. 480 с.
7. Мандиа К. Защита от вторжений. Расследование компьютерных преступлений / К. Мандиа, К. Просис. М.: Лори, 2005. 476 с.
8. Митчелл М. Программирование для Linux. Профессиональный подход: пер. с англ. / М. Митчелл, Дж. Оулдем, А. Самьюэл. М.: Издательский дом «Вильямс», 2003. 288 с.
9. Робачевский А.М. Операционная система UNIX / А.М. Робачевский. СПб.: БХВ-Санкт-Петербург, 2000. 528 с.
10. Роббинс А. Linux: программирование в примерах / А. Роббинс. М.: КУДИЦ-ОБРАЗ, 2005. 656 с.
11. Скляров И.С. Программирование боевого софта под Linux / И.С. Скляров. СПб.: БХВ-Санкт-Петербург, 2007. 416 с.
12. Таненбаум Э. Современные операционные системы. / Э. Таненбаум. 2-е изд. СПб.: Питер, 2002. 1040 с.
13. Фленов М.Е. Linux глазами хакера / М.Е. Фленов. СПб.: БХВ–Петербург, 2005. 544 с.
14. Фликенгер Р. Взломы и настройка LINUX. 100 профессиональных советов и инструментов: практ. пособ; пер. с англ. / Р. Фликенгер М.: ЭКОМ, 2006. 288 с.
15. The new ext4 filesystem: current status and future plans / A. Mathor, M. Cao, S.Bhattacharya, A. Dilger, A. Tomas, and L. Vivier; Proceedings of the 2007 Ottawa Linux Symposium, 2007. 16 с.
16. Ext4 block and inode allocator improvements / A. Kumar, M. Cao, J. Santos, and A. Dilger; Proceedings of the 2008 Ottawa Linux Symposium, 2008. 14 с.
17. Tim Johns. Анатомия ext4. Режим доступа: www.ibm.com

ПРИЛОЖЕНИЕ 1

КРАТКИЙ СПРАВОЧНИК ПО КОМАНДАМ LINUX

Просмотр содержимого файлов

cat [arg] <file_name> – просмотр содержимого текстового файла. Команда корректно работает лишь при выводе алфавитно-цифровых символов.

od <file_name> – вывод восьмеричного представления файла. Обычно используется для двоичных файлов.

xxd <file_name> – вывод комбинированного шестнадцатеричного и символьного представления файла. Обычно используется для двоичных файлов с текстовыми фрагментами.

mcedit <file_name> – открытие файла с произвольным форматом для чтения и редактирования в редакторе оболочки **Midnight Commander**.

«Навигация» по файловой системе

cd <dir> − изменение текущего каталога. Варианты переходов:

- **cd** (без аргументов) – переход в «домашний» каталог,
- **cd /** – в корневой каталог,
- **cd ..** – в родительский каталог,
- **cd /home/user1** – переход в домашний каталог пользователя **user1**.

pwd (print working directory) – вывод имени текущего каталога.

find <dir> [arg] <file_name> – поиск файла по имени или иным параметрам. Аргументы поиска:
-name «*шаблон*» – по именам файлов,
-inum <inode> – по номеру индексного дескриптора,
-mtime «*число*» – по числу дней до последнего изменения файла,
-type «*тип_файла*» – по типу файлов (обычные – f, каталоги – d, ссылки – l, сокеты – s и др.),
-perm «*режим*» – по указанному режиму доступа и т. д.

Общие операции над обычными файлами, каталогами и ссылками

mkdir -arg dir – создание нового каталога. Атрибут **–m** mode задает права доступа к каталогу. Пример: **mkdir –m 1555 /mnt/usb**.

rm -arg [file_name, dir] – удаление файлов и каталогов. Аргументы:
–f – безусловное удаление файла,

-d – удаление непустого каталога,
-r – рекурсивное удаление каталогов.
При обычном удалении файла система выводит запрос на удаление, который необходимо подтвердить символом «**y**» (yes) и **<Enter>**.

rmdir <dir> – удаление пустого каталога.

shred [arg] <file_name> – гарантированное удаление файла с многократным (25 раз) «затиранием» **inode** и блоков данных псевдослучайными комбинациями. Аргументы:

-v – показывать процесс удаления,
-u – без этого аргумента удаление не происходит,
-n раз – число повторов.

mv [arg] file1 file2 – изменение имени (переименование) файла.

mv [arg] <file_name> <dir> – перемещение указанного файла в другой каталог.

md5sum /sbin/* >> /home/md_sbin – вычисление контрольных сумм файлов каталога и запись их в файл в целях контроля целостности.

md5sum -c /home/md_sbin – повторное вычисление контрольных сумм и сравнение с прежними результатами.

ln <file_name> <link> – создание жесткой ссылки на файл.

ln -s <file_name> <link> – создание символической ссылки на файл.

chattr +(-)[arg] <file_name> – установка дополнительных атрибутов файла. Аргументы:

+i – блокирование любых изменений файла,
+a – запрет любых операций, кроме добавления данных,
+A – отключение режима автоматического обновления временной отметки последнего доступа.

Знак "**+**" означает присвоение атрибута, знак "**-**" – его удаление.

ls [arg] <dir> – вывод списка файлов в директории. Аргументы:

-l – подробная информация,
-a – все файлы и подкаталоги,
-i – вывод номеров **inode**.

ls -li <file_name> – получение подробной информации о конкретном файле и его индексном дескрипторе.

ls /dev/hd* – получение информации о логических и физических IDE-дисках.

stat <file_name> – получение информации о метаданных конкретного файла.

lsattr <file_name> [*dir*] – вывод информации о дополнительных атрибутах файла (-ов).

file <file_name> – получение информации о типе файла. Информация о разпознаваемых системой типах файлов хранится в **/usr/share/magic**.

fdisk –lu <device> – вывод информации о логических разделах на физическом диске.

fdisk –lu – вывод информации о логических разделах всех подключенных устройств дисковой памяти.

Непосредственная работа с машинными носителями

cat /dev/fd0 > /home/floppy – копирование всей дискеты в файл.

cat /home/file1 > /dev/fd0 – копирование файла **file1** на дискету, начиная с ее первого сектора.

cat /home/file2 >> /dev/fd0 – копирование файла **file2** на дискету, начиная с первого свободного сектора.

Изменение прав доступа

chmod mode <file_name> – изменение прав доступа к файлу.

Вариант 1: **chmod wXp <file_name>** ,

где

- вместо **w** подставляется: **u** (user) – пользователь, **g** (group) – группа пользователя, **o** (other) – остальные пользователи, **a** (all) – все,
- вместо **X** подставляется: **+** – предоставление права, **–** – лишение права, **=** – установление указанного права вместо имеющегося,
- вместо **p** подставляется символ, обозначающий соответствующее право: **r** (чтение), **w** (запись), **x** (выполнение), **s** (бит SUID), **t** (sticky bit).

Пример: **chmod o -wx /home/user1/file1** – лишение остальных пользователей прав на запись и исполнение указанного файла.

Вариант 2: **chmod XXXX <file_name>,** где **X** – восьмеричное число (порядок записи слева направо): дополнительные права, права пользователя, права группы, права остальных.

umask XXX (user mask) – изменение режима доступа по умолчанию для вновь создаваемых файлов. **Umask** без аргументов – вызов текущего значения маски.

chown [arg] <user> <file_name> – передача прав на файл другому владельцу.

Работа с процессами

ps [arg] – (process status) вывод информации о существующих процессах. Аргументы:

-a или **-e** – отобразить все процессы,
-f – полный листинг,
-l – расширенный листинг.

top – вывод с обновлением через определенное время.

top –b –n1 > outfile – сохранение однократной информации о наиболее активных процессах в файле с текстовым форматом.

kill sign PID – направление одного из сигналов процессу с указанным **PID** (**sign** = 15 – обычный сигнал о завершении, 9 – сигнал планировщику задач для принудительного завершения процесса).

kill -9 0 – остановка всех активных процессов пользователя.

skill –STOP ttyN – блокировка физического или виртуального терминала.

skill –CONT ttyN – разблокировка физического или виртуального терминала.

Работа с учетными записями пользователей

groupadd –g GID <group_name> – создание новой группы пользователей.

groupdel <group_name> – удаление зарегистрированной группы.

su (substitute user) – подстановка пользователя. Если объект не указан – попытка приобрести полномочия суперпользователя (администратора). Ввод команды сопровождается запросом пароля (за исключением случая, когда администратор хочет стать обычным пользователем).

su -<user_name> – смена пользователя со сменой пользовательского окружения.

su <user_name> – смена пользователя без смены окружения.

passwd – смена пароля текущего пользователя (будет предложено ввести новый пароль).

passwd <user_name> – назначение или смена пароля пользователю от имени администратора.

chage –l <user_name> – (change aging) получение или изменение

информации об устаревании пароля названного пользователя. Пользователи могут запрашивать только сведения об устаревании собственного пароля.

adduser – добавить учетную запись пользователя (ввод данных производится в интерактивном режиме).

useradd -u UID -g GID –G <group_name> -d <dir_home> -m -e <date_del_user> <user_name> – создание новой учетной записи пользователя с именем **<user_name>.** Аргументы:
-u – числовой идентификатор пользователя **UID**,
-g – числовой идентификатор группы **GID**,
-G – имена дополнительных групп (через запятую и без пробелов), в которые включается пользователь,
-d – путь к домашнему каталогу пользователя,
-m – указание создать домашний каталог пользователя по умолчанию,
-e – дата удаления учетной записи пользователя.

usermod –G alfa,beta petrov – добавление указанного пользователя к ранее созданным группам.

userdel -r <user_name> – удаление учетной записи пользователя (до удаления пользователь должен завершить сеанс работы). При наличии аргумента **-r** учетная запись удаляется вместе с каталогами и файлами пользователя.

w <user_name> – вывод данных о текущих процессах и терминалах каждого пользователя. При указании имени пользователя – сведения только о нем.

who <user_name> – вывод данных о пользователях, зарегистрированных в системе.

id –[arg] <user_name> – (identifier) вывод имен и числовых идентификаторов указанного пользователя и его групп. Если пользователь не указан – вывод информации о пользователе, зарегистрированном в данной консоли.

Клавиши быстрого вызова и общие команды

Alt+Fn – переключение на другой текстовый терминал.

Ctrl+Alt+Fn – переключение на текстовый терминал из графического режима.

Ctrl+D, или **logout**, или **exit** – завершение работы с терминалом.

Tab (после набора одного или нескольких первых символов команды) – вывод списка команд, начинающихся данными символами.

history – перечень ранее введенных команд (по умолчанию список от 500 до 1000 команд).

↑ – возврат к предыдущей команде.

man <name_command> – вызов электронной справки по нужной команде. Выход из электронного справочника по **q**.

info <name_command> – еще один вариант вызова электронной справки по нужной команде. Выход из электронного справочника по **q**.

<command_name> --help – вызов краткой справки о команде.

shutdown -h t – полная остановка системы через **t** минут.

shutdown -h 0, или **poweroff**, или **init 0**, или **halt** – немедленная полная остановка (завершение работы системы все равно потребует несколько минут).

shutdown -r , или **reboot**, или **init 6**, или **Ctrl+Alt+Del** – перезагрузка операционной системы.

Отображение информации о дисковом пространстве и файловых системах

fdisk -lu <dev>

<dev> – имя специального файла устройства, которое идентифицирует физический диск, например **/dev/hda**.

fdisk -lu – вывод информации обо всех устройствах памяти, подключенных к IDE, SCSI, SATA и USB интерфейсам данного компьютера.

fdisk –lu /dev/hda – вывод информации о разделах жесткого магнитного диска с IDE-интерфейсом, подключенного в качестве ведущего к первому интерфейсу.

extview –i <inode_number> <dev> – вывод информации, содержащейся в указанном **inode**.

extview –b <block_number> <dev> | more – вывод поэкранного дампа логического блока с указанным номером. Обычно логические блоки нумеруются шестнадцатеричными числами, в этом случае номер блока указывать в виде **0x12345678**.

Монтирование и размонтирование файловых систем

mount -t type -o option <dev> <dir>

type – тип монтируемой системы (**ext2, ext3, msdos, vfat, ntfs, iso9660** и т. д.). Тип **auto** – предоставление системе возможно-

сти автоматически определить монтируемую файловую систему.

option – дополнительные опции монтирования:

ro или **rw** – файловая система монтируется только для чтения либо для чтения и записи,
iocharset=koi8-r – кодировка для отображения символов кириллицы (просмотр кодировки по команде **locale**),
exec/noexec – разрешить или запретить запуск исполняемых файлов, расположенных в примонтированной файловой системе.
suid/nosuid – принимать во внимание или игнорировать дополнительные атрибуты **SUID/SGID**, позволяющие запуск исполняемых файлов из данной файловой системы с правами владельца файла или его группы.

<dev> – имя специального файла устройства, которое идентифицирует логический диск с монтируемой файловой системой, например **/dev/hda2**.

<dir> – точка монтирования (каталог), который к моменту монтирования уже должен существовать.

umount <dev> или **umount <dir>** – размонтирование файловой системы. Вместо **<dev>** или **<dir>** следует указать конкретные значения (см. команду **mount**). К моменту монтирования все каталоги и файлы смонтированной файловой системы должны быть закрыты.

umount /mnt/cdrom – размонтирование оптического диска.

Копирование данных

cp [arg] file1 file2 – создание копии файла с другим именем.

cp [arg] file1 <dir> – копирование файла с прежним именем в другой каталог.
cp [arg] <dir1> <dir2> – копирование файлов из **<dir1>** в **<dir2>**.

Аргументы:

-a – сохранение атрибутов файла,
-p – сохранение режима доступа к файлу, его принадлежности и временных отметок (без этого параметра файл переходит в собственность копирующего).

tar –cf backup.tar /home /etc – копирование (группировка) в «ленточный» резервный файл содержимого одного или нескольких каталогов (каталоги отделяются в командной строке пробелами).

tar –xf backup.tar <dir> – распаковка данных из «ленточного» файла.

dd if=<источник> of=<приемник> bs=<размер_блока> seek= <чис-
ло_блоков> skip=<число_блоков> count=<число_блоков> conv=noerror,fsync

if=<источник> – файл, откуда копируются данные. Если источник не указан, копируются данные из стандартного ввода **stdin**, в случае интерактивной работы – введенные с клавиатуры. Поток данных может передаваться команде **dd** из другой программы через конвейер; в этом случае параметр **if=** не указывается.

of=<приемник> – файл, в который записываются данные. В случае отсутствия адресуемого файла в файловой системе он будет создан. Если параметр **of=** не указан, данные выводятся в стандартный вывод **stdout** или на экран. В случае отсутствия параметра **of=** вывод утилиты через конвейер можно перенаправить другой программе.

bs=<размер_блока> – размер блока копируемых данных, который по умолчанию равен 512 байтов. Размер блока может задаваться раздельно для источника (**ibs** – input block size) и для приемника (**obs** – output block size). Если копируемые блоки одинаковы, то задается величина **bs**. Размер может задаваться в байтах (единица измерения не указывается), килобайтах (K), мегабайтах (M), гигабайтах (G).

skip=<число_блоков> – количество (десятичное число) пропущенных при копировании из источника блоков указанного размера.

seek=<число_блоков> – количество пропущенных приемником блоков.

count=<число_блоков> – количество копируемых блоков указанного размера.

conv=noerror,fsync – режим обработки ошибок, при котором блок, скопированный с ошибкой контрольной суммы в приемнике, заполняется нулями, а процесс копирования не прерывается. При отсутствии этого параметра копирование завершается после первой ошибки чтения. Аргумент **fsync** служит для того, чтобы скопированные данные не «застревали» в дисковом кэше, а сразу записывались на диск.

Варианты использования утилиты dd

dd if=/dev/fd0 of=/home/floppy conv=noerror,sync – создание файл-образа гибкого магнитного диска.

echo 1234567890 | dd of=/dev/fd0 bs=1 seek=10 count=10 – запись десяти цифр на гибкий магнитный диск со смещением 10 байтов относительно его начала.

dd if=/dev/hda bs=512 count=1|xxd – вывод на экран для просмотра содержимого первого сектора главной загрузочной записи жесткого магнитного диска. Утилита **xxd** позволяет просмотреть данные в шестнадцатеричном виде и ASCII кодировке.

dd if=/dev/hda6 bs=4096 skip=1 count=1|dd bs=32 skip=5 count=1|xxd – вывод на экран для просмотра дампа описателя 6-й группы блоков файловой системы **ext2fs.**

dd if=/dev/hda6 bs=4096 skip=4 count=1|dd bs=128 skip=10 count=1|xxd – вывод на экран для просмотра дампа 11-го индексного дескриптора файловой системы **ext2fs.**

dd if=/dev/hda7 of=/home/hd7 bs=4k conv=noerror,fsync – создание в домашнем каталоге целевого компьютера файл-образа 7-го раздела жесткого магнитного диска с IDE-интерфейсом.

dd if=/dev/sda1 bs=4k|nc –w 3 192.168.1.20 2222 – сетевое копирование на целевой компьютер 1-го раздела первого SATA-диска. Указан IP-адрес и номер порта транспортного уровня целевого компьютера.

nc –l –p 2222 > /home/sd1 – командная строка, обеспечивающая сетевое копирование на целевом компьютере.

cdrecord –v -eject -sao dev=1000,0,0 speed=6 image.iso

Команда для записи файла на компакт–диск (CD). Аргументы:
speed – скорость записи для привода,
dev=1000,0,0 – номер устройства для чтения и записи CD-RW,
eject – открывание лотка привода CD после окончания записи,
image.iso – имя файл–образа в формате **iso9660** либо другого файла.
Номер устройства определяется с помощью команды

cdrecord --scanbus

Очистка CD-RW перед записью

cdrecord –v dev=1000,0,0 blank=fast

Подготовка файл-образа будущего диска

mkisofs –R –l -o /tmp/disk.iso <dir>

Синтаксис некоторых сетевых команд

1. Команда **ping** служит для зондирования эхо-запросами сетевых узлов для установления их наличия и доступности.

ping <параметры> <адрес_хоста> <номер_порта>

Параметры:

-l count или **-c count** – отправка указанного числа пакетов. По умолчанию (в зависимости от ОС) посылается либо один пакет, либо бесконечная серия пакетов с интервалом в одну секунду. Прерывается нажатием **Ctrl – C**,
-s count_byte – общее количество байтов в **icmp**-пакете с эхо-запросом (длина заголовка **icmp**-пакета – 8 байт),
-i timeout – временной интервал в следовании пакетов в секундах,
-f – направление пакетов с максимально возможной скоростью (только с правами **root**),
<адрес_хоста> – доменное имя или IP – адрес целевого компьютера,
<номер_порта> – номер, закрепленный за сетевой службой, запущенной на удаленном компьютере (смотри файл **/etc/services**).

Например: **ping –c 3 –i 5 192.168.1.2 21** – направление трех стандартных пакетов с пятисекундным интервалом в адрес FTP-сервера на узле с IP-адресом 192.168.1.2.

Утилита выводит данные построчно в следующем порядке: число байтов в принятом пакете, IP-адрес исследуемого узла, порядковый номер пакета, счетчик «жизни» пакета и время возврата.

2. Команда **arp** служит для просмотра или изменения содержания ARP-таблицы. Аргументы:

-a <hostname> – отображение ARP-записей для IP-адреса указанного узла или всех известных узлов, если **hostname** не задается,
-d – удаление всех ARP-записей,
-s <hostname> hwaddr – принудительное задание ARP-записи в таблицу.

3. Протокол **telnet** (сетевой протокол телекоммуникации) служит для удаленного доступа на уровне команд между узлами сети.

telnet <адрес_хоста> <адрес_порта>

<адрес_хоста> – доменное имя или IP – адрес целевого компьютера.
По умолчанию происходит TCP-соединение с портом 23 на удаленной машине. Чтобы соединиться с другим портом, надо указать его номер в конце командной строки:

telnet 192.168.0.10 31337

После ввода команды и успешного соединения выводится приглашение для ввода идентификатора и пароля. Система обычно не допускает удаленной регистрации с правами суперпользователя. Поэтому на удаленном компьютере нужно иметь учетную запись обычного пользователя и регистрироваться с его правами. После регистрации и получения соответствующего уведомления можно управлять удаленным компьютером в режиме командной строки, как своим собственным. В ходе сеанса с помощью команды **su**

можно повысить свои права (при знании пароля). Сеанс завершается по команде **exit**.

4. Команда **ifconfig** служит для отображения или установки параметров сетевого интерфейса (адаптера или модема).

ifconfig <интерфейс> <адрес> <параметры>

<интерфейс>
eth0 … **ethN** – установленные сетевые адаптеры Ethernet,
lo (Local Loopback) – «кольцевой интерфейс» или «обратная петля», для нее обычно устанавливается IP-адрес 127.0.0.1,
ppp0 ... pppN - модемы,
vmnet0 ... vmnetN – виртуальные сетевые интерфейсы,
tr0 ... trN – сетевые интерфейсы Token Ring.

<адрес> – IP или MAC адрес, присваиваемый узлу (IP-адрес присваивается на сеанс работы).

<параметры>
-a – вывод информации обо всех сетевых интерфейсах компьютера, включая виртуальные,
(-) promisc – выключение или включение режима «беспорядочного» перехвата всех пакетов в физическом канале,
(-) arp – выключение или включение ответов на **arp**-запросы с других узлов сети,
mtu N – установка параметра Maximum Transfer Unit (MTU),
down – отключение IP-трафика через интерфейс,
up – включение интерфейса (аргумент обязателен, когда производится перезапуск интерфейса, который был временно выключен опцией **down**).

***Варианты использования утилиты* ifconfig**

ifconfig -a – вывод информации обо всех сетевых интерфейсах данного компьютера,

ifconfig eth0 – вывод информации о первом (или единственном) Ethernet-адаптере (его номер не обязательно равен нулю),

ifconfig eth0 192.168.0.3 netmask 255.255.255.0 – динамическое (до перезагрузки компьютера) присвоение интерфейсу сетевого адреса и маски сети,

ifconfig eth0:1 192.168.0.3 – динамическое присвоение интерфейсу дополнительного IP-адреса,

ifconfig eth0 –arp – отключение ответов на ARP-запросы других узлов сети,

ifconfig eth0 promisc – перевод Ethernet-адаптера в режим перехвата всех пакетов,

ifconfig eth0 down – отключение Ethernet-адаптера,

ifconfig eth0 hw ether 01:02:03:04:05:06 – динамическая установка нового MAC-адреса (производится после отключения интерфейса с аргументом **down**),

ifconfig eth0 up – активизация Ethernet-адаптера после смены MAC-адреса.

5. Команда **netstat** предназначена для получения разнообразной информации о состоянии сети. Аргументы:

-a – вывод полной информации,
-i – вывод информации о сетевых интерфейсах. В последней колонке таблицы кратко отображается информация о состоянии интерфейса (B – установлен широковещательный адрес, L – интерфейс задает устройство loopback, M – интерфейс работает в режиме захвата всех пакетов, O – ARP выключен, R – интерфейс работает),
-t – вывод информации о **tcp**-сеансах,
-u – вывод информации о **udp**-сеансах,
-n – отображение только IP-адресов вместо имен хостов.

6. Команда **nmap** служит для разведки сети.

nmap <тип_сканирования> <параметры> <список узлов/сетей>

<тип_сканирования>
-sT – TCP-сканирование с установлением соединения. Используется по умолчанию и может запускаться обычным пользователем,
-sS – TCP-сканирование с помощью сообщений SYN (считается наилучшим из методов TCP-сканирования),
-sU – UDP-сканирование,
-sP – ping-сканирование,
-sF – FIN-сканирование,
-sN – нуль-сканирование.

<параметры>
-O – режим изучения откликов для определения типа удаленной операционной системы,
-p <диапазон> – диапазон портов, которые будут сканироваться (указываются через запятую или дефис),
-v (**vv**) – режим вывода подробной информации,
-T <число> – темп сканирования от «0» – очень медленно (один пакет в пять секунд) до «5» – максимально быстро (один пакет за 0.3 секунды).

7. Команда **tcpdump** – универсальная утилита для прослушивания моноканала. Автоматически переводит сетевой адаптер в режим захвата всех пакетов, но отображает или сохраняет в файл только отфильтрованные пакеты.

tcpdump <параметры> <параметры_фильтрации>

<параметры> (приводятся наиболее важные параметры):
-v, -vv, -vvv – вывод информации с различной степенью подробности,
-c <число_пакетов> – завершение работы после захвата указанного числа пакетов,
-s <длина_пакета> – размер перехватываемого пакета в байтах. Обычные размеры заголовков пакетов: Ethernet – 14 байтов, IP – 20 байтов, TCP – 20 байтов. По умолчанию перехватываются первые 68 байтов из пакета,
-w <имя_файла> – запись перехваченной информации в файл специального формата,
-r <имя_файла> – чтение и вывод на экран содержимого ранее записанного файла,
-i <интерфейс> – тип сетевого адаптера или модема, с помощью которого производится перехват пакетов,
-n – вывод номеров хостов вместо их символьных имен,
-e – вывод MAC-адресов передатчика и приемника,
-x (-xx) – отображение содержимого IP-пакета в шестнадцатеричном виде,
-X – отображение содержимого пакета в шестнадцатеричных и ASCII– кодах.

<параметры фильтрации> используют несколько ключевых слов:

- **Протокол (proto)** – указывает, какие именно пакеты подлежат перехвату. Используются ключевые слова: **ether, ip, arp, rarp, tcp, udp**;
- **Направление** – указывает источники и получателей сообщений: **src** (source – источник), **dst** (destination – получатель) или их комбинацию: **src or dst** или **src and dst**;
- **Объекты прослушивания**, к которым могут относиться:

host (номер или имя) – сетевой узел, являющийся источником или получателем сообщений,
net (сетевая часть адреса) – локальная сеть или ее часть,
port – номер или символическое обозначение службы, указанной в таблице **/etc/services**.

В параметры фильтрации могут входить математические выражения.

Ключевые слова и математические выражения могут объединяться с использованием логических условий: **not**, **and** или **or**.

Варианты использования утилиты **tcpdump**

tcpdump –i eth0 –c 25 –vv -w /home/net_dump1 ip host 192.168.0.34 – перехват Ethernet-адаптером 25 сетевых пакетов (**ip**) стандартного размера, адресованных узлу **192.168.0.34** или исходящих от него, с детальным описанием и записью информации в указанный файл,

tcpdump –i eth0 –c 100 –s 1514 ether src 00:01:02:03:04:05 – перехват 100 пакетов канального уровня длиной 1514 байтов, передаваемых сетевым адаптером Ethernet (**ether**) с указанным MAC-адресом (ключевое слово **host** после **src** или **dst** можно не указывать),

tcpdump –i eth0 –c 50 –v –w /home/tcp_dump2 src 192.168.0.1 and dst 192.168.0.12 – перехват 50 пакетов любого типа, направленных от узла **192.168.0.1** в адрес узла **192.168.0.12**, с записью информации в файл,

tcpdump –r /home/tcp_dump2|more – чтение и вывод ранее записанного файла с сетевым дампом,

tcpdump –i eth0 (tcp or udp) port 53 – перехват сетевым адаптером Ethernet всех **tcp** и **udp** пакетов, исходящих или направляемых в 53-й порт,

tcpdump –i eth0 (ip or arp or rarp) src net 128.3 – перехват всех пакетов формата **ip, arp** и **rarp**, исходящих из сети с указанной маской,

tcpdump –i eth0 not src net 192.168 and (ip[19] = 0xff or ip[19] = 0x00) – перехват всего широковещательного трафика от любого внешнего источника, кроме собственной подсети,

tcpdump –i eth0 udp[2:2] >= 33000 and udp[2:2] < 34000 – фильтр для отслеживания попыток установить соединение с **udp**-портами в диапазоне 33000 – 33999.

Контроль и конфигурация беспроводной ЛВС

Команда iwconfig

iwconfig <dev> mode master – перевести адаптер в режим работы точки доступа,

iwconfig <dev> mode managed – обычный режим при работе с точкой доступа,

iwconfig <dev> mode ad-hoc – установка одноранговой сети точка-точка без выделенной точки доступа,
iwconfig <dev> mode monitor – перевод адаптера в режим перехвата всех пакетов на данном канале (поддерживается не всеми картами),
iwconfig <dev> essid any – отключить проверку ESSID и подключаться к любой доступной сети,
iwconfig <dev> essid "abcd" – задать ESSID сети,
iwconfig <dev> key 1234567890123456789012345678 – установить 128 bit WEP или 26-символьный hex-ключ,
iwconfig <dev> key 1234567890 – установить 64 bit WEP или 10-символьный hex-ключ,
iwconfig <dev> key off – отключить WEP ключ,
iwconfig <dev> key open – установить открытый режим, не требующий авторизации при подключении и шифрования трафика,
iwconfig <dev> channel N – установить рабочий канал в диапазоне от 1 до 11,
iwconfig <dev> channel auto – автоматический выбор канала,
iwconfig <dev> freq 2.422G – канал также может быть задан в виде указания определенной частоты,
iwconfig <dev> ap 11:11:11:11:11:11 – установка MAC-адреса точки доступа,
iwconfig <dev> rate 11M – указание скорости обмена в [Мбит/сек],
iwconfig <dev> rate auto – автоматический выбор скорости обмена.

Команда iwlist

iwlist показывает некоторые параметры беспроводной карты, которые недоступны **iwconfig**:
iwlist <dev> scan – получить список доступных точек доступа и компьютеров, настроенных в режим точка–точка, а также такие параметры как имя сети, качество сигнала, частота, режим работы и другое,
iwlist <dev> channel – получить список доступных каналов и частот доступных устройств,
iwlist <dev> rate – список скоростей доступных устройств,
iwlist <dev> key – список и размеры ключей для подключения,
iwlist <dev> power – показать режимы управления питанием адаптера,
iwlist <dev> txpower – показать диапазоны доступной мощности сигнала адаптера,
iwlist <dev> ap – получить список точек доступа и уровень их сигнала.

Команды `Madwifi-ng`

`MadWiFi` поддерживает виртуальные точки доступа (**`VAPS`**), что помогает эмулировать несколько беспроводных устройств на основе одной физической карты (основная карта = **`wifi0`**). Поддерживается картами, созданными на основе чипсетов **`Atheros`**. Для краткости устройство называется **`athx`** – хотя в конкретной системе она может называться **`ath0`** или **`ath1`**:

`wlanconfig athx destroy` – убрать виртуальную точку доступа **`athx`**,

`wlanconfig athx create wlandev wifi0 wlanmode sta` – создать работающую в режиме **`managed`** виртуальную точку доступа **`athx`**,

`wlanconfig athx create wlandev wifi0 wlanmode ap` – создать виртуальную точку доступа **`athx`**,

`wlanconfig athx create wlandev wifi0 wlanmode adhoc` – создать работающую в режиме **`Ad-Hoc`** виртуальную точку доступа **`athx`**,

`wlanconfig athx create wlandev wifi0 wlanmode monitor` – создать работающую в режиме **`Monitor`** виртуальную точку доступа **`athx`**.

ПРИЛОЖЕНИЕ 2

АРХИТЕКТУРНЫЕ ОСОБЕННОСТИ ФАЙЛОВОЙ СИСТЕМЫ EXT4FS

С 2006 года в операционных системах Linux начато использование новой файловой системы **ext4fs**, сначала в виде дополнения к существующей ФС **ext3fs**, а затем, с 2008 года, в виде стабильной, законченной файловой системы. Однако многие из запланированных преимуществ ФС еще не реализованы ее разработчиками, и эта система продолжает развиваться и дорабатываться. Новая ФС содержит много особенностей и сильно отличается от предшествующей версии.

Несмотря на некоторую незавершенность файловой системы, она уже устанавливается по умолчанию во многих дистрибутивах Linux и стремительно набирает популярность. 15 января 2010 года компания Google объявила о переводе своих серверов с файловой системы **ext2** на файловую систему **ext4**.

Поскольку файловая система разработана недавно и все еще находится в процессе развития, документации на русском языке по ней практически нет. Отсутствуют и утилиты для исследования файловой системы и ее структур. Формат данных, хранимых на диске, в **ext4** несколько изменился, поэтому программы, предназначенные для исследования и восстановления данных в системах **ext2** и **ext3**, с новой ФС работают некорректно.

Использование экстентов

В файловой системе адресация данных выполнялась традиционным образом, поблочно. Такой способ адресации становится менее эффективным с ростом размера файлов. Экстенты позволяют адресовать большое количество (до 128 Мб) последовательно идущих блоков одним дескриптором. До 4 указателей на экстенты может размещаться непосредственно в индексном дескрипторе, что достаточно для файлов маленького и среднего размера [15]. Экстенты используются многими файловыми системами, в том числе: **HFS Plus** (для Mac OS), **NTFS** (для Windows), **UDF**, **XFS**, **JFS**, **Reiser4**, **Btrfs** и **HPFS**.

48-битные номера блоков

Ext3 является 32-битной файловой системой. Это значит, что ее максимальный размер при размере блока в 4 Кб составляет $2^{(32+12)} = 17$ Тб. **Ext4** при размере блока 4 Кб позволяет адресовать до одного экзабайта: $2^{48} \cdot 4KB = 2^{50} \cdot 1KB = 2^{60} B = 1EB$. Многие популярные файловые системы, например XFS, являются 64-битными. 48-битная адресация блоков в **ext4** была выбрана вместо полной 64-битной адресации по причине того, что предел для файловой системы в 1 Эб еще много лет будет достаточным.

При существующих скоростях одна полная проверка файловой системы в 1 Эб утилитой **e2fsck** займет 119 лет, что в 65 536 раз меньше, чем потребуется для системы с 64-битной адресацией [16]. Но возможности для поддержки 64-битной адресации создаются уже сейчас: некоторые поля системы зарезервированы под 64-битную адресацию.

Новые алгоритмы выделения блоков и индексных дескрипторов

Выделение блоков группами (multiblock allocation). Файловая система хранит не только информацию о местоположении свободных блоков, но и количество свободных блоков, идущих друг за другом. При выделении места файловая система находит такой фрагмент, в который данные могут быть записаны без фрагментации. Это снижает уровень фрагментации файловой системы в целом.

Ext3 одинаково выделяла блоки как для маленьких, так и для больших файлов, что увеличивало фрагментированность ФС. **Ext4** в зависимости от размера файла применяет разные алгоритмы выделения дискового пространства. Маленькие файлы из одной директории ФС старается держать рядом, чтобы минимизировать время и количество обращений к диску. Большие файлы размещаются отдельно, в непрерывной области памяти [18].

Отложенное выделение блоков (delayed allocation). Выделение блоков для хранения данных файла происходит непосредственно перед физической записью на диск (например, при вызове **sync**), а не при вызове **write**. В результате операции выделения блоков можно делать не по одной, а группами, что в свою очередь минимизирует фрагментацию, снижает нагрузку на процессор и ускоряет процесс выделения блоков. Отложенное выделение блоков также предотвращает ненужное резервирование блоков для короткоживущих файлов. С другой стороны, это увеличивает риск потери данных в случае внезапного пропадания питания.

Предварительное выделение (persistent preallocation). Сейчас приложения для того, чтобы гарантированно занять какое-то место, заполняют нулями. В **ext4** появилась возможность зарезервировать множество блоков для записи и не тратить на инициализацию лишнее время. Если приложение попробует прочитать данные, оно получит сообщение о том, что они не проинициализированы. Таким образом, несанкционированно прочитать удалённые данные не получится.

Резервирование индексных дескрипторов при создании каталога (directory inodes reservation). При создании каталога резервируется несколько индексных дескрипторов. Впоследствии при создании файлов в этом каталоге сначала используются зарезервированные индексные дескрипторы, и если таких не осталось, выполняется обычная процедура выделения **inode** [17].

Формат индексного дескриптора

Размер индексного дескриптора. Размер индексного дескриптора (по умолчанию) увеличен с 128 до 256 байтов. Это дает возможность реализовать некоторые преимущества, например хранить расширенные атрибуты в самом дескрипторе, использовать более точные временные метки и т. д.

Версия индексного дескриптора. В индексном дескрипторе появился номер, который увеличивается при каждом изменении индексного дескриптора файла. Это может использоваться, например, для того чтобы узнавать, изменился ли файл.

Хранение расширенных атрибутов в индексном дескрипторе (EA in inode). Хранение расширенных атрибутов, таких как ACL, атрибутов SELinux и прочих, позволяет повысить производительность. Атрибуты, для которых недостаточно места в индексном дескрипторе, хранятся в отдельном блоке размером 4 Кб. Предполагается снять это ограничение в будущем [17].

Временные отметки с наносекундной точностью

Временные отметки в **ext3** имеют секундную точность. Однако некоторым приложениям, особенно клиент-серверным, требуется более высокая точность. **Ext4** предоставляет такую возможность, увеличивая точность времен, хранимых в индексном дескрипторе, суперблоке и журнале. Достигается это путем увеличения разрядности временной отметки с 4 до 8 байтов. Диапазон хранящихся времён тоже расширен: если раньше верхней границей хранимого времени было 18 января 2038 года, то теперь – это 25 апреля 2514 года [17].

Общая организация системы

Основные принципы организации хранения данных в **ext4** соответствуют организации файловых систем **ext2** и **ext3**. Весь жесткий диск разбивается на блоки данных, которые являются минимально адресуемыми единицами информации.

Здесь и проявляется основное отличие **ext4** от предшественницы – теперь номера блоков могут быть 48-битными, если есть поддержка со стороны ядра.

Блоки группируются на группы блоков, в каждой группе есть битовая карта блоков, битовая карта индексных дескрипторов, таблица индексных дескрипторов и собственно блоки данных. Битовая карта блоков определяет свободные и занятые блоки в группе установкой или сбросом соответствующего бита. Битовая карта индексных дескрипторов аналогичным образом определяет свободные или занятые дескрипторы. Статическая таблица индексных дескрипторов хранит информацию о файлах, которым эти дескрипторы соответствуют.

Начальной точкой файловой системы является суперблок. Он находится по смещению 1024 байта от начала диска (в самом начале диска размещается загрузчик) и занимает 1 Кб. Суперблок определяет многие важные параметры файловой системы: раз-

мер блока, число блоков, параметры групп блоков, параметры системы и т.д.

В следующем блоке располагаются дескрипторы групп блоков. Они указывают на блоки, в которых находятся битовые карты блоков и индексных дескрипторов, начало таблицы индексных дескрипторов и другую информацию о группе [17].

Традиционная организация файловой системы предполагает, что группы блоков идут на диске одна за другой и каждая начинается с битовых карт и таблиц индексных дескрипторов. Такая разбивка файловой системы представлена на рис. П.2.1.

В **ext4** вводится опция так называемых гибких групп блоков. Гибкая группа блоков состоит из нескольких групп блоков. В пределах гибкой группы битовые карты и таблицы индексных дескрипторов хранятся вместе, и лишь затем идут блоки данных [18]. Это позволяет уменьшить время на чтение информации о свободных блоках и дескрипторах, а также ускорить проверку диска с помощью **e2fsck**.

Разбивка диска при использовании опции «гибкая группа блоков» представлена на рис. П.2.2.

Гибкие группы блоков можно наблюдать с помощью редактора **debugfs**. Команда **debugfs -R stats <device>** выводит в том числе и группы блоков. На рис. П.2.3 видно, что в каждой группе блоков битовые карты имеют не смежные номера блоков, зато в соседних группах номера соответствующих битовых карт возрастают последовательно.

Ext4 не резервирует блоки для журнала после описателей групп блоков, как это делалось в **ext3**. Вместо этого журнал имеет определенный индексный дескриптор и хранится в выделенных для него блоках, которые в принципе могут находиться в любой области диска. Перед началом битовых карт групп блоков пространство зарезервировано под описатели групп. В **ext4** существует понятие «системная зона» – это блоки, выделенные под хранение метаданных, которые нельзя использовать индексным дескрипторам [18]. К метаданным относятся суперблок, в том числе и его копии, описатели групп блоков и журнал.

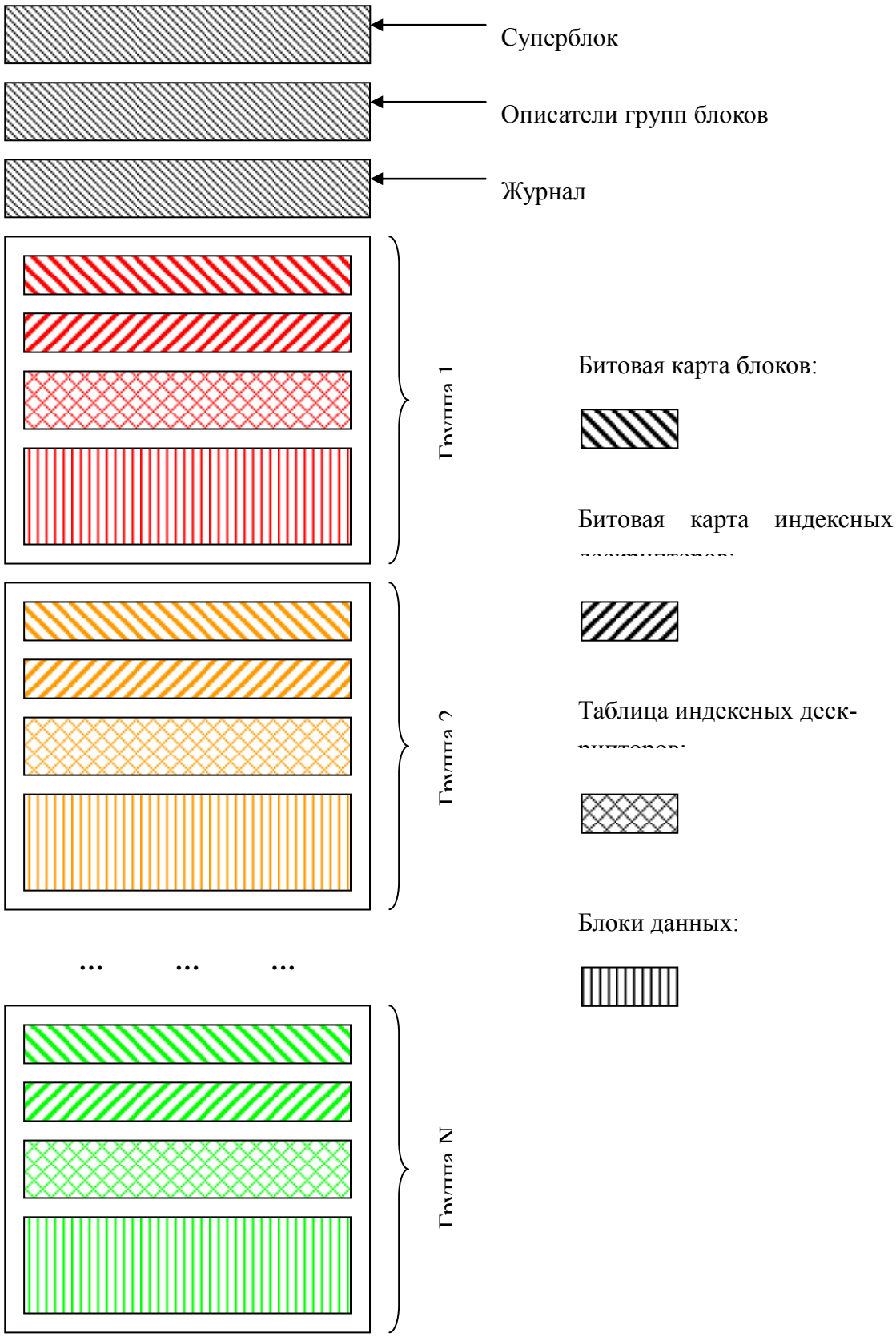

Рис. П.2.1. Группы блоков в файловой системе **`ext3fs`**

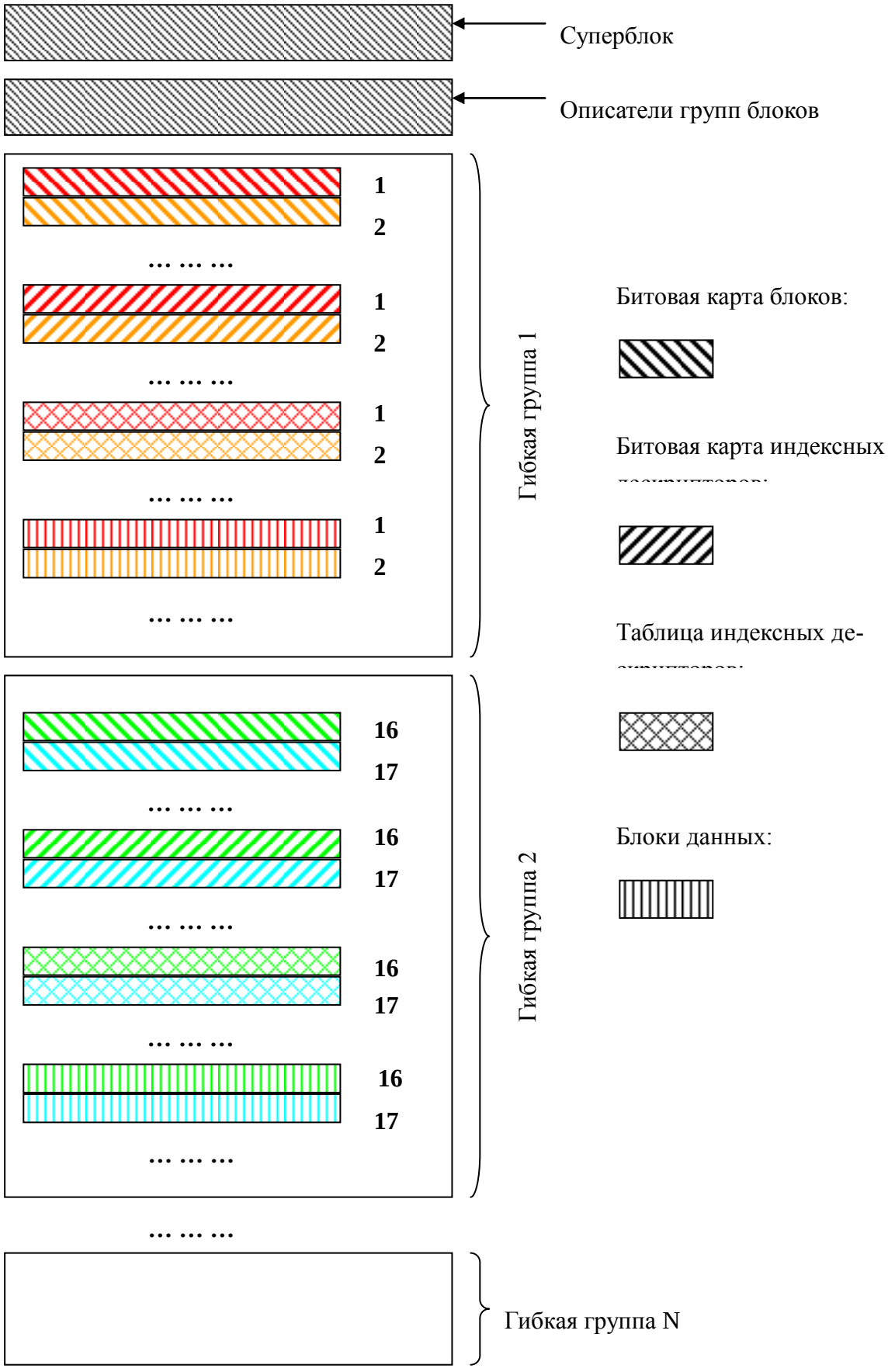
Суперблок
Описатели групп блоков
1
2
… … …
1
2
… … …
1
2
… … …
1
2
… … …
Гибкая группа 1
Битовая карта блоков:
Битовая карта индексных
Таблица индексных де-
16
17
… … …
16
17
… … …
16
17
… … …
16
17
… … …
Гибкая группа 2
Блоки данных:
… … …
Гибкая группа N

Рис. П.2.2. Гибкие группы блоков в файловой системе **ext4fs**

```
Group  0: block bitmap at 591, inode bitmap at 607, inode table at 623
           3436 free blocks, 29 free inodes, 4 used directories, 0 unused
inodes
           [Checksum 0x6dab]
 Group  1: block bitmap at 592, inode bitmap at 608, inode table at 1134
           4182 free blocks, 32 free inodes, 753 used directories, 0 unused
inodes
           [Checksum 0x2f06]
 Group  2: block bitmap at 593, inode bitmap at 609, inode table at 1645
           5587 free blocks, 24 free inodes, 702 used directories, 0 unused
inodes
           [Checksum 0x29d8]
 Group  3: block bitmap at 594, inode bitmap at 610, inode table at 2156
           7755 free blocks, 50 free inodes, 625 used directories, 0 unused
inodes
           [Checksum 0x5b66]
 Group  4: block bitmap at 595, inode bitmap at 611, inode table at 2667
           6894 free blocks, 186 free inodes, 1488 used directories, 0 un-
used inodes
           [Checksum 0x9d36]
 Group  5: block bitmap at 596, inode bitmap at 612, inode table at 3178
           8311 free blocks, 70 free inodes, 362 used directories, 0 unused
inodes
           [Checksum 0xf453]
 Group  6: block bitmap at 597, inode bitmap at 613, inode table at 3689
           7086 free blocks, 0 free inodes, 2167 used directories, 0 unused
inodes
           [Checksum 0x07c0]
 Group  7: block bitmap at 598, inode bitmap at 614, inode table at 4200
           2314 free blocks, 0 free inodes, 1658 used directories, 0 unused
inodes
           [Checksum 0xdc31]
 Group  8: block bitmap at 599, inode bitmap at 615, inode table at 4711
           4177 free blocks, 141 free inodes, 450 used directories, 0 un-
used inodes
           [Checksum 0x50f9]
```

Рис. П.2.3. Фрагмент гибкой группы блоков, выводимой **debugfs**

Кроме собственно числа блоков в ФС, ее размер ограничен также максимально возможным числом групп блоков. В **ext3** копии всех описателей групп блоков должны храниться в первой группе блоков. В **ext4** размер описателя группы блоков составляет 64 байта (если используется 48-битная адресация блоков). При размере группы блоков в 128 Мб всего в **ext4** может быть $2^{27}/64 = 2^{21}$ групп блоков. Это ограничивает размер файловой системы до $2^{21} \cdot 2^{27} = 2^{48}$ байтов, или 256 Тб. Для снятия этого ограничения используют метагруппы блоков. Файловые системы **ext4** разделены на метагруппы блоков [17].

Метагруппа – это кластер из групп блоков, описатель которого может храниться в одном блоке. При размере блока в 4 Кб одна метагруппа может содержать 64 группы блоков, или 8 Гб. Описатели групп блоков теперь хранятся не в первой группе блоков, а «рассыпаны» по всей файловой системе, что записано в первый блок метагруппы. Резервные копии хранятся во втором и последнем блоках метагруппы [15].

Таким образом, максимальное число групп блоков в системе может достигать 2^{32}, что соответствует 1 Эб дискового пространства.

Суперблок и описатели групп блоков

Формат суперблока определен в заголовочном файле **ext4.h** исходного кода ядра ОС Linux как структура **ext4_super_block**.

Таблица П.2.1

Формат суперблока

Размер поля, байт	**Смещение**		**Назначение**
	десятичное	**шестнадцатеричное**	
4	0	0	Число индексных дескрипторов в ФС
4	4	4h	Число блоков в ФС
4	8	8h	Число блоков, зарезервированных для нужд суперпользователя
4	12	Ch	Число свободных блоков
4	16	10h	Число свободных индексных дескрипторов
4	20	14h	Номер первого блока, содержащего данные (0 или 1)
4	24	18h	Индикатор размера логического блока
4	28	1Ch	Индикатор размера фрагментов
4	32	20h	Число блоков в каждой группе блоков
4	36	24h	Число фрагментов в каждой группе блоков
4	40	28h	Число индексных дескрипторов в каждой группе блоков

4	44	2Ch	Время последнего монтирования ФС
4	48	30h	Время последней записи в ФС
2	52	34h	Число монтирований ФС. При достижении предельного числа монтирований обнуляется
2	54	36h	Предельное число монтирований ФС
2	56	38h	Магическое число **`0xEF53`**, указывающее, что ФС принадлежит к **`ext*fs`**
2	58	3Ah	Флаги текущего состояния ФС
2	60	3Ch	Флаги процедур обработки сообщений об ошибках
2	62	3Eh	Младший номер устройства
4	64	40h	Время последней проверки ФС
4	68	44h	Максимальный период времени между проверками
4	72	48h	Указание на тип ОС, в которой создана ФС. Коды операционных систем: 0 – Linux, 1 – Hurd, 2 – Masix, 3 – FreeBSD, 4 – Lites
4	76	4Ch	Версия ФС
2	80	50h	UID по умолчанию для зарезервированных блоков
2	82	52h	GID по умолчанию для зарезервированных блоков
4	84	54h	Номер первого индексного дескриптора, не зарезервированного системой
2	88	58h	Размер индексного дескриптора
2	90	5Ah	Номер группы блоков, в которой находится этот суперблок

Окончание табл. П.2.1

4	92	5Ch	Флаги COMPAT
4	96	60h	Флаги INCOMPAT
4	100	64h	Флаги ROCOMPAT
16	104	68h	UUID тома
16	120	78h	Имя тома
64	136	88h	Каталог, в котором последний раз была смонтирована система
4	200	C8h	Алгоритм использования битовых карт (для сжатия)
1	204	CCh	Число блоков, которые надо попытаться зарезервировать. Используется, если включена COMPAT-опция DIR_PREALLOC
1	205	CDh	То же для директорий. Используется, если включена COMPAT-опция DIR_PREALLOC
2	206	CEh	Зарезервированные блоки на групповой дескриптор.

			Используется, если включена COMPAT-опция DIR_PREALLOC
16	208	D0h	UUID журнального суперблока
4	224	E0h	Номер индексного дескриптора файла журнала
4	228	E4h	Номер устройства файла журнала
4	232	E8h	Указатель на список индексных дескрипторов, которые были удалены, но использовались каким-либо приложением в момент удаления
16	236	ECh	Зерно для хэш-функции H-дерева
1	252	FCh	Версия хэш-функции по умолчанию
1	253	FDh	Зарезервировано
2	254	FEh	Размер описателя группы блоков
4	256	100h	Параметры монтирования по умолчанию
4	260	104h	Первая метагруппа блоков
4	264	108h	Время создания ФС
68	268	10Ch	Бэкап для файла журнала
4	336	150h	Старшие 32 значащих бита числа блоков
4	340	154h	Старшие 32 бита зарезервированных блоков
4	344	158h	Старшие 32 бита числа свободных блоков
2	348	15Ch	Минимальный размер фиксированных полей индексного дескриптора
2	350	15Eh	Новые индексные дескрипторы должны резервировать столько байт для фиксированных полей
4	352	160h	Смешанные флаги
2	356	164h	RAID stride
2	358	166h	Число секунд в ожидании проверки MMP
8	360	168h	Блок для защиты от множественного монтирования
4	368	170h	Блоки на всех дисках с данными (N*stride)
1	372	174h	Размер гибкой группы блоков
1	373	175h	Зарезервировано
2	374	176h	Зарезервировано
8	376	178h	Число килобайт, записанных на диск за время жизни системы
640	384	180h	Заполнение до конца суперблока

Вывести информацию, содержащуюся в суперблоке, можно с помощью команды **`debugfs –R stats <device>`**. Результат выполнения этой команды приведен на рис. П.2.4.

```
Filesystem volume name:   <none>
```

```
Last mounted on:          /
Filesystem UUID:          aaca0a90-3a12-41b8-9f3d-c82b1b48680c
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index
filetype needs_recovery extent flex_bg sparse_super large_file huge_file
uninit_bg dir_nlink extra_isize
Filesystem flags:         signed_directory_hash
Default mount options:    (none)
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              605024
Block count:              2415766
Reserved block count:     120788
Free blocks:              1211855
Free inodes:              378277
First block:              0
Block size:               4096
Fragment size:            4096
Reserved GDT blocks:      589
Blocks per group:         32768
Fragments per group:      32768
Inodes per group:         8176
Inode blocks per group:   511
Flex block group size:    16
Filesystem created:       Mon Nov  2 16:24:05 2009
Last mount time:          Sun Feb 21 17:46:13 2010
Last write time:          Mon Feb 15 18:17:16 2010
Mount count:              14
Maximum mount count:      29
Last checked:             Sat Feb 13 18:03:59 2010
Check interval:           15552000 (6 months)
Next check after:         Thu Aug 12 19:03:59 2010
Lifetime writes:          29 GB
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:           256
```

```
Required extra isize:     28
Desired extra isize:      28
Journal inode:            8
First orphan inode:       142509
Default directory hash:   half_md4
Directory Hash Seed:      c24ea3e3-8847-4a42-8330-498487728212
Journal backup:           inode blocks
Directories:              30357
```

Рис. П.2.4. Вывод информации о файловой системе отладчиком **debugfs**

Для обеспечения совместимости с предыдущими версиями файловой системы и нормальной работы утилиты проверки файловой системы первые поля суперблока не претерпели никаких изменений. Основы для будущих возможностей **ext4** были заранее заложены еще в исходном коде файловой системы **ext3**. Так, в суперблоке **ext3** есть зарезервированные поля для 64-битных номеров блоков, размера гибкой группы блоков, размера фиксированных полей индексного дескриптора и т. д. Совершенно новыми для **ext4** оказались только 2 поля: размер описателя группы блоков и общее число килобайтов, записанных на диск за «время жизни» файловой системы.

Ниже приведены различные флаги файловой системы, которые содержатся в суперблоке. В табл. П.2.2 приведены значения флагов состояния файловой системы. Этими флагами файловая система помечается после проверки утилитой **e2fsck** в зависимости от результатов работы утилиты. В табл. П.2.3 приведены флаги, определяющие реакцию ФС при обнаружении ошибок файловой системы.

Таблица П.2.2

Флаги текущего состояния файловой системы

Флаг	**Описание**
0x0001	При размонтировании не произошло ошибок
0x0002	Замечены ошибки
0x0004	Режим обработки логически удаленных, но все еще открытых каким-либо процессом индексных дескрипторов

Таблица П.2.3

Флаги процедур обработки сообщений об ошибке

Флаг	**Описание**
0x0001	При наличии ошибок продолжить работу ФС
0x0002	Перемонтировать ФС в режим «только для чтения»
0x0004	«Паника» при ошибках: вывод сообщения об ошибке, прекращение рабо-

	ты файловой системы и уведомление пользователя о необходимости перезагрузки

Основные опции ФС, определяющие ее свойства, содержатся в трех полях суперблока: это флаги COMPAT, INCOMPAT и ROCOMPAT. В **ext4** появились новые опции, а именно: поддержка очень больших файлов, контрольное суммирование описателей групп блоков, поддержка неограниченного числа подкаталогов, поддержка протокола MMP, введение фиксированных полей в индексном дескрипторе, поддержка экстентов, 64-битной адресации блоков и гибких групп блоков.

Описатели групп блоков занимают следующий за суперблоком блок данных. Формат описателя группы блоков определен в заголовочном файле **ext4.h** как структура **ext4_group_desc**. Эта структура представлена в табл. П.2.4. В **ext4** описатель групп блоков имеет размер 64 байта, вместо 32 байтов в **ext3**. Однако если не включена INCOMPAT-опция EXT_64BIT (64-битная поддержка), система монтируется с 32-байтными описателями [18].

В **ext4** в описатель групп блоков вводятся новые поля: флаги, контрольная сумма описателя, число неиспользованных индексных дескрипторов, а также старшие значащие биты в случае использования 64-байтного описателя. Число неиспользованных индексных дескрипторов в общем случае отличается от числа свободных дескрипторов: если в группе был удален индексный дескриптор, число свободных дескрипторов станет на 1 больше, а число неиспользованных не изменится. Флаги описателя группы блоков EXT4_BG приведены в табл. П.2.5, они позволяют ускорить проверку файловой системы: если битовая карта блоков или индексных дескрипторов помечена как неиспользованная, e2fsck пропускает ее, делая проверку только в том случае, если не совпадает контрольная сумма.

Таблица П.2.4

Формат описателя группы блоков

Размер поля, байт	**Смещение, dec (hex)**	**Назначение**
4	0	Адрес битовой карты блоков (младшие значащие биты)
4	4 (4h)	Адрес битовой карты индексных дескрипторов
4	8 (8h)	Адрес блока с началом таблицы индексных дескрипторов
2	12 (Ch)	Число свободных блоков
2	14 (Eh)	Число свободных индексных дескрипторов
2	16 (10h)	Число директорий
2	18 (12h)	Флаги EXT4_BG
4x2	20 (14h)	Зарезервировано под контрольные суммы битовых карт блоков и индексных дескрипторов

2	28 (1Ah)	Число неиспользованных индексных дескрипторов
2	30 (1Ch)	Контрольная сумма нескольких полей
4	32 (1Eh)	Старшие значащие биты адреса битовой карты блоков
4	36 (22h)	Старшие значащие биты адреса битовой карты индексных дескрипторов
4	40 (26h)	Старшие значащие биты адреса первого блока таблицы индексных дескрипторов
2	44 (2Ah)	Старшие значащие биты числа свободных блоков
2	46 (2Ch)	Старшие значащие биты числа свободных индексных дескрипторов
2	48 (2Eh)	Старшие значащие биты числа директорий
2	50 (30h)	Старшие значащие биты числа неиспользованных индексных дескрипторов
14	52 (32h)	Заполнение

Таблица П.2.5

Флаги описателя группы блоков

Флаг	**Название**	**Описание**
0x0001	INODE_UNINIT	Таблица и битовая карта индексных дескрипторов не используется
0x0002	BLOCK_UNINIT	Битовая карта блоков не используется
0x0004	INODE_ZEROED	Таблица индексных дескрипторов заполнена нулями

Формат индексного дескриптора

Формат индексного дескриптора описан в заголовочном файле **`ext4.h`** как структура **`ext4_inode`**, формат приведен в табл. П.2.6.

Ext4, как и предыдущая файловая система, резервирует некоторые индексные дескрипторы, например 2-й отводится для корневой директории, 8-й – для журнала транзакций. В ext4 по умолчанию используются 256-байтные индексные дескрипторы. Чтобы не переписывать заново многочисленные функции ядра и утилиту проверки файловой системы **`e2fsck`**, первые 128 байтов индексного дескриптора оставили фактически без изменений [15]. Обобщенная структура индексного дескриптора представлена на рис. П.2.5.

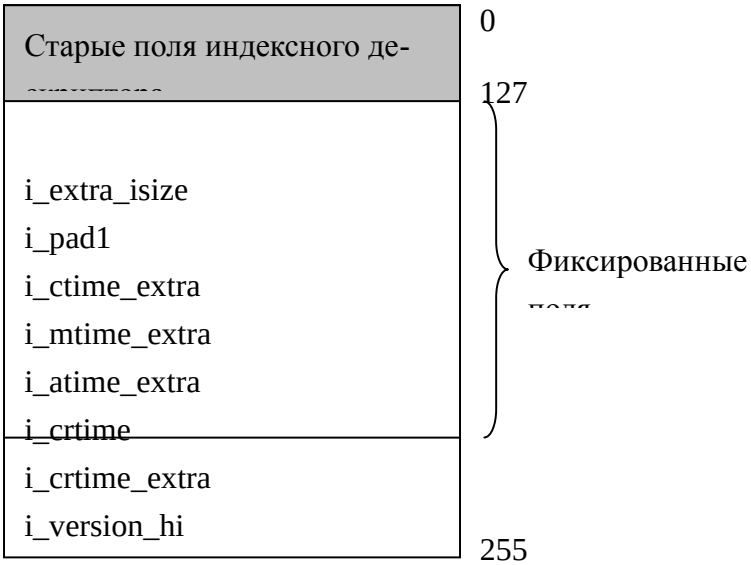

Рис. П.2.5. Индексный дескриптор в **ext4**

Фиксированные поля выделяются динамически, в зависимости от того, какая информация есть у текущего ядра об этих полях. Размер области фиксированных полей хранится в каждом индексном дескрипторе в поле *i_extra_isize*. В самом суперблоке также содержатся два поля: *s_min_extra_isize*, которое гарантирует минимальный объем области фиксированных полей в каждом индексном дескрипторе, и *s_want_extra_isize*, которое указывает на желаемый размер области фиксированных полей во вновь создаваемом индексном дескрипторе, но вовсе не гарантирует, что именно такой объем будет в нем выделен [18].

Таблица П.2.6

Формат индексного дескриптора

Размер, байт	**Смещение (dec)**	**Смещение (hex)**	**Назначение**
2	0	0	Тип файла и права доступа к нему
2	2	2h	Младшие 16 бит идентификатора пользователя
4	4	4h	Размер файла в байтах
4	8	8h	Время последнего доступа
4	12	Ch	Время изменения индексного дескриптора
4	16	10h	Время последнего изменения данных файла
4	20	14h	Время удаления файла
2	24	18h	Младшие 16 бит идентификатора группы
2	26	1Ah	Число жестких ссылок на файл
4	28	1Ch	Число блоков или секторов, занимаемых файлом
4	32	20h	Флаги файла

4	36	24h	Зарезервировано для ОС
15x4 или 5x12	40	28h	В зависимости от способа адресации 15 указателей на блоки данных или 1 заголовок экстента (12 байтов) и непосредственно 4 экстента
4	100	64h	Версия файла (для NFS)
4	104	68h	ACL файла
4	108	6Ch	Старшие 32 бита размера файла в байтах
4	112	70h	Устаревшее поле: адрес фрагмента
12	116	74h	Структура, зависящая от типа ОС, определяющая номера фрагментов
2	128	80h	Размер фиксированных полей inode
2	130	82h	Зарезервировано
4	132	84h	Дополнительные 32 бита ctime
4	136	88h	Дополнительные 32 бита mtime
4	140	8Ch	Дополнительные 32 бита atime
4	144	90h	Время создания файла
4	148	94h	Дополнительные 32 бита к ВО создания файла
4	152	98h	Старшие 32 бита версии файла

Оставшееся пространство в индексном дескрипторе может быть использовано для хранения быстрых расширенных атрибутов файла. Это позволяет не искать каждый раз внешний блок данных с расширенными атрибутами, что для некоторых приложений может весьма увеличить производительность [15].

В **`ext4`** поле, где располагалось время создания файла, заменили на поле с временной отметкой последнего изменения индексного дескриптора. Время создания файла вынесено за пределы стандартных полей.

Индексный дескриптор содержит флаги, которые называются файловыми атрибутами и выводятся утилитой **`lsattr`**. Эти флаги приведены в табл. П.2.7.

Таблица П.2.7

Флаги индексного дескриптора

Флаг	**Название**	**Описание**
0x00000001	SECRM_FL	Безопасное удаление файла
0x00000002	UNRM_FL	Неудаляемый файл
0x00000004	COMPR_FL	Сжатый файл
0x00000008	SYNC_FL	Синхронное обновление
0x00000010	IMMUTABLE_FL	Неизменяемый файл
0x00000020	APPEND_FL	Любая запись может только добавляться в конец файла

0x00000040	NODUMP_FL	Не дампировать данные файла
0x00000080	NOATIME_FL	Не обновлять время последнего доступа к файлу
0x00004000	JOURNAL_DATA_FL	Данные файла должны журналироваться
0x00040000	HUGE_FILE_FL	Гигантский файл. Размер файла указывается в логических блоках, а не в секторах
0x00080000	EXTENTS_FL	Файл использует экстенты

Расширенные атрибуты

В **ext3** расширенные атрибуты хранились в отдельном блоке на диске. В индексном дескрипторе в стандартных полях содержится поле ACL, в котором хранится номер блока с расширенными атрибутами. Атрибуты могут быть прочитаны независимо от основных данных файла. Несколько индексных дескрипторов с одинаковыми расширенными атрибутами могут ссылаться на один и тот же блок. Сходство блоков определяется по истории недавних вызовов файлов. **Ext4** может хранить расширенные атрибуты как в отдельном блоке, так и прямо в индексном дескрипторе, если они туда поместятся.

ACL или Access Control List состоит из набора записей. Каждый из трех типов пользователей представлен записью в ACL. Минимальный ACL содержит 3 записи: владелец, группа владельца и остальные. Этот набор указан в первых 2 байтах индексного дескриптора и выводится командой **`ls -l`**. Расширенный ACL содержит больше записей: в них можно указывать конкретного пользователя и конкретную группу, а также маску [18].

Записи в ACL могут быть нескольких типов, которые представлены в табл. П.2.8. Каждый ACL состоит из заголовка, описателей элементов (или просто элементов) и их значений.

Таблица П.2.8

Типы записей в ACL

Тип	**Текстовая форма**
Owner	user::rwx
Named user	user:name:rwx
Owning group	group::rwx
Named group	group:name:rwx
Mask	mask::rwx
Others	other::rwx

Заголовки атрибутов, хранимых в индексном дескрипторе и в блоке, разные, но

хранятся элементы списков в одинаковом формате. Формат ACL, распределенного по свободному пространству блока или внутри индексного дескриптора, может быть представлен в следующем виде:

Заголовок

Элемент 1

Элемент 2

Элемент 3

4 нулевых байта

…

Величина 3

Величина 2

Величина 1

В отдельном блоке диска дескрипторы элементов сортируются, в индексном дескрипторе элементы несортированы. Значения атрибутов (величины) выровнены по концу блока без особого порядка [17].

Заголовки в блоке и в индексном дескрипторе отличаются друг от друга. В табл. П.2.9 представлен формат заголовка в блоке.

Таблица П.2.9

Формат заголовка ACL в блоке

Размер, байт	**Поле**
4	Магическое число 0xEA020000
4	Число ссылок
4	Число использованных блоков на диске
4	Хэш-функция от всех атрибутов
4x4	Зарезервировано

В качестве заголовка ACL в индексном дескрипторе используется магическое число **`0xEA020000`**. Описатели элементов расширенного атрибута имеют формат, представленный в табл. 2.10.

Таблица П.2.10

Формат элемента расширенного атрибута

Размер, байт	**Поле**
1	Длина имени

1	Индекс имени атрибута
2	Смещение величины атрибута от начала блока
4	Номер блока, в котором хранится значение
4	Размер величины атрибута
4	Хэш-функция от имени и значения атрибута
-	Имя атрибута

Установить расширенные атрибуты для файла можно утилитой **setfacl**. Например, установить для пользователя право на запись в файл можно командой **setfacl –m u:username:rw- <path_to_file>**. Просмотреть расширенные атрибуты можно командой **getfacl <path_to_file>**.

В **ext4** расширенные атрибуты хранятся прямо в индексном дескрипторе, что можно наблюдать на рис. П.2.6, где представлен индексный дескриптор с расширенными атрибутами, хранимыми внутри самого дескриптора за пределами фиксированных полей.

В листинге выделено подчеркиванием магическое число **0xEA020000**, идущее сразу после фиксированных полей индексного дескриптора. Следующие 16 байтов занимает заголовок элемента атрибута (его имя имеет нулевую длину), а последние 28 байтов (именно такой размер указан в заголовке элемента) в блоке занимает значение атрибута. При этом поле «ACL» индексного дескриптора (выделено подчеркиванием) содержит нулевое значение, то есть не содержит ссылок на внешний блок данных.

```
0x00000000  B4 81 00 00 44 42 00 00 : 1E 23 82 4B 55 21 82 4B  ....DB...#.KU!.K
0x00000010  79 68 81 4B 00 00 00 00 : 00 00 01 00 28 00 00 00  yh.K........(...
0x00000020  00 00 08 00 01 00 00 00 : 0A F3 01 00 04 00 00 00  ................
0x00000030  00 00 00 00 00 00 00 00 : 05 00 00 00 EA 88 0C 00  ................
0x00000040  00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00  ................
0x00000050  00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00  ................
0x00000060  00 00 00 00 A9 91 3B 8A : 00 00 00 00 00 00 00 00  ......;.........
0x00000070  00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00  ................
0x00000080  1C 00 00 00 C4 6B A7 33 : F4 8A DA C3 20 4E 14 B9  .....k.3.... N..
0x00000090  79 68 81 4B CC 7A F7 AC : 00 00 00 00 00 00 02 EA  yh.K.z..........
0x000000A0  00 02 44 00 00 00 00 00 : 1C 00 00 00 00 00 00 00  ..D.............
0x000000B0  00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00  ................
0x000000C0  00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00  ................
0x000000D0  00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00  ................
0x000000E0  00 00 00 00 01 00 00 00 : 01 00 06 00 02 00 06 00  ................
```

```
0x000000F0  E8 03 00 00 04 00 04 00 : 10 00 06 00 20 00 04 00  ............ ...
```

Рис. П.2.6. Индексный дескриптор с расширенными атрибутами

Формат директорий

В **ext3** директория представляет собой фиксированный список сопоставлений индексного дескриптора файла и его имени. Структура записи в директории показана в табл. П.2.11.

Таблица П.2.11

Структура записи в директории в **ext3**

Размер, байт	**Поле**
4	Индексный дескриптор
2	Длина записи
2	Длина имени файла
255	Имя файла

В **ext4** структура записи в директории претерпела небольшие изменения, что показано в табл. П.2.12.

Таблица П.2.12

Структура записи в директории в **ext4**

Размер, байт	**Поле**
4	Индексный дескриптор
2	Длина записи
1	Длина имени файла
1	Тип файла
255	Имя файла

Типы файлов: 0 – неизвестный, 1 – обычный, 2 – каталог, 3 – файл символьного устройства, 4 – файл блочного устройства, 5 – именованный канал, 6 – сокет, 7 – символическая ссылка.

В **ext4** для превышения лимита в 32 000 подкаталогов введено индексирование директорий. Если установлен соответствующий флаг файловой системы, записи в директориях хэшируются. Неиндексированная директория занимает минимум 1 блок. Минимальный размер индексированной директории должен составлять 3 блока. 0-й блок

при этом становится корнем дерева, остальные становятся «листьями».

Блоки-«листья» представляют собой традиционный список имен файлов и индексных дескрипторов. Поэтому, если корень дерева окажется поврежденным, можно будет отыскать файлы, используя традиционную схему. Корень дерева содержит хэш-функции блоков-«листьев», каждый такой хэш представляет последовательность записей директории. Хэш в **ext4** может быть как 32-битным, так и 64-битным. Для обеспечения совместимости с файловыми системами, не использующими индексирование, первые 8 байтов корня дерева содержат указание на следующий блок директории, т. е. ядро, не поддерживающее индексирование, будет игнорировать набор хэш-функций как удаленные из директории файлы.

Хэш-версии (алгоритмы хэширования) определяются в суперблоке файловой системы и могут принимать следующие значения: 0 – LEGACY, 1 – HALF MD4, 2 – TEA, 3 – LEGACY UNSIGNED, 4 – HALF MD4 UNSIGNED, 5 – TEA UNSIGNED.

Для хэширования используется так называемое «зерно» – некоторое инициализирующее значение, которое также определено в соответствующем поле суперблока. Если зерно содержит нули по какой-то причине, используется зерно по умолчанию [15].

Реализация H-дерева сейчас ограничена 510 – 511 4-килобайтными блоками-«листьями»: именно такое количество может быть проиндексировано 2-уровневым деревом (дерево должно иметь постоянную глубину). Возможно, в будущем количество уровней увеличат до 3. Также в директориях существует ограничение на размер файла в 2 Гб, потому что размер поля i_size в директории ограничен 32 битами [17].

Предполагается целиком помещать индексный дескриптор директории в саму директорию для ускорения чтения-записи при обращении к функции **readdir**. Также предполагается при динамическом выделении индексных дескрипторов использовать директории как контейнер для таблицы индексных дескрипторов. Для файлов, имеющих жесткие ссылки в других директориях, проблему решает счетчик числа связей в индексном дескрипторе.

Также разработчики предлагают хранить одно или несколько имен файлов в самом индексном дескрипторе в поле расширенных атрибутов файла. Имя должно хранить номер индексного дескриптора директории, к которой файл принадлежит [16].

Удаление файла из директории в англоязычной литературе носит название **unlink**, то есть удаление ссылки. На самом деле при удалении файла в предыдущей записи директории модифицируется поле «длина записи», которое может указывать на конец директории или на следующую запись. Сам индексный дескриптор и имя файла остаются в директории и могут быть прочитаны, что используется некоторыми программами по восстановлению файлов [17].

При чтении директорий с диска игнорируются ошибки ввода/вывода, чтобы у пользователей была возможность восстановить часть данных, если в директории есть сбойные сектора [18].

Формат дерева экстентов

Экстент представляет собой непрерывную последовательность блоков, идентифицируемую номером первого блока последовательности и длиной последовательности.

Один экстент может адресовать до 128 Мб. Для отображения файлов большего размера используется структура дерева. Экстенты хранятся в узлах дерева. Конечные узлы, содержащие указатели на сами экстенты, называются «листьями», остальные узлы, содержащие указатели на другие узлы дерева, называются «индексами» [16].

Каждый узел, лист или индекс, даже хранимый непосредственно в индексном дескрипторе, начинается с заголовка узла, представленного в табл. П.2.13. Формат экстента представлен в табл. П.2.14, а формат индекса экстента – в табл. П.2.15.

Таблица П.2.13

Формат заголовка узла

Размер поля, байт	**Поле**
2	Магическое число 0xF30A
2	Число входов
2	Максимальное число входов
2	Глубина дерева
4	Поколение дерева (пока не используется)

Таблица П.2.14

Формат экстента

Размер поля, байт	**Поле**
4	Первый логический блок экстента
2	Длина экстента
2	Старшие 16 битов номера физического блока
4	Младшие 32 бита номера физического блока

Таблица П.2.15

Формат индекса экстента

Размер	**Поле**

поля, байт	
4	Номер логического блока, с которого начинается индекс
4	Указатель на физический блок следующего уровня
2	Старшие 16 битов физического блока
2	Зарезервировано

Максимально возможное число блоков, которые может адресовать один экстент, составляет $2^{15} = 32768$, поскольку старший значащий бит в поле «длина экстента» используется для указания на неинициализированный (зарезервированный) экстент. Если это поле принимает максимальное значение, т. е. **0x8000**, то, несмотря на единичное значение старшего значащего бита, это инициализированный экстент, имеющий длину 32768. Таким образом, максимальная длина инициализированного экстента – 32768, а неинициализированного – 32767.

На рис. П.2.7 представлено дерево экстентов, состоящее из основных описанных выше структур [17].

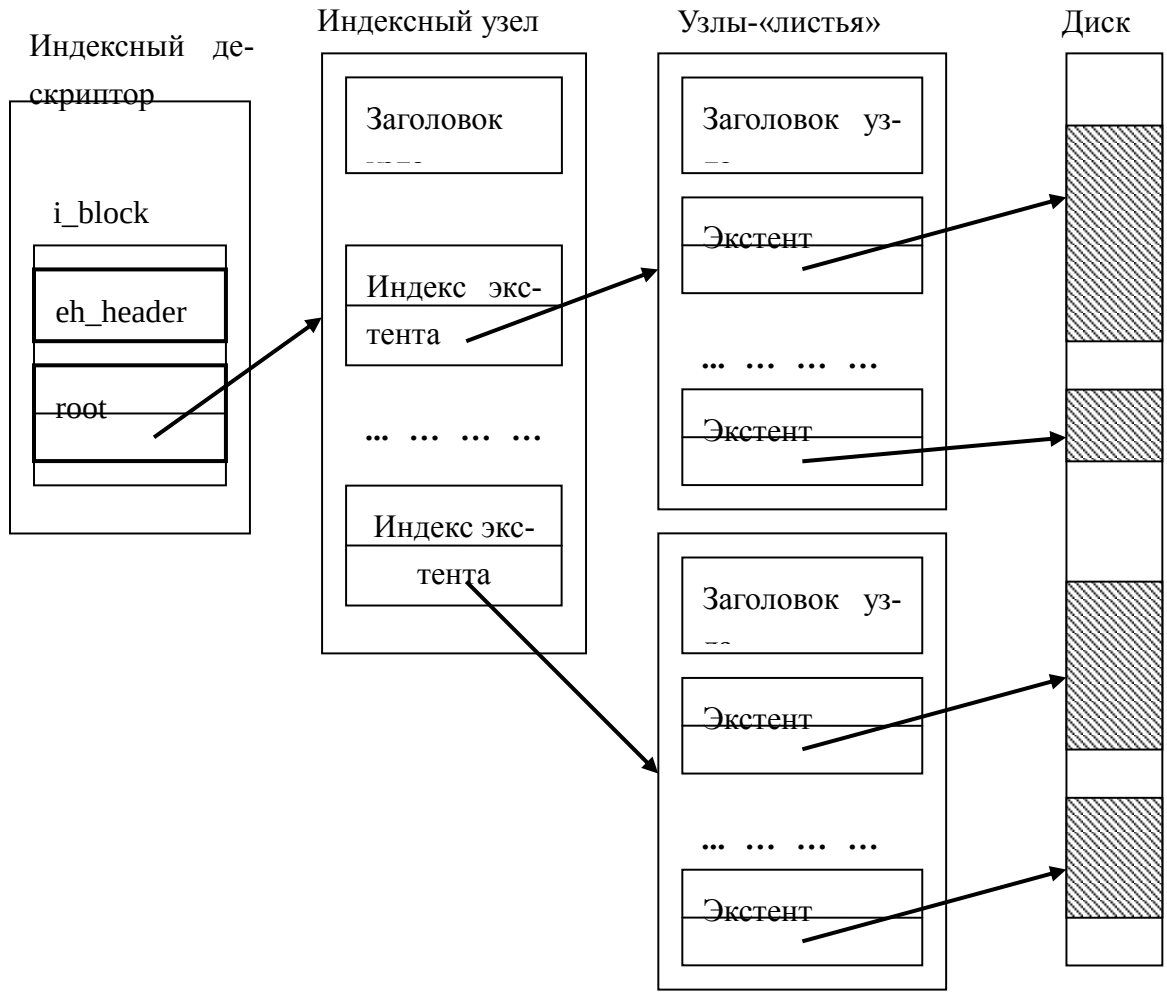

Рис. П.2.7. Структура дерева экстентов

После того как ФС смонтирована, все новые файлы создаются с картой экстен-

тов.

Карта экстентов – не слишком эффективный способ хранения разрозненных или сильно фрагментированных файлов. Разработчики собираются ввести новый тип экстента, с картой блоков. Другое магическое число в заголовке экстента будет определять новый тип «листа», содержащего список номеров выделенных блоков, так, как это делается сейчас в **ext3** [17]. Карта блоков по-прежнему доступна из **ext4**, если в индексном дескрипторе не выставлен флаг EXTENTS_FL.

Принципы выделения блоков и индексных дескрипторов

В **ext4** алгоритмы выделения блоков подчиняются принципам уменьшения фрагментации диска и уменьшению времени обращения к файлам.

Исходные коды ядра содержат разнообразные алгоритмы выделения (аллокации) блоков. За выделение блоков группами (множественное выделение блоков) отвечает так называемый аллокатор Орлова.

При выделении блоков для файла ФС учитывает номер группы, в которой находится данный индексный дескриптор. Таким образом, предпочтение отдается блокам той же группы, где находится дескриптор, а дочерние дескрипторы по возможности размещаются недалеко от родительского.

Если контрольная сумма группы блоков оказывается неверной, группа становится доступной только для чтения, и в ней запрещается выделение блоков и индексных дескрипторов.

Если группа блоков еще не инициализирована при выделении в ней индексного дескриптора, она помечается как инициализированная [15].

Выделение блоков

В индексном дескрипторе есть список предварительно зарезервированных блоков, в котором указывается начало зарезервированного пространства (логический и физический блоки), длина и доступное свободное место в этом зарезервированном пространстве.

При выделении блоков файлу аллокатор первым делом смотрит именно на этот список. Если файловая система не смогла выделить блоки из зарезервированного для индексного дескриптора пространства или если используется групповое резервирование, происходит поиск блоков в зарезервированном пространстве локальной группы.

Одним из новшеств **ext4** является множественное выделение блоков, что подразумевает поиск длинного непрерывного пространства на диске. Перед запросом на выделение блоков количество требуемых блоков нормализуется, т. е. выделяется больше блоков, чем файлу необходимо. Лишние блоки добавляются в список зарезервированных блоков для файла. Нормирование запроса происходит либо на основе данных о файле (если используется индивидуальное выделение блоков), либо, для группового

выделения, на основе постоянной величины в 512 блоков. Эта величина содержится в **/sys/fs/ext4/<partition/mb_group _prealloc** и может быть изменена.

При выделении блоков ищется группа с наиболее длинной последовательностью блоков и именно она выбирается. Для большей безопасности записи все остальные операции поиска экстента в других группах и в этой блокируются, пока система не завершит выделение [18].

Резервирование блоков

Некоторые приложения, например базы данных или потоковые медиапрограммы, нуждаются в выделении блоков для файла заранее, при этом не инициализируя блоков данными или нулями. Предварительное выделение гарантирует, что под файл будет выделено столько непрерывного дискового пространства, сколько возможно. Эта функция полезна, если приложению заранее известно, сколько пространства необходимо будет выделить под файл. Файловая система интерпретирует выделенные, но неинициализированные блоки как заполненные нулями. Предварительное выделение должно быть устойчиво к перезагрузкам [16].

Для приложений, использующих только последовательную запись в файл, довольно легко отделить инициализированную и неинициализированную части файла. Это можно сделать, добавив к файлу «водяной знак», определяющий размер инициализированной части файла. Однако это неэффективно для баз данных или других приложений, в которых запись в предварительно выделенные блоки происходит случайным образом в разные участки. ФС должна определять последовательность неинициализированных блоков данных в середине файла.

В **ext4** это решается использованием старшего значащего бита в поле длины экстента, который определяет, содержит ли экстент неинициализированные данные. При чтении неинициализированного экстента переключатель виртуальной файловой системы VFS возвращает блоки, заполненные нулями. При записи экстент может быть разбит на инициализированную и неинициализированную части, вторая из которых может быть слита со смежным экстентом, если пространство на диске непрерывно [16].

В **ext4** можно выделить два вида резервирования блоков: групповое и индивидуальное для конкретного индексного дескриптора. В зависимости от размера файла система решает, делать ли групповое или индивидуальное резервирование. Порог этот по умолчанию равен 16 блокам. Если файл займет на диске меньше 16 блоков, происходит групповое резервирование для того, чтобы маленькие файлы лежали на диске как можно ближе друг к другу. Пороговое значение может быть изменено в разделе **/sys/fs/ext4/<partition>/mb_stream_req**.

Дескриптор резервирования содержит информацию о типе резервирования (групповое или индивидуальное), список индексных дескрипторов или групп, начале и длине последовательности блоков.

Дескриптор резервирования активен, пока блок не будет помечен на битовой

карте как занятый. Дескриптор меняется только после того, как изменена битовая карта [18].

Выделение индексных дескрипторов

Если создаваемый индексный дескриптор относится к каталогу, система ищет группы блоков с большим свободным пространством и маленьким отношением количества каталогов к количеству обычных файлов. Если создаваемый индексный дескриптор относится к обычному файлу, ФС старается поместить его в ту же группу блоков, где находится родительский каталог [18].

Аллокатор Орлова для каталогов учитывает количество уже существующих директорий в группе блоков (их не должно быть слишком много), число свободных индексных дескрипторов (их не должно быть слишком мало), число свободных блоков (их также не должно быть слишком мало).

Дескрипторы каталогов стараются помещать в первую группу гибкой группы блоков, а дескрипторы обычных файлов – во все остальные (т. е. начиная со второй).

При удалении файла (последнего имени) индексный дескриптор очищается после того, как удалена запись в директории. Очистка индексного дескриптора происходит перед тем, как он помечается в битовой карте как свободный.

Функция множественного выделения блоков хранит последние N выделений в памяти. Их можно просмотреть в файле **`/proc/fs/ext4/<dev>/mb_history`**. В истории сохраняются данные о выделении и удалении блоков, о предварительном резервировании и снятии резервирования [18].

Журналирование

Структуры журнала описаны в заголовочном файле **`jbd2.h`**. Файл находится по адресу **`/usr/include/linux/jbd2.h`** или **`/usr/src/linux-headers-2.6.31-17/include/linux/jbd2.h`** (зависит от версии ядра системы). Журнал для файловой системы может быть и внешним, хранимым на отдельном блочном устройстве.

В **`ext3`** и **`ext4`** используется «физическое журналирование», т. е. в качестве основной единицы ведения журнала используется физический блок. Подход, когда хранятся только изменяемые байты, а не целые блоки, называется «логическим журналированием» (используется, например, XFS).

Драйвер файловой системы отслеживает обработку целых блоков данных и группирует их в отдельный объект, называемый транзакцией. Транзакция будет завершена только после записи на диск всех данных.

Транзакции являются групповыми операциями, выполняемыми или невыполняемыми как одна единая операция, т. е. атомарно. Журналируемые файловые системы (ext3, NTFS) в случае сбоя делают автоматический «откат» при следующей загрузке, и потери кластеров/блоков не происходит. Создание/удаление/переименование файла –

это атомарные операции, которые не могут допустить промежуточных состояний. А вот перемещение файла или запись на диск – более сложные операции. Поддержка транзакций не может застраховать от потери записываемых данных.

Журнал состоит из транзакций, имеющих непрерывно возрастающие номера. Как только достигнут конец журнала, запись начинается сначала. Таким образом, журнал – это файл с круговой записью. Если система была размонтирована без ошибок, то запись всегда начинается с самого начала [15].

Когда в файловой системе происходят какие-либо изменения, в журнал записываются целые блоки с метаданными, которые подлежат изменению (это может быть таблица индексных дескрипторов, суперблок и т.д.), затем эти блоки переписываются на жесткий диск и после этого в журнал записывается подтверждение.

Каждая транзакция состоит из одного и более дескрипторов, а также блоков данных. Последний дескриптор в транзакции – это блок подтверждения, который сигнализирует о том, что транзакция прошла успешно и данные из предыдущих дескрипторов были записаны.

Существует еще два вида дескрипторов: блоки отмены и блоки, содержащие тэги (иногда их называют просто дескрипторами). Блоки отмены заполняются номерами блоков, которые должны быть удалены из журнала во время этой транзакции. Дескриптор состоит из последовательности тэгов и идет перед блоками с метаданными.

Тэг – это структура, которая определяет последовательность журнальных блоков (не блоков файловой системы), данные с которых должны быть записаны в указанные блоки файловой системы [16]. Номер журнального блока, который должен быть записан в указанный блок файловой системы, определяется порядковым номером тэга в дескрипторе. Структура тэга определена в заголовочном файле **`jbd2.h`** как *journal_block_tag_s* и приведена в табл. П.2.16.

Таблица П.2.16

Структура тэга

Размер, байт	**Смещение**	**Назначение**
4	0 (0h)	Номер блока на диске
4	4 (4h)	Флаги
4	8 (8h)	Если включена поддержка 48-битной адресации, в тэг записываются старшие значащие биты номера блока

Первый блок журнала содержит «журнальный суперблок». Его структура определена в заголовочном файле **`jbd2.h`** как *journal_superblock_t* и приведена в таблице П.2.17.

Таблица П.2.17

Формат журнального суперблока

Размер, байт	**Смещение**	**Назначение**
12	0 (0h)	Заголовок (одинаков для всех дескрипторов)
4	12 (Ch)	Размер блоков журнального устройства. Блоки журнала отличаются от блоков файловой системы
4	16 (10h)	Количество блоков в журнальном устройстве
4	20 (14h)	Первый блок журнала, содержащий информацию
4	24 (18h)	Первый ID подтверждения, ожидаемый журналом
4	28 (1Ch)	Номер блока начала журнала
4	32 (20h)	Код ошибки (устанавливается функцией jbd2_journal_abort())
Следующие поля действительны только для журнального суперблока версии 2		
4	36 (24h)	Флаги compat
4	40 (28h)	Флаги incompat
4	44 (2Ch)	Флаги rocompat
16	48 (30h)	Журнальный UUID
4	64 (40h)	Число файловых систем, совместно использующих журнал
4	68 (44h)	Номер блока динамической копии суперблока журнала
4	72 (48h)	Максимальное число журнальных блоков на транзакцию
4	76 (4Ch)	Максимальное число блоков с данными на транзакцию
176	80 (50h)	Заполнение
16x48	256 (100h)	ID всех файловых систем, совместно использующих журнал

На рис. П.2.8 приведен фрагмент дескриптора, содержащего тэги. Номера физических блоков выделены подчеркиванием.

```
0x00094000  C0 3B 39 98 00 00 00 01 : 00 01 62 0D 00 00 04 BC  .;9.......b.....
0x00094010  00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00  ................
0x00094020  00 00 00 00 00 00 05 E6 : 00 00 00 02 00 00 04 A5  ................
0x00094030  00 00 00 02 00 00 18 6B : 00 00 00 02 00 00 04 BB  .......k........
0x00094040  00 00 00 02 00 00 0E 69 : 00 00 00 02 00 00 05 84  .......i........
0x00094050  00 00 00 02 00 08 02 02 : 00 00 00 02 00 08 00 23  ...............#
0x00094060  00 00 00 02 00 00 02 74 : 00 00 00 02 00 08 00 21  .......t.......!
0x00094070  00 00 00 02 00 00 03 F5 : 00 00 00 02 00 00 02 80  ................
```

Рис. П.2.8. Фрагмент журнального блока-дескриптора

Заголовок журнала имеет тот же вид, что и заголовки остальных дескрипторов. Заголовок определен в файле **`jbd2.h`** как структура *journal_header_s*. Формат заголовка дескриптора приведен в табл. П.2.18.

Таблица П.2.18

Формат заголовка журнального дескриптора

Размер, байт	**Смещение**	**Назначение**
4	0 (0h)	Магическое число 0xc03b3998 (первые 4 байта /dev/random)
4	4 (4h)	Тип дескриптора: 1 — дескриптор-тэг или просто дескриптор; 2 — подтверждение; 3 — суперблок версия 1; 4 — суперблок версия 2; 5 — блок отмены
4	8 (8h)	Номер последовательности

Существует несколько режимов журналирования для файловых систем **`ext3/ext4`**. Режим работы журнала определяется на этапе монтирования.

1. Journal. В журнал записываются как метаданные, так и содержимое файла до того, как их запись будет подтверждена в основной файловой системе. В этом случае риск потери данных даже при пропадании питания минимален. Однако в большинстве случаев такой режим сильно снижает производительность системы, так как данные необходимо записывать дважды: в основную файловую систему и в журнал.

2. Ordered. В этом режиме журналируются только метаданные, но не содержимое файлов. Однако ФС гарантирует, что содержимое файла будет записано на диск до того, как метаданные, относящиеся к этой транзакции, будут помечены как подтвержденные. Этот режим используется по умолчанию в большинстве дистрибутивов Linux. Если происходит пропадание питания или критическая ошибка ядра во время записи файла, журнал при следующем включении находит новый файл, создание которого не было подтверждено, и запускает процесс очистки, т.е. может записать в журнал подтверждение. Однако бывают ситуации, когда файл перезаписан поверх другой информацией. В этом случае может быть восстановлено промежуточное состояние файла. Худший из случаев – если в восстановленном файле будут перемешаны старые и новые данные.

3. Writeback. Журналируются только метаданные. Содержимое файла может быть записано на диск как до, так и после того, как обновляется журнал. В результате файлы, записанные непосредственно перед крахом системы, могут оказаться поврежденными. Например, файл, присоединенный к другому, может иметь в журнале большую длину, чем есть на самом деле, таким образом, конец файла забивается мусором. Также после восстановления журнала могут неожиданно появиться старые версии файлов. Однако во многих случаях разрыв в синхронизации между журналом и данными

невелик. Например, JFS по умолчанию использует именно этот режим.

В **Ext3** не предусмотрено контрольного суммирования блоков журнала. Недостаток такого метода в том, что если при пропадании питания будут повреждены какие-то блоки журнала, но записан блок подтверждения, система при следующей загрузке посчитает, что журнал не содержит ошибок, и проведет неверные транзакции. В **ext4** при несовпадении контрольных сумм журнала транзакция не состоится, что предотвращает запись поврежденных данных на диск [17].

Контрольные суммы транзакций хранятся в блоке подтверждения. Тип контрольной суммы определяется заголовком этого блока и может принимать значения: CRC32, MD5, SHA1.

Журнал необходим файловой системе для сохранения целостности метаданных, в том числе даже после сбоя системы, но не для восстановления целостности данных пользователя, например данных, по неосторожности удаленных с диска. Существуют программные продукты, которые могут восстанавливать файлы, анализируя журнал, иногда с помощью довольно сложных эвристических алгоритмов, однако полной гарантии восстановления они не дают [18].

ПРИЛОЖЕНИЕ 3

СПРАВКА ОБ ОТЛАДЧИКЕ DEBUGFS

DebugFS является самой известной утилитой, предназначенной для работы с файловыми системами **ext2fs** и **ext3fs**. DebugFS является программой интерактивного режима, и основные ее команды реализуются только «внутри» отладчика. Больших удобств в плане наглядности такой режим не предоставляет, тем более что пользователь при многократном повторении одних и тех же команд лишен возможности использовать память командной строки, причем вывод информации в файл реализуется только для отдельных команд.

Отладчик может ограниченно работать с одиночными командами или заранее подготовленным пакетным файлом. Но с большинством «внутренних» команд приходится использовать номер **inode** в угловых скобках **< >**, которые командный интерпретатор воспринимает как команды перенаправления ввода/вывода.

В режиме ввода одиночных команд отладчик можно использовать только для одного практического случая:

```
debugfs –R stats <dev>
```

выводит достаточно полную информацию о суперблоке и обо всех группах блоков. Фрагмент, выведенный такой командой, был показан на рис. 4.6.

Вход в интерактивный режим отладчика производится командой **debugfs** без аргументов. Однако ничего полезного при этом сделать еще нельзя. На попытку ввода любой команды отладчик ответит, что файловая система еще не открыта. Для ее открытия следует задать следующую команду:

```
open –w <dev>
```

Вместо **<dev>** указывается логический раздел блочного устройства. Можно сразу воспользоваться командой

```
debugfs –w <dev>
```

Атрибут **–w** указывается, если нужен доступ к файловой системе в режиме чтения и записи. Все команды, модифицирующие, устанавливающие или удаляющие что-либо, нуждаются в таком режиме. Нелишне предупредить, что ошибка в режиме записи может привести к нежелательным последствиям.

После открытия логического раздела или физического устройства система может известить пользователя об успехе или просто предложить ввести следующую команду. Всего в арсенале отладчика несколько десятков команд, но администратору могут понадобиться только некоторые из них. Если есть необходимость в использовании других команд, следует познакомиться с электронным справочником **man**. Рассмотрим основные ко-

манды (они сгруппированы по выполняемым функциям или последовательности применения):

clri <file_name> – очистить индексный дескриптор указанного файла,

freeb block_number – команда устанавливает в «0» бит, соответствующий данному логическому блоку в битовой карте блоков, тем самым объявляя его свободным,

setb block_number – противоположная по смыслу команда устанавливает соответствующий бит в «1», объявляя блок занятым. Если эти установки произведены необдуманно, утилита **fsck**, будучи автоматически запущена при очередной загрузке системы, найдет эти блоки и поместит их в каталог **/lost+found**,

freei <inode_number> – команда устанавливает в «0» бит указанного индексного дескриптора в битовой карте **inode**, объявляя его свободным,

setb <inode_number> – команда устанавливает соответствующий бит в «1», объявляя блок **inode** занятым,

icheck block_number – команда, позволяющая по номеру логического блока данных найти его индексный дескриптор. Требует от нескольких десятков секунд до нескольких минут поиска на диске. Не всегда может найти **inode** логически удаленного файла,

ncheck inode_number – команда, позволяющая узнать имя файла по заданному номеру индексного дескриптора. Также нуждается в непродолжительном времени поиска,

stats – эта команда уже приводилась выше. В интерактивном режиме она выводит информацию о суперблоке и всех группах блоков на анализируемом дисковом пространстве,

stat <inode_number> – команда позволяет вывести краткую справку о содержимом заданного индексного дескриптора,

lsdel – команда не нуждается в дополнительных атрибутах и выводит перечень удаленных **inode**, которые исследователю придется запомнить или записать. Для вывода информации в файл рекомендуется использовать конструкцию с неименованным каналом:

lsdel | debugfs /dev/hdc3 > /home/file_lsdel,

mi <inode_number> – команда, позволяющая модифицировать каждую запись в указанном индексном дескрипторе. Строки выводятся поочередно: **Mode**, **UID**, **GID**, **Size**, 4 временные отметки и так далее. Слева от курсора выводится действующее значение, в позицию курсора пользователь может ввести, что ему требуется. Пример использования команды:

```
debugfs:  mi <148003>
Mode  [0100644]
User ID  [503]
Group ID  [100]
Size  [6065]
Creation time  [833201524]
Modification time  [832708049]
Access time  [826012887]
Deletion time  [833201524]
Link count  [0] 1
Block count  [12]
File flags  [0x0]
Reserved1  [0]
File acl  [0]
Directory acl  [0]
Fragment address  [0]
Fragment number  [0]
Fragment size  [0]
Direct Block #0  [594810]
Direct Block #1  [594811]
Direct Block #2  [594814]
Direct Block #3  [594815]
Direct Block #4  [594816]
Direct Block #5  [594817
Direct Block #6  [0]
Direct Block #7  [0]
Direct Block #8  [0]
Direct Block #9  [0]
Direct Block #10  [0]
Direct Block #11  [0]
Indirect Block  [0]
Double Indirect Block  [0]
Triple Indirect Block  [0]
```

help – вывод справки о командах отладчика,

close – закрыть файловую систему, открытую командой **open**,

quit – выйти из отладчика в режим командной строки стандартной оболочки.

ПРИЛОЖЕНИЕ 4

СТРУКТУРА И ИСХОДНЫЙ КОД ПРОГРАММЫ EXTVIEW

Дисковый редактор **lde** и отладчик ФС **debugfs** долго и успешно использовались для исследования и ремонта файловой системы **ext2fs**. Но они давно не обновлялись, и для работы с третьей и четвертой системами фактически непригодны.

Lde некорректно выводит информацию об индексных дескрипторах, начиная с файловой системы **ext3**. **Debugfs** не выводит списка удаленных индексных дескрипторов, хотя при восстановлении файлов это может сыграть существенную роль. Кроме того, **debugfs** требует почти во всех случаях открытия файловой системы. Ни одна из этих программ не работает с экстентами и не может отображать информацию из журнала файловой системы.

На основании полученных в результате исследования данных о файловой системе **ext4**, Желтышевой Екатериной Дмитриевной разработана альтернативная программа для просмотра и изучения объектов этой ФС под названием **extview** (то есть обозреватель файловой системы ext). Она работает на всех файловых системах ext2/3/4, просматривает как экстенты, так и блоки данных и имеет достаточную функциональность для отображения объектов системы. При этом программа не изменяет данные на диске и не требует монтирования рассматриваемой файловой системы.

Для разработки программы был выбран язык программирования C/C++, поскольку этот язык считается «родным» для Linux, а в операционной системе имеется встроенный компилятор **gcc**.

Для удобства чтения кода и написания программы исходный код **extview** разделен на несколько модулей, каждый из которых выполняет свои функции независимо от других. Благодаря этому программа легко расширяема: достаточно добавить в систему новый модуль.

Сбор отдельных частей программы в единое целое осуществляется средствами автосборки ОС Linux, а именно утилитой *make*.

Разработанная программа открывает непосредственно файл устройства в режиме «только для чтения». Это позволяет ей работать как при загрузке с внешнего носителя, так и на работающей системе. Система не обязательно должна быть смонтирована, а также не обязательно может являться блочным устройством, это может быть и файл-образ файловой системы.

Программа может выводить информацию на двух языках: русском, если используется языковая локаль с кодировкой UTF-8, и английском во всех остальных случаях.

Характеристика модулей программы

Модули программы и связи между ними представлены на рис. П.4.1.

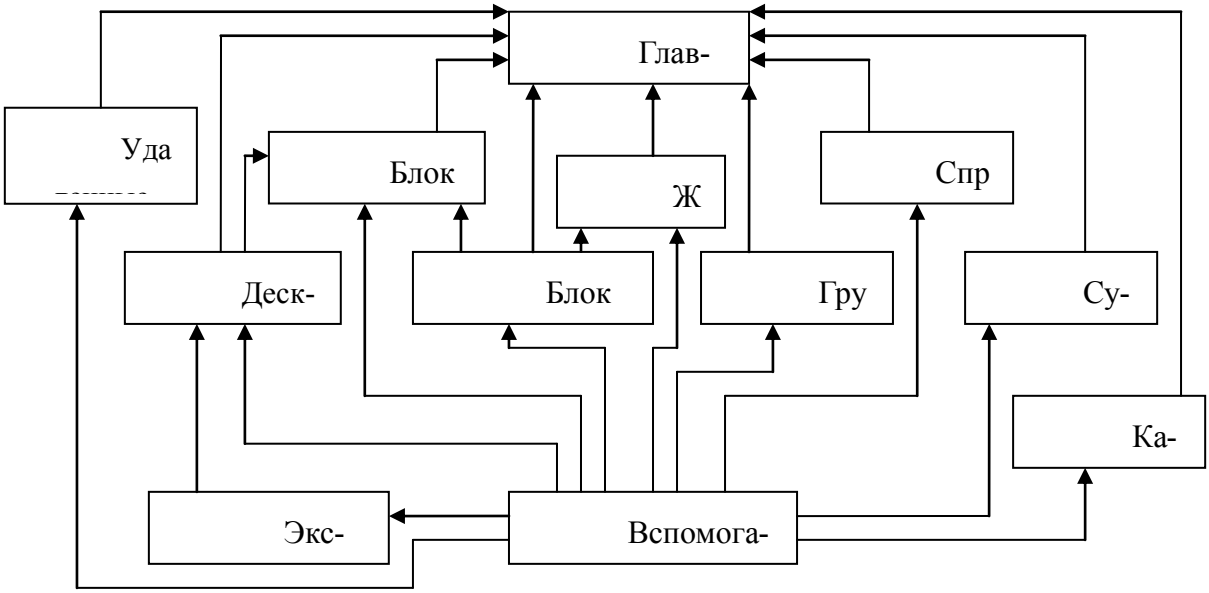

Рис. П.4.1. Состав и связи модулей программы

Main (главный) включает в себя все остальные модули. Этот модуль ведет обработку информации, поступающей из командной строки, частично обрабатывает ошибки и решает, какому из модулей передать полученную информацию.

Модуль *option* относится к вспомогательным модулям. В нем описаны функции, необходимые для работы всех остальных модулей. Это чтение с диска определенного количества байтов, определение размера блока файловой системы, преобразование в шестнадцатеричные или бинарные числа и чтение языковой переменной окружения. В перспективе программа сможет поддерживать несколько языков и кодировок.

К вспомогательным модулям, которые отсутствуют в исходном коде main.c, относится также модуль *exttree* (дерево экстентов). Он входит в состав модуля *inodeinfo* (индексный дескриптор) и отвечает за вывод на экран дерева экстентов, если адресация экстентами используется в файле. Модуль состоит всего из одной рекурсивной функции. Функция ищет в блоке данных магическое число экстентов, определяет глубину дерева, считывает и выводит блоки данных, в том числе индексы экстентов.

Модуль *get_block* (блок данных) выводит на экран определенный

блок файловой системы в шестнадцатеричном виде. За основу берется интерфейс дискового редактора *lde*. В данном случае по функционалу *get_block* ничем не отличается от *lde*, этот модуль нужен для удобства пользования программой.

Модуль *group* (группы) выводит информацию о группах блоков в файловой системе. За основу взят интерфейс редактора *debugfs*, в котором отдельной такой команды не было, информация о группах блоков выводилась наряду с информацией о файловой системе командой **`debugfs –R stats <device>.`**

Модуль *inode_block* (блок индексного дескриптора) выводит на экран индексный дескриптор в шестнадцатеричном виде с указанием, в каком блоке и по какому смещению находится запрашиваемый дескриптор. Это нужно для проверки действий программы или для вывода полей, которые в программе не отображаются.

Модуль *inodeinfo* (индексный дескриптор) используется для чтения и вывода на экран информации об указанном индексном дескрипторе. За основу взят интерфейс программы *lde*. Кроме того, в модуле *inodeinfo* содержится несколько новых функций, отсутствующих в программе *lde*: например, вывод экстентов или расширенных атрибутов файла.

Модуль *super* (суперблок) выводит информацию о файловой системе, анализируя суперблок. За основу взят интерфейс встроенного отладчика файловой системы debugfs, вывод команды **`debugfs –R stats <device>`**.

Модуль *direct* (каталоги) выводит элементы директории, указанной по индексному дескриптору, в том числе удаленные и скрытые файлы. В сочетании с опцией «B» интерпретирует входной аргумент как номер блока и пытается вывести элементы директорий из этого блока. Эта опция заложена на будущее, если придется искать имя файла в блоках данных (такая возможность в процессе написания программы рассматривалась), тогда найденный блок можно попытаться интерпретировать как директорию, даже если ее индексный дескриптор неизвестен.

Модуль *jour* (журнал) позволяет просматривать журнальные блоки, если в файловой системе есть внутренний журнал. Позволяет вывести информацию о блоке журнала, а при указании опции «B» еще и сам журнальный блок.

Модуль *lost* (удаленные файлы) используется для поиска удаленных файлов в журнале и на диске. Если указано только имя устройства, выводит список удаленных индексных дескрипторов, перед этим запрашивая время, по которому необходимо отсортировать удаленные дескрипторы. Дескрипторы, удаление которых произошло до указанного времени, на эк-

ран не выводятся.

Модуль *help* выводит справку о программе на русском или английском языке, в зависимости от настроек языковой локализации.

Make-файл

```
extview: main.o inodeinfo.o option.o exttree.o super.o group.o get_block.o inode_block.o helping.o jour.o lost.o direct.o context.o
	gcc -o extview main.o inodeinfo.o option.o exttree.o super.o group.o get_block.o inode_block.o helping.o jour.o lost.o direct.o context.o

helping.o: helping.c helping.h
	gcc -c helping.c

exttree.o: exttree.c exttree.h
	gcc -c exttree.c

option.o: option.c option.h
	gcc -c option.c

inodeinfo.o: inodeinfo.c inodeinfo.h
	gcc -c inodeinfo.c

super.o: super.c super.h
	gcc -c super.c

group.o: group.c group.h
	gcc -c group.c

get_block.o: get_block.c get_block.h
	gcc -c get_block.c

inode_block.o: inode_block.c inode_block.h
	gcc -c inode_block.c
```

```
jour.o: jour.c jour.h
 gcc -c jour.c

lost.o: lost.c lost.h
 gcc -c lost.c

direct.o: direct.c direct.h
 gcc -c direct.c

context.o: context.c context.h
 gcc -c context.c

main.o: main.c
 gcc -c main.c

clean:
 rm -f *.o
 rm -f extview
```

Модуль main

Файл main.c

```
#include <stdio.h>
#include <getopt.h>
#include "inodeinfo.h"
#include "super.h"
#include "group.h"
#include "get_block.h"
#include "inode_block.h"
#include "helping.h"
#include "jour.h"
#include "lost.h"
#include "direct.h"
#include "context.h"
void main(int argc,char *argv[])
{
int opt,type1;
unsigned int block,inode;
FILE *disk;
char diskname[80];
char *file=NULL;
int n;
char entr,entr2;
opt=getopt(argc,argv,"i:sgld:cb:j:h-");
switch (opt)
{
      case 'i':
      inode=atoi(optarg);
      if (argc<4) printf("Not enough parameters\n");
      else
      {
            sscanf(argv[3],"%s",&diskname);
            disk=fopen(diskname,"r");
            if (disk!=0)
            {
                  inodeinfo(inode,disk);
                  fclose (disk);
            }
            else printf("Can not open file %s\n",diskname);
```

```
}
break;

case 's':
if (argc<3) printf("Not enough parameters\n");
else
{
      sscanf(argv[2],"%s",&diskname);
      disk=fopen(diskname,"r");
      if (disk==0) printf("Can not open file %s\n",diskname);
      else {super(disk); fclose(disk);}
}
break;

case 'g':
if (argc<3) printf("Not enough parameters\n");
else
{
      sscanf(argv[2],"%s",&diskname);
      disk=fopen(diskname,"r");
      if (disk==0) printf("Can not open file %s\n",diskname);
      else {group(disk); fclose(disk);}
}
break;

case 'l':
if (argc<3) printf("Not enough parameters\n");
if (argc==3)
{
      type1=3;
      sscanf(argv[2],"%s",&diskname);
      inode=0;
      disk=fopen(diskname,"r");
      if (disk==0) printf("Can not open file %s\n",diskname);
      else {lost(disk,inode,type1); fclose(disk);}
}
if (argc==4)
{
      type1=1;
      sscanf(argv[2],"%d",&inode);
```

```
      sscanf(argv[3],"%s",&diskname);
      disk=fopen(diskname,"r");
      if (disk==0) printf("Can not open file %s\n",diskname);
      else {lost(disk,inode,type1); fclose(disk);}
}
if (argc>=5)
{
      sscanf(argv[2],"%c",&entr);
      if (entr=='i') type1=1;
      if (entr=='b') type1=2;
      if (entr!='i'&&entr!='b') type1=1;
      sscanf(argv[3],"%d",&inode);
      sscanf(argv[4],"%s",&diskname);
      disk=fopen(diskname,"r");
      if (disk==0) printf("Can not open file %s\n",diskname);
      else {lost(disk,inode,type1); fclose(disk);}
}
break;

case 'd':
inode=atoi(optarg);
if (argc<4) printf("Not enough parameters\n");
else
{
      sscanf(argv[3],"%s",&diskname);
      if(argc>4) sscanf(argv[4],"%c",&entr);
      if (entr=='B') type1=1;
      else if (entr=='F') type1=2;
      else type1=0;
      if (argc>5) sscanf(argv[5],"%c",&entr2);
      if (entr2=='F'&&entr=='B') type1=3;
      disk=fopen(diskname,"r");
      if (disk==0) printf("Can not open file %s\n",diskname);
      else {direct(inode,disk,type1); fclose(disk);}
}
break;

case 'c':
if (argc<3) printf("Not enough parameters\n");
else
```

```
{
      sscanf(argv[2],"%s",&diskname);
      disk=fopen(diskname,"r");
      if (disk==0) printf("Can not open file %s\n",diskname);
      else {context(disk); fclose(disk);}
}
break;

case 'b':
block=atoi(optarg);
if (argc<4) printf("Not enough parameters\n");
else
{
      sscanf(argv[3],"%s",&diskname);
      disk=fopen(diskname,"r");
      if (disk==0) printf("Can not open file %s\n",diskname);
      else {get_block(block,disk); fclose(disk);}
}
break;

case 'j':
block=atoi(optarg);
if (argc<4) printf("Not enough parameters\n");
else
{
      sscanf(argv[3],"%s",&diskname);
      disk=fopen(diskname,"r");
      int j_type;
      char arg4=0;
      if (argc==5) sscanf(argv[4],"%c",&arg4);
      if (arg4=='B') j_type=1;
      else j_type=0;
      if (disk==0) printf("Can not open file %s\n",diskname);
      else {jour(block,disk,j_type); fclose(disk);}
}
break;

case 'h':
helping();
break;
```

```
      case '-':
      break;
      case '?':
      printf("Unknown argument\n");
      break;
}
const struct option long_opts[] = {
      { "inodeblock", required_argument, NULL, 'a'},
      { "help", no_argument, NULL, 'h'},
      { NULL, 0, NULL, 0}
};
opt=getopt_long(argc,argv,"a:h",long_opts,NULL);
if (opt=='a')
{
      inode=atoi(optarg);
      if (argc<4) printf("Not enough parameters\n");
      else
      {
            sscanf(argv[3],"%s",&diskname);
            disk=fopen(diskname,"r");
            if (disk==0) printf("Can not open file %s\n",diskname);
            else {inode_block(disk,inode); fclose(disk);}
      }
}
if (opt=='h')
      helping();
}
```

Модуль option

Файл option.h

```
int power(int p1,int p2);
int read_2(FILE *f, unsigned int block_number,int
block_size,\ int block_offset);
unsigned int read_4(FILE *f,unsigned int block_number, \
int block_size,int block_offset);
int get_block_size(FILE *disk);
void hexing(unsigned int decimal,char hex[8],int del[8]);
void binaring(int number, char binar[4]);
int get_local();
```

Файл option.c

```
#include <stdio.h>
#include "option.h"
int power(int p1,int p2){
  int i,result;
  result=p1;
  for (i=0;i<(p2-1);i++)
    result=result*p1;
  if (p2==0) result=1;
  return (result);
}
//Функция считывает с диска 2 байта из указанного номера
блока
//по указанному смещению
int read_2(FILE *f,unsigned int block_number,int
block_size,int block_offset) {
  int g1,g2,i;
  int result;
  int k=0;
  fseek(f,0,SEEK_SET);
  //для быстроты чтения с диска читаем большими блоками
  for (i=0;i<8192;i++)
  {
    if (block_number>524287) block_number=block_number-
524287;
    else {k=i; break;}
  }
```

```
  for (i=0;i<k;i++)
    fseek(f,524287*block_size,SEEK_CUR);
  fseek(f,block_number*block_size,SEEK_CUR);
  fseek(f,block_offset,SEEK_CUR);
  g1=fgetc(f);
  g2=fgetc(f);
  result=g1+g2*256;
  return(result);
}
//Функция считывает 4 байта с диска из блока с указанным
номером
//по указанному смещению
unsigned int read_4(FILE *f,unsigned int block_number,int
block_size,int block_offset)
{
  int g1,g2,g3,g4,i,k;
  int result;
  fseek(f,0,SEEK_SET);
  k=0;
  for (i=0;i<8192;i++)
  {
    if   (block_number>524287)   block_number=block_number-
524287;
    else {k=i; break;}
  }
  for (i=0;i<k;i++)
    fseek(f,524287*block_size,SEEK_CUR);
  fseek(f,block_number*block_size,SEEK_CUR);
  fseek(f,block_offset,SEEK_CUR);
  g1=fgetc(f);
  g2=fgetc(f);
  g3=fgetc(f);
  g4=fgetc(f);
  result=g1+g2*256+g3*256*256+g4*256*256*256;
  return(result);
}
int get_block_size(FILE *disk)
{
  int block_indic,block_size;
  block_indic=read_4(disk,1,1024,24); //1024+24
```

```
    switch(block_indic)
    {
      case 0: block_size=1024; break;
      case 1: block_size=2048; break;
      case 2: block_size=4096; break;
      default: block_size=0;
    }
    return(block_size);
  }
  //Функция записывает шестнадцатеричное представление числа
в //строку и в числовой массив
  void hexing(unsigned int decimal,char hex[8],int del[8]) {
    int i,step;
    for (i=0;i<8;i++){
      step=power(16,8-i-1);
      if (decimal>=step){
        del[i]=decimal/step;
        switch(del[i]){
          case 1: hex[i]='1'; break;
          case 2: hex[i]='2'; break;
          case 3: hex[i]='3'; break;
          case 4: hex[i]='4'; break;
          case 5: hex[i]='5'; break;
          case 6: hex[i]='6'; break;
          case 7: hex[i]='7'; break;
          case 8: hex[i]='8'; break;
          case 9: hex[i]='9'; break;
          case 10: hex[i]='A'; break;
          case 11: hex[i]='B'; break;
          case 12: hex[i]='C'; break;
          case 13: hex[i]='D'; break;
          case 14: hex[i]='E'; break;
          case 15: hex[i]='F'; break;
          default: hex[i]='?';
        }
        decimal=decimal-(del[i]*step);
      }
      else {hex[i]='0'; del[i]=0;}
    }
  }
```

```
    //Функция записывает в символьный массив двоичное пред-
ставление переданного ей числа
    void binaring(int number, char binar[4]) {
      int i,step;
      for (i=0;i<4;i++) {
        step=power(2,3-i);
        if (number>=step) {
          binar[i]='1';
          number=number-step;
        }
        else binar[i]='0';
      }
    }
    //Функция возвращает 1, если используется русская кодиров-
ка UTF-8 и 0, если другие кодировки
    int get_local()
    {
      char *value;
      value=(char*)getenv("LANG");
      if (strcmp("ru_RU.UTF-8",value)==0) return(1);
      else return(0);
    }
```

Модуль exttree

Файл exttree.h

```
int exttree(FILE *disk,unsigned int offsets[2],\
int block_size,int depth);
```

Файл exttree.c

```
#include <stdio.h>
#include "option.h"
#include "exttree.h"
int exttree(FILE *disk,unsigned int offsets[2],int
block_size,int dep)
{
  int magic,magic_set,depth,entries;
  int i,j,ext_len;
  unsigned int ext_block,next_block,next_offsets[2];
  char hex[8];
  int del[8];
  int loc=get_local();
  magic=read_2(disk,offsets[0],block_size,offsets[1]+40);
  if (magic==62218) magic_set=40+offsets[1];
  else
  {
    magic=read_2(disk,offsets[0],block_size,offsets[1]);
    if (magic==62218) magic_set=0+offsets[1];
    else {
      { if (loc==1)
        printf("Этот узел не содержит магического чис-
ла\n");
        else printf("This node does not contain magic num-
ber");}
        return(1);}
  }
  depth=read_2(disk,offsets[0],block_size,magic_set+6);
  entries=read_2(disk,offsets[0],block_size,magic_set+2);
  if (depth==0)
  {
    for (i=0;i<entries;i++)
    {
    ext_block = read_4 (disk, offsets[0], block_size, \\
```

```
magic_set+20+i*12);
        ext_len = read_2 (disk, offsets[0], block_size, \\
magic_set+16+i*12);
          for (j=0;j<dep;j++) printf("\t");
          if (loc==1) printf("%d-й экстент: первый блок: %d \\
(0x",i+1,ext_block);
          else printf("Extnent %d: first block: %d \\
(0x",i+1,ext_block);
          hexing(ext_block,hex,del);
          printf("%s",hex);
          if (loc==1) printf("), длина: %d\n",ext_len);
          else printf("), length: %d\n",ext_len);
        }
      }
      else
      {
        for (i=0;i<entries;i++)
        {
          next_block = read_4 (disk, offsets[0], block_size,
\\ magic_set+16+i*12);
          hexing(next_block,hex,del);
          for (j=0;j<dep;j++) printf("\t");
          if (loc==1) printf("Следующий узел дерева в блоке:
%d \\ (0x%s)\n",next_block,hex);
          else printf("Next node in block: %d (0x%s)\n", \\
next_block,hex);
          next_offsets[0]=next_block;
          next_offsets[1]=0;
          exttree(disk,next_offsets,block_size,dep+1);
        }
      }
    }
```

Модуль get_block

Файл get_block.h

```
int get_block(unsigned int block_number, FILE *disk);
```

Файл get_block.c

```
#include <stdio.h>
#include "option.h"
#include "get_block.h"
/*Вместо целого значения функция выводит восьмеричное
 *представление числа
 */
void hexing_pos (int sym,char pos[2])
{
  int del[8];
  char hex[8];
  hexing(sym,hex,del);
  pos[0]=hex[6];
  pos[1]=hex[7];
}
/*Функция считывает подряд 8 байт от текущей позиции кур-
сора в
 *файле и выводит их на экран
 */
void get8(FILE *disk,char points[9])
{
  int sym,j;
  char pos[2];
  for (j=0;j<8;j++)
  {
    sym=fgetc(disk);
    hexing_pos(sym,pos);
    printf(" ");
    printf("%c",pos[0]);
    printf("%c",pos[1]);
    points[j]=sym;
    if (sym<32||sym>126) points[j]='.';
  }
  points[8]='\0';
}
int get_block(unsigned int block_number, FILE *disk)
```

```
    {
      int magic,block_size,loc;
      unsigned int block_pos,block_count;
      int i,j,k=0,str_count,del[8],sym;
      char hex_pos[8],pos[2],points1[9],points2[9];
      magic=read_2(disk,1,1024,56);
      loc=get_local();
      if (magic!=61267)
      {
        if (loc==1)
          printf("Файл устройства не содержит магического чис-
ла\n");
        else printf("File does not contain magic number\n");
        return(1);
      }
      block_size=get_block_size(disk);
      str_count=block_size/16;          /*Вычисляем       число
строк в
                                        *блоке  (строки  по  16
                                  байт)
                                        */
      block_pos=block_number*block_size;    /*Позиция   курсо-
ра, нужна
                                        *для вывода левой
                                        *колонки */
     block_count=block_number;          /*Вспомогательная   пе-
ременная
                                   *для установки курсора в
                                   *начало блока */
      //Устанавливаем курсор в начало блока
      fseek(disk,0,SEEK_SET);
      for (i=0;i<8192;i++)
      {
        if    (block_count>524287)    block_count=block_count-
524287;
        else {k=i; break;}
      }
      for (i=0;i<k;i++)
        fseek(disk,524287*block_size,SEEK_CUR);
      fseek(disk,block_count*block_size,SEEK_CUR);
```

```
  //Считываем байты и выводим на экран
  printf("\n");
  for (i=0;i<str_count;i++)
  {
    hexing(block_pos,hex_pos,del);
    printf("0x%s ",hex_pos);
    get8(disk,points1);
    printf(" :");
    get8(disk,points2);
    printf("  %s%s",points1,points2);
    printf("\n");
    block_pos=block_pos+16;
  }
  printf("\n");
  return (0);
}
```

Модуль group

Файл group.h

```
int group(FILE *disk);
```

Файл group.c

```
#include <stdio.h>
#include "option.h"
#include "group.h"
int group(FILE *disk)
{
  int magic,block_size,block_count,blopergro;
  unsigned int gr_number,gr_block,gr_count;
  unsigned int block_bitmap, inode_bitmap, inode_table,
  unsigned int free_blocks, free_inodes, directories,
  unsigned int unused_inodes, check_sum;
  char hex_check[8];
  int del[8];
  int loc=get_local();
  int i,j;
  int desc_size=32;
  magic=read_2(disk,1,1024,56);
  if (magic!=61267)
  {
    if (loc==1)
      printf("Файл устройства не содержит магического чис-
ла\n");
    else printf("File does not contain magic number\n");
    return(1);
  }
  block_size=get_block_size(disk);
  if (block_size==1024) gr_block=2;
  else gr_block=1;
  block_count=read_4(disk,1,1024,4);
  blopergro=read_4(disk,1,1024,32);
  gr_count=block_count/blopergro+1;
  if (loc==1) printf("Число групп блоков: %d\n",gr_count);
  else printf("Block group count: %d\n",gr_count);
  for (i=0;i<gr_count;i++)
  {
    if (loc==1) printf("Группа %d:\t",i);
```

```
        else printf("Group %d:\t",i);

block_bitmap=read_4(disk,gr_block,block_size,i*desc_size);
        if (loc==1) printf("карта блоков: %d, ",block_bitmap);
        else printf("block bitmap at: %d, ",block_bitmap);

inode_bitmap=read_4(disk,gr_block,block_size,4+i*desc_size);
        if (loc==1)
          printf("карта дескрипторов: %d,\n",inode_bitmap);
        else printf("inode bitmap at: %d,\n",inode_bitmap);

inode_table=read_4(disk,gr_block,block_size,8+i*desc_size);
        if (loc==1)
          printf("\t\tтаблица   индексных   дескрипторов:   \\
%d,\n",inode_table);
        else printf("\t\tinode table at: %d,\n",inode_table);

free_blocks=read_2(disk,gr_block,block_size,12+i*desc_size);
        if (loc==1)
          printf("\t\tсвободных блоков: %d, ",free_blocks);
        else printf("\t\tfree block count: %d, ",free_blocks);

free_inodes=read_2(disk,gr_block,block_size,14+i*desc_size);
        if (loc==1)
          printf("свободных дескрипторов: %d,\n",free_inodes);
        else printf("free inodes: %d,\n",free_inodes);
        directo-
ries=read_2(disk,gr_block,block_size,16+i*desc_size);
        if      (loc==1)      printf("\t\tдиректорий:      %d,
",directories);
        else printf("\t\tdirectories: %d, ",directories);
        unused_inodes = read_2 (disk, gr_block, block_size, \\
28+i*desc_size);
        if (loc==1)
          printf("неиспользованных      дескрипторов:      \\
%d.\n",unused_inodes);
        else printf("unused inodes: %d.\n",unused_inodes);

check_sum=read_2(disk,gr_block,block_size,30+i*desc_size);
        hexing(check_sum,hex_check,del);
```

```
    if (loc==1) printf("\t\tКонтрольная сумма: 0x");
    else printf("\t\tCheck sum: 0x");
    for (j=0;j<4;j++) printf("%c",hex_check[j+4]);
    printf("\n\n");
  }
  return (0);
}
```

Модуль inodeinfo

Файл inodeinfo.h

```
int inodeinfo(unsigned int inode, FILE *disk);
```

Файл inodeinfo.c

```
#include <stdio.h>
#include <time.h>
#include <string.h>
#include "option.h"
#include "exttree.h"
#include "inodeinfo.h"
//Функция вычисляет номер блока введенного индексного дескрип-
тора и смещение его в этом блоке, и записывает их в двухэлементный
массив
void inode_offset(FILE *disk, unsigned int inode, int \\
block_size, unsigned int offsets[2])
{
  int inopergro; /*Число индексных дескрипторов в каждой группе
                   *блоков */
  int groupn; //Номер группы блоков
  int gr_desc_size; //Размер описателя группы блоков
  unsigned int inotab; /*Блок, в котором начинается таблица
                     *индексных дескрипторов в данной группе
                     */
  int inode_size; //Размер индексного  дескриптора
  int inoperblo; /*Число индексных дескрипторов в одном блоке
                   *таблицы дескрипторов */
  int offinoblo; /*Относительный номер блока с индексным
                   *дескриптором в группе блоков */
  unsigned int inode_block; /*Номер блока, в котором лежит
                           *индексный дескриптор */
  int inode_num; /*Относительный номер индексного дескриптора в
                   *блоке */
  int inooff; //Смещение индексного дескриптора в блоке
  int p1,p2;
  //Ищем, в какой группе блоков находится индексный дескриптор
  inopergro=read_4(disk,1,1024,40);
  groupn=inode/inopergro;
  if ((inopergro*groupn)==inode) groupn--;
  /*Ищем, в каком блоке начинается таблица индексных
```

```
   *дескрипторов
   */
  gr_desc_size=read_2(disk,1,1024,254);
  if (gr_desc_size==0) gr_desc_size=32;
  if (block_size==1024)
    inotab=read_4(disk,2,block_size,groupn*gr_desc_size+8);
  else inotab=read_4(disk,1,block_size,groupn*gr_desc_size+8);
  //Ищем, в каком блоке лежит индексный дескриптор
  p1=inopergro*groupn;
  p2=inode-p1; /*Относительный номер индексного дескриптора в
            *группе */
  inode_size=read_2(disk,1,1024,88);
  inoperblo=block_size/inode_size;
  offinoblo=p2/inoperblo;
  if ((inoperblo*offinoblo)==p2) offinoblo--;
  inode_block=inotab+offinoblo;
  offsets[0]=inode_block;
  //Ищем относительное смещение индексного дескриптора в блоке
  p1=offinoblo*inoperblo;
  inode_num=p2-p1;
  inooff=(inode_num-1)*inode_size;
  offsets[1]=inooff;
}
//Функция выводит временные метки файла на экран
void time_marks(FILE *disk,unsigned int offsets[2], \\
int block_size)
{
  unsigned int call_time,create_time,modif_time,delete_time;
  time_t call,create,modif,delete;
  struct tm *times;
  int loc=get_local();
  call_time=read_4(disk,offsets[0],block_size,offsets[1]+8);
  create_time=read_4(disk,offsets[0],block_size,offsets[1]+12);
  modif_time=read_4(disk,offsets[0],block_size,offsets[1]+16);
  delete_time=read_4(disk,offsets[0],block_size,offsets[1]+20);
  call=call_time;
  create=create_time;
  modif=modif_time;
  delete=delete_time;
  times=localtime(&call);
```

```
  if (loc==1) printf("Последнее обращение:\t");
  else printf("Last access:\t\t");
  fputs(asctime(times),stdout);
  times=localtime(&create);
  if (loc==1) printf("Создан:\t\t\t");
  else printf("Created:\t\t");
  fputs(asctime(times),stdout);
  times=localtime(&modif);
  if (loc==1) printf("Изменен:\t\t");
  else printf("Modified:\t\t");
  fputs(asctime(times),stdout);
  times=localtime(&delete);
  if (loc==1) printf("Удален:\t\t\t");
  else printf("Deleted:\t\t");
  if (delete_time==0)
  { if (loc==1) printf("Не удалялся\n");
    else printf("Is not deleted yet\n");}
  else fputs(asctime(times),stdout);
}
//Функция выводит на экран тип файла и права доступа к нему
void file_type(FILE *disk,unsigned int offsets[2], \\
int block_size) {
  int type,type2,rights;
  char right[12],resul_right[10];
  char symb_type;
  int i,step;
  int loc=get_local();
  type2=read_2(disk,offsets[0],block_size,offsets[1]);
  if (loc==1) printf("Тип файла:\t\t");
  else printf("File type:\t\t");
  type=type2/4096;
  switch (type) {
    case 8: if (loc==1) printf("Обычный файл\n");
      else printf("Regular file\n"); symb_type='-'; break;
    case 4: if (loc==1) printf("Каталог\n");
      else printf("Directory\n"); symb_type='d'; break;
    case 10: if (loc==1) printf("Символическая ссылка\n");
      else printf("Symlink\n"); symb_type='l'; break;
    case 12: if (loc==1) printf("Сокет\n");
      else printf("Socket\n"); symb_type='s'; break;
```

```
        case 1: if (loc==1) printf("Именованный канал\n");
          else printf("Named pipe\n"); symb_type='f'; break;
        case 6: if (loc==1) printf("Файл блочного устройства\n");
          else printf("Block device\n"); symb_type='b'; break;
        case 2: if (loc==1) printf("Файл символьного устройст-
ва\n");
          else printf("Symbolic device\n");symb_type='c'; break;
        default: if (loc==1) printf("Неизвестный тип файла\n");
          else printf("Unknown file type\n"); symb_type='?';
      }
      rights=type2-(type*4096);
      for (i=0;i<12;i++) {
        step=power(2,(12-i-1));
        if (rights<step) right[i]='-';
        else {
          rights=rights-step;
          if (i==3||i==6||i==9) right[i]='r';
          if (i==4||i==7||i==10) right[i]='w';
          if (i==5||i==8||i==11) right[i]='x';
          if (i<3) right[i]=1;
        }
      }
      resul_right[0]=symb_type;
      for (i=1;i<10;i++) {
        resul_right[i]=right[i+2];
      }
      if (right[0]==1&&right[5]=='x') resul_right[3]='s';
      if (right[0]==1&&right[5]=='-') resul_right[3]='S';
      if (right[2]==1&&right[11]=='x'&&symb_type=='d')
        resul_right[9]='t';
      if (right[2]==1&&right[11]=='-') resul_right[9]='T';
      if (loc==1) printf("Права доступа:\t\t%s\n",resul_right);
      else printf("Rights:\t\t\t%s\n",resul_right);
    }
    /*Функция находит идентификатор пользователя или группы в
     *служебном файле и записывает его имя в символьный массив
     */
    void ugid(FILE *f,int id, char username[60]) {
      int k=0,i,id2;
      char pass[100];
```

```
  do {
    fgets(pass,100,f);
    int l=0,len1,len2;
    char uidstr[6];
    for(i=0;i<100;i++) {
      if (pass[i]==':') l++;
      if (l==2) {len1=i; l++;}
      if (l==4) {len2=i; break;}
    }
    for (i=0;i<(len2-len1-1);i++) uidstr[i]=pass[i+len1+1];
    uidstr[len2-len1]='\0';
    id2=atoi(uidstr);
    k++;
  } while (id2!=id&&k<65536);
  if (id2!=id) username[0]='0';
  else {
    for (i=0;i<60;i++) {
      username[i]=pass[i];
      if (pass[i]==':') {username[i]='\0'; break;}
    }
  }
}
//Определение и вывод на экран расширенных атрибутов файла
void lsattr(int flag_mas[8]) {
  char attr1[4],attr2[4],attr3[4],attr4[4],attr5[4];
  char bin_attr[21],symb_attr[21],str_attr[300];
  int i,str_count=0;
  int loc=get_local();
  str_attr[0]='\0';
  binaring(flag_mas[7],attr1);
  binaring(flag_mas[6],attr2);
  binaring(flag_mas[5],attr3);
  binaring(flag_mas[4],attr4);
  binaring(flag_mas[3],attr5);
  for (i=0;i<4;i++) bin_attr[4-i-1]=attr1[i];
  for (i=0;i<4;i++) bin_attr[8-i-1]=attr2[i];
  for (i=0;i<4;i++) bin_attr[12-i-1]=attr3[i];
  for (i=0;i<4;i++) bin_attr[16-i-1]=attr4[i];
  for (i=0;i<4;i++) bin_attr[20-i-1]=attr5[i];
  bin_attr[20]='\0';
```

```
for (i=0;i<20;i++) {
  if (bin_attr[i]!='0')
  switch(i){
    case 0: symb_attr[i]='s';
      if(str_count!=0) strcat(str_attr,", ");
      if (loc==1)
        strcat(str_attr,"с гарантированным удалением");
      else strcat(str_attr,"secure deletion");
      str_count++;break;
    case 1: symb_attr[i]='u';
      if(str_count!=0) strcat(str_attr,", ");
      if (loc==1) strcat(str_attr,"неудаляемый");
      else strcat(str_attr,"undelete");
      str_count++;break;
    case 2: symb_attr[i]='c';
      if(str_count!=0) strcat(str_attr,", ");
      if (loc==1) strcat(str_attr,"сжатый");
      else strcat(str_attr,"compress file");
      str_count++;break;
    case 3: symb_attr[i]='S';
      if(str_count!=0) strcat(str_attr,", ");
      if (loc==1) strcat(str_attr,"синхронно обновляемый");
      else strcat(str_attr,"synchronous updates");
      str_count++;break;
    case 4: symb_attr[i]='i';
      if(str_count!=0) strcat(str_attr,", ");
      if (loc==1) strcat(str_attr,"недосягаемый");
      else strcat(str_attr,"immutable file");
      str_count++;break;
    case 5: symb_attr[i]='a';
      if(str_count!=0) strcat(str_attr,", ");
      if (loc==1)
        strcat(str_attr,"только добавление в конец");
      else strcat(str_attr,"writes to file may only append");
      str_count++;break;
    case 6: symb_attr[i]='d';
      if(str_count!=0) strcat(str_attr,", ");
      if (loc==1) strcat(str_attr,"недампируемый");
      else strcat(str_attr,"do not dump file");
      str_count++;break;
```

```
            case 7: symb_attr[i]='A';
              if(str_count!=0) strcat(str_attr,", ");
              if (loc==1)
                strcat(str_attr,"не обновлять время доступа");
              else strcat(str_attr,"do not update atime");
              str_count++;break;
            case 14: symb_attr[i]='j';
              if(str_count!=0) strcat(str_attr,", ");
              if  (loc==1)  strcat(str_attr,"с  журналированием  дан-
ных");
              else strcat(str_attr,"file data should be journaled");
              str_count++;break;
            case 15: symb_attr[i]='t';
              if(str_count!=0) strcat(str_attr,", ");
              if (loc==1) strcat(str_attr,"без склеивания хвостов");
              else strcat(str_attr,"file tail should not be merged");
              str_count++;break;
            case 18: symb_attr[i]='-';
              if(str_count!=0) strcat(str_attr,", ");
              if (loc==1) strcat(str_attr,"гигантский файл");
              else strcat(str_attr,"huge file");
              str_count++;break;
            case 19: symb_attr[i]='e';
              if(str_count!=0) strcat(str_attr,", ");
              if (loc==1) strcat(str_attr,"экстенты");
              else strcat(str_attr,"extents");
              str_count++;break;
            default: symb_attr[i]='-';
           }
           else symb_attr[i]='-';
         }
      symb_attr[20]='\0';
      if (loc==1) printf("Расширенные атрибуты:\t%s\n",symb_attr);
      else printf("Extended attributes:\t%s\n",symb_attr);
      printf("              (");
      if   (str_attr[0]=='\0')   {if   (loc==1)   printf("нет");   else
printf("no");}
      else printf("%s",str_attr);
      printf(")\n");
      }
```

```
     /*Если в файле используется адресация блоками, функция выводит
      *номера блоков данных */
     void   block_info(FILE   *disk,unsigned   int   offsets[2],int
block_size)
     {
       int i;
       unsigned int block_number;
       int loc=get_local();
       if (loc==1) printf("Блоки данных: ");
       else printf("Blocks: ");
       for (i=0;i<15;i++)
       {
         block_number=read_4(disk,offsets[0],block_size,   \\   off-
sets[1]+40+4*i);
         if (block_number==0)
           { if(i==0) {if (loc==1) printf("\t\tНет");
             else printf("\t\tNo");} break;}
         else printf("\n   N %d: %d",i+1,block_number);
       }
       printf("\n");
     }
     //Функция выводит на экран информацию о файле
     void print_inode(FILE *disk,unsigned int offsets[2], \\
     int block_size)
     {
       int uid,gid,links;
       unsigned int file_size,sectors;
       int flags,flag_mas[8];
       FILE *passwd,*group;
       char username[60],groupname[60],hex_flags[8],flag4[4];
       int loc=get_local();
       //Временные метки файла
       time_marks(disk,offsets,block_size);
       //Тип файла и права доступа к нему
       file_type(disk,offsets,block_size);
       //Идентификатор владельца
       uid=read_2(disk,offsets[0],block_size,offsets[1]+2);
       passwd=fopen("/etc/passwd","r");
       if (passwd==0) printf("Error opening file passwd\n");
       ugid(passwd,uid,username);
```

```
fclose(passwd);
if (username[0]!='0')
{ if (loc==1) printf("Владелец:\t\t%s\n",username);
  else printf("User:\t\t\t%s\n",username);}
else {if (loc==1) printf("Владелец:\t\t%d\n",uid);
  else printf("User ID:\t\t\t%d\n",uid);}
//Идентификатор группы
gid=read_2(disk,offsets[0],block_size,offsets[1]+24);
group=fopen("/etc/group","r");
if (group==0) printf("Error opening file group\n");
ugid(group,gid,groupname);
fclose(group);
if (groupname[0]!='0')
{ if (loc==1) printf("Группа владельца:\t%s\n",groupname);
  else printf("Group:\t\t\t%s\n",groupname);}
else {if (loc==1) printf("Группа владельца:\t%d\n",gid);
  else printf("Group ID:\t\t\t%s\n",groupname);}
//Размер файла
file_size=read_4(disk,offsets[0],block_size,offsets[1]+4);
if (loc==1) printf("Размер файла:\t\t%d байт\n",file_size);
else printf("File size:\t\t%d bytes\n",file_size);
//Число жестких ссылок на файл
links=read_2(disk,offsets[0],block_size,offsets[1]+26);
if (loc==1) printf("Число связей:\t\t%d\n",links);
else printf("Links:\t\t\t%d\n",links);
//Число секторов по 512 байт, занимаемых файлом
sectors=read_4(disk,offsets[0],block_size,offsets[1]+28);
if (loc==1) printf("Число секторов:\t\t%d\n",sectors);
else printf("Sectors:\t\t%d\n",sectors);
//Флаги файла
flags=read_4(disk,offsets[0],block_size,offsets[1]+32);
hexing(flags,hex_flags,flag_mas);
if (loc==1) printf("Флаги файла:\t\t0x%s\n",hex_flags);
else printf("File flags:\t\t0x%s\n",hex_flags);
//Расширенные атрибуты файла
lsattr (flag_mas);
//Способ адресации
binaring(flag_mas[3],flag4);
if (loc==1) printf("Способ адресации:\t");
else printf("Way of mapping:\t\t");
```

```
        if (flag4[0]=='1')
        {
          {
            if (loc==1) printf("Экстенты\n");
            else printf("Extents\n");
          }
          exttree(disk,offsets,block_size,0);
        }
        else
        {
          {
            if (loc==1) printf("Поблочно\n");
            else printf("Blocks\n");
          }
          block_info(disk,offsets,block_size);
        }
      }
      int inodeinfo(unsigned int inode, FILE *disk)
      {
        int magic,block_size;
        unsigned int max_inode;
        unsigned int offsets[2];
        int loc=get_local();
        //Определяем, является ли файловая система ext-системой
        magic=read_2(disk,1,1024,56); //1024+56
        if (magic!=61267)
        {
          if (loc==1)
            printf("Файл устройства не содержит магического чис-
ла\n");
          else printf("File does not contain magic number\n");
          return(1);
        }
        //Вычисляем размер блока
        block_size=get_block_size(disk);
        if (block_size==0)
        {
          {if (loc==1) printf("Ошибка чтения размера блока дан-
ных\n");
           else printf("Error in block size reading\n");} return (2);
```

```
    }
  /*Вычисляем количество индексных дескрипторов в файловой
   *системе */
  max_inode=read_4(disk,1,1024,0);
  if (inode<1||inode>max_inode)
  {
    if (loc==1)
      printf("Вы ввели несуществующий индексный дескриптор\n");
    else printf("Such inode does not exist\n");
    return(3);
  }
  /*Вычисляем номер блока и смещение в блоке, по которому
   *находится индексный дескриптор */
  inode_offset(disk,inode,block_size,offsets);
  if (loc==1)
    printf("Индексный дескриптор %d находится в блоке: %d ",\\
inode,offsets[0]);
  else printf("Inode %d is in block: %d ",inode,offsets[0]);
  if (loc==1) printf("по смещению: %d байт\n",offsets[1]);
  else printf("offset is: %d bytes\n",offsets[1]);
  //Выводим информацию о файле
  print_inode(disk,offsets,block_size);
  return (0);
}
```

Модуль super

Файл super.h

```
int super(FILE *disk);
```

Файл super.c

```
#include <stdio.h>
#include <time.h>
#include "super.h"
#include "option.h"
int super (FILE *disk)
{
  int magic;
  int loc=get_local();
  int first_block,block_size,fragment_size;
  unsigned int inode_count, block_count, free_blocks;
```

```
       unsigned int free_inodes, su_blocks;
       int   frag_indic,   blopergro,   frapergro,   inopergro,
mount_count;
       unsigned int max_mount,os_type;
       unsigned int cur_flags,err_flags;
       int fs_version,def_uid,def_gid,first_inode,inode_size;
       unsigned                                              int
compat,incompat,rocompat,jor_inode,first_meta;
       unsigned int inode_min,inode_wanted,flex_size;
       unsigned int last_mount, last_write, last_check;
       unsigned int check_int,create;
       char
hex[8],hex_compat[8],hex_incompat[8],hex_rocompat[8];
       int del[8],i;
       time_t mount_time,write_time,check_time,create_time;
       struct tm *times;
       magic=read_2(disk,1,1024,56);
       if (magic!=61267)
       {
         if (loc==1)
           printf("Файл устройства не содержит магического чис-
ла\n");
         else printf("File does not contain magic number\n");
         return(1);
       }
       if  (loc==1)  printf("\tИнформация  о  файловой  систе-
ме:\n");
       else printf("\tFilesystem information:\n");
       inode_count=read_4(disk,1,1024,0);
     if        (loc==1)        printf("Число        индексных
дескрипторов:\t\t%d\n",inode_count);
       else printf("Inode count:\t\t\t%d\n",inode_count);
       block_count=read_4(disk,1,1024,4);
       if (loc==1)
        printf("Число         блоков         в         файловой
системе:\t%d\n",block_count);
       else printf("Block count:\t\t\t%d\n",block_count);
       su_blocks=read_4(disk,1,1024,8);
       if (loc==1)
       printf("Число                 блоков                 для
```

```
суперпользователя:\t%d\n",su_blocks);
      else          printf("Block          count          for
superuser:\t%d\n",su_blocks);
      free_blocks=read_4(disk,1,1024,12);
      if (loc==1)
        printf("Число                                свободных
блоков:\t\t\t%d\n",free_blocks);
      else printf("Free block count:\t\t%d\n",free_blocks);
      free_inodes=read_4(disk,1,1024,16);
      if (loc==1)
      printf("Число свободных индексных дескрипторов:\t%d\n",
\\ free_inodes);
      else printf("Free inodes count:\t\t%d\n",free_inodes);
      first_block=read_4(disk,1,1024,20);
      if (loc==1)
       printf("Номер         первого         блока         с
данными:\t\t%d\n",first_block);
      else     printf("First     block     containing     da-
ta:\t%d\n",first_block);
      block_size=get_block_size(disk);
      if (loc==1)
        printf("Размер блока данных:\t\t\t%d\n",block_size);
      else printf("Logical block size:\t\t%d\n",block_size);
      frag_indic=read_4(disk,1,1024,28);
      switch(frag_indic)
      {
        case 0: fragment_size=1024; break;
        case 1: fragment_size=2048; break;
        case 2: fragment_size=4096; break;
        default: fragment_size=0;
      }
      if (loc==1)
        printf("Размер фрагмента:\t\t\t%d\n",fragment_size);
      else printf("Fragment_size:\t\t\t%d\n",fragment_size);
      blopergro=read_4(disk,1,1024,32);
      if          (loc==1)          printf("Блоков          в
группе:\t\t\t%d\n",blopergro);
      else printf("Blocks per group:\t\t%d\n",blopergro);
      frapergro=read_4(disk,1,1024,36);
      if (loc==1)
```

```
        printf("Фрагментов в группе:\t\t\t%d\n",frapergro);
      else printf("Fragments per group:\t\t%d\n",frapergro);
      inopergro=read_4(disk,1,1024,40);
      if (loc==1)
        printf("Индексных дескрипторов в
группе:\t%d\n",inopergro);
      else printf("Inodes per group:\t\t%d\n",inopergro);
      create=read_4(disk,1,1024,264);
      create_time=create;
      times=localtime(&create_time);
      if (loc==1) printf("Время создания:\t\t\t\t");
      else printf("Creation time:\t\t\t");
      fputs(asctime(times),stdout);
      last_mount=read_4(disk,1,1024,44);
      mount_time=last_mount;
      times=localtime(&mount_time);
      if (loc==1) printf("Последнее монтирование:\t\t\t");
      else printf("Last mounted:\t\t\t");
      fputs(asctime(times),stdout);
      last_write=read_4(disk,1,1024,48);
      write_time=last_write;
      times=localtime(&write_time);
      if (loc==1) printf("Время последней записи:\t\t\t");
      else printf("Last write time:\t\t");
      fputs(asctime(times),stdout);
      last_check=read_4(disk,1,1024,64);
      check_time=last_check;
      times=localtime(&check_time);
      if (loc==1) printf("Время последней проверки ФС:\t\t");
      else printf("Last check time:\t\t");
      fputs(asctime(times),stdout);
      check_int=read_4(disk,1,1024,68);
      if (loc==1)
        printf("Максимальный интервал между проверка-
ми:\t%d\n", \ check_int);
      else printf("Maximum check inter-
val:\t\t%d\n",check_int);
      mount_count=read_2(disk,1,1024,52);
      if (loc==1)
        printf("Число монтирований:\t\t\t%d\n",mount_count);
```

```
       else printf("Mount count:\t\t\t%d\n",mount_count);
       max_mount=read_2(disk,1,1024,54);
       if (loc==1)
         printf("Предельное                                    число
монтирований:\t\t%d\n",max_mount);
       else printf("Maximum mount count:\t\t%d\n",max_mount);
       hexing(magic,hex,del);
       if (loc==1) printf("Магическое число:\t\t\t0x");
       else printf("Magic number:\t\t\t0x");
       for (i=4;i<8;i++) printf("%c",hex[i]);
       printf("\n");
       cur_flags=read_2(disk,1,1024,58);
       hexing(cur_flags,hex,del);
       if (loc==1) printf("Флаги текущего состояния:\t\t0x");
       else printf("Current flags:\t\t\t0x");
       for (i=4;i<8;i++) printf("%c",hex[i]);
       printf("\n");
       err_flags=read_2(disk,1,1024,60);
       hexing(err_flags,hex,del);
       if (loc==1) printf("Флаги обработки ошибок:\t\t\t0x");
       else printf("Error flags:\t\t\t0x");
       for (i=4;i<8;i++) printf("%c",hex[i]);
       printf("\n");
       os_type=read_4(disk,1,1024,72);
       if (loc==1) printf("Тип ОС:\t\t\t\t\t");
       else printf("OS type:\t\t\t");
       if (os_type==0) printf("Linux\n");
       else printf("%d\n",os_type);
       fs_version=read_4(disk,1,1024,76);
       if (loc==1)
         printf("Версия файловой системы:\t\t%d\n",fs_version);
       else printf("Filesystem version:\t\t%d\n",fs_version);
       def_uid=read_2(disk,1,1024,80);
       if           (loc==1)           printf("UID           по
умолчанию:\t\t\t%d\n",def_uid);
       else printf("Default UID:\t\t\t%d\n",def_uid);
       def_gid=read_2(disk,1,1024,82);
       if           (loc==1)           printf("GID           по
умолчанию:\t\t\t%d\n",def_gid);
       else printf("Default GID:\t\t\t%d\n",def_gid);
```

```
       first_inode=read_4(disk,1,1024,84);
       if (loc==1)
         printf("Первый                                    индексный
дескриптор:\t\t%d\n",first_inode);
       else              printf("First              non-reserved
inode:\t%d\n",first_inode);
       inode_size=read_2(disk,1,1024,88);
       if (loc==1)
         printf("Размер                                    индексного
дескриптора:\t\t%d\n",inode_size);
       else printf("Inode size:\t\t\t%d\n",inode_size);
       compat=read_4(disk,1,1024,92);
       hexing(compat,hex_compat,del);
       if (loc==1) printf("Флаги COMPAT:\t\t\t\t0x");
       else printf("COMPAT flags:\t\t\t0x");
       for(i=0;i<8;i++) printf("%c",hex_compat[i]);
       printf("\n");
       incompat=read_4(disk,1,1024,96);
       hexing(incompat,hex_incompat,del);
       if (loc==1) printf("Флаги INCOMPAT:\t\t\t\t0x");
       else printf("INCOMPAT flags:\t\t\t0x");
       for (i=0;i<8;i++) printf("%c",hex_incompat[i]);
       printf("\n");
       rocompat=read_4(disk,1,1024,100);
       hexing(rocompat,hex_rocompat,del);
       if (loc==1) printf("Флаги ROCOMPAT:\t\t\t\t0x");
       else printf("ROCOMPAT flags:\t\t\t0x");
       for(i=0;i<8;i++) printf("%c",hex_rocompat[i]);
       printf("\n");
       jor_inode=read_4(disk,1,1024,224);
       if (loc==1)
         printf("Индексный                                 дескриптор
журнала:\t\t%d\n",jor_inode);
       else printf("Journal inode:\t\t\t%d\n",jor_inode);
       first_meta=read_4(disk,1,1024,260);
       if (loc==1)
         printf("Первая                                    метагруппа
блоков:\t\t%d\n",first_meta);
       else printf("First metagroup:\t\t%d\n",first_meta);
       inode_min=read_2(disk,1,1024,348);
```

```
       if (loc==1)
         printf("Минимальный          размер          дескрипто-
ра:\t\t%d\n",inode_min);
       else printf("Inode extra isize:\t\t%d\n",inode_min);
       inode_wanted=read_2(disk,1,1024,350);
       if (loc==1)
         printf("Желаемый                                  размер
дескриптора:\t\t%d\n",inode_wanted);
       else printf("Inode wanted isize:\t\t%d\n",inode_wanted);
       fseek(disk,1396,SEEK_SET);
       flex_size=fgetc(disk);
       if (loc==1)
         printf("Размер              гибкой              группы
блоков:\t\t%d\n",flex_size);
       else printf("Flex block group size:\t\t%d\n",flex_size);
       return (0);
     }
```

ПРИЛОЖЕНИЕ 5

ПРИМЕРЫ РАБОТЫ С ПРОГРАММОЙ EXTVIEW

Ниже представлены листинги с примерами работы программы. **Extview** имеет в своем составе различные опции, состав которых может быть расширен в дальнейшем.

Справка по программе выводится командой **extview --help**. На рис. П.5.1 показан вывод этой команды.

```
Extview - программа для просмотра файловых систем ext2/3/4.
Для выполнения программы нужно право на чтение диска или образа диска.
Может выполняться как на монтированной системе, так и на размонтированной.
Открывает файл устройства только для чтения, ничего не изменяя на диске.
Использование (опции):
-i <inode_number> <device>
      Выводит информацию об индексном дескрипторе с номером <inode_number>
      <device> - раздел диска или образ файла блочного устройства
      Например: extview -i 2 /dev/sda1
-s <device>
      Выводит информацию о файловой системе на устройстве <device>
      Например: extview -s /dev/sda4
-g <device>
      Выводит информацию о группах блоков в файловой системе
      на устройстве <device>
      Например: extview -g /dev/hda5
-b <block_number> <device>
      Выводит шестнадцатеричное представление блока с номером <block_number>
      с устройства <device>
      Например: extview -b 997 /dev/sda5
-d <dir_inode/dir_block> <device> [B]
      Выводит элементы директории, указанной по индексному дескриптору, в том
      числе удаленные и скрытые файлы. Если указана необязательная опция B,
      интерпретирует входной параметр как номер блока и пытается прочесть блок
      данных как каталог.
      Например: extview -d 2 /dev/hda4 - выводит корневой каталог
-j <journal_block_number> <device> [B]
      Выводит информацию об указанном блоке из журнала устройства <device>
      Если указать опцию B, выводит еще и сам блок данных журнала
      Например: extview -j 300 /dev/sda5 B
```

```
-l [b/i] [<inode_number/block_number>] <device>
      Выводит список блоков журнала, в которых есть упоминание об указанном
      индексном дескрипторе. Если указана необязательная опция b, то выводит
      список журнальных блоков с этим блоком. Если перед номером индексного
      дескриптора опций не указано, то считает введенный номер номером
      индексного дескриптора
      При указании только имени устройства выводит список удаленных индексных
      дескрипторов на этом устройстве
      Например: extview -l i 161121 /dev/sda3
-c <device>
      Поиск ключевой фразы на устройстве <device>. При запуске требуется
      ввести фразу для поиска и диапазон блоков, в котором следует вести поиск
      Выдает номера блоков данных, где была найдена ключевая фраза.
      Например: extview -c /dev/sda2--inodeblock <inode_number> <device>
      Выводит шестнадцатеричное представление индексного
      дескриптора <inode_number> с устройства <device>
      Например: extview --inodeblock 2 /dev/hdb3
-h, --help
      Выводит справку о программе
```

Рис. П.5.1. Справочная информация

Индексный дескриптор файла можно вывести с помощью опции **–i**. На рис. П.5.2 показан вывод информации об индексном дескрипторе для небольшого файла, представленного двумя экстентами. Команда **extview –i 131634 /dev/sda5**.

```
Индексный дескриптор 131634 находится в блоке: 524371 по смещению: 256 байт
Последнее обращение:    Sun Dec 27 12:41:08 2009
Создан:                 Sun Dec 27 12:41:08 2009
Изменен:                Tue Dec 19 05:33:13 2006
Удален:                 Не удалялся
Тип файла:              Обычный файл
Права доступа:          -rwxrwxrwx
Владелец:               madcat
Группа владельца:       madcat
Размер файла:           4540213 байт
Число связей:           1
Число секторов:         8872
Флаги файла:            0x00080000
Расширенные атрибуты:   ------------------e
```

```
                        (экстенты)
Способ адресации: Экстенты
1-й экстент: первый блок: 1231872 (0x0012CC00), длина: 1024
2-й экстент: первый блок: 1275904 (0x00137800), длина: 85
```

Рис. П.5.2. Вывод информации об индексном дескрипторе среднего файла

Для файла с количеством экстентов, большим, чем 4, программа выводит дерево экстентов полностью. Пример представлен на рис. П.5.3, команда **`extview -i 161121 /dev/sda5`**.

```
Индексный дескриптор 161121 находится в блоке: 526214 по смещению: 0 байт
Последнее обращение:    Sun Feb 14 11:41:56 2010
Создан:                 Fri Nov 27 21:11:59 2009
Изменен:                Fri Nov 27 21:11:59 2009
Удален:                 Не удалялся
Тип файла:              Обычный файл
Права доступа:          -rw-r--r--
Владелец:               madcat
Группа владельца:       madcat
Размер файла:           777038562 байт
Число связей:           1
Число секторов:         1517664
Флаги файла:            0x00080000
Расширенные атрибуты:   ------------------e
                        (экстенты)
Способ адресации: Экстенты
Следующий узел дерева в блоке: 786607 (0x000C00AF)
      1-й экстент: первый блок: 804864 (0x000C4800), длина: 2048
      2-й экстент: первый блок: 813056 (0x000C6800), длина: 4096
      3-й экстент: первый блок: 894976 (0x000DA800), длина: 30720
      4-й экстент: первый блок: 925696 (0x000E2000), длина: 21504
      5-й экстент: первый блок: 947200 (0x000E7400), длина: 29696
      6-й экстент: первый блок: 976896 (0x000EE800), длина: 13076
      7-й экстент: первый блок: 989972 (0x000F1B14), длина: 32736
      8-й экстент: первый блок: 1022708 (0x000F9AF4), длина: 25868
      9-й экстент: первый блок: 1114112 (0x00110000), длина: 29963
```

Рис. П.5.3. Вывод информации об индексном дескрипторе большого файла

Вывод индексного дескриптора в шестнадцатеричном виде представлен на рис. П.5.4. Команда

extview --inodeblock 131634 /dev/sda5

```
Индексный дескриптор 131634 лежит в блоке 524371 по смещению 256

0x00000000  FF 81 E8 03 35 47 45 00 : 94 0F 37 4B 94 0F 37 4B  ....5GE...7K..7K
0x00000010  49 33 87 45 00 00 00 00 : E8 03 01 00 A8 22 00 00  I3.E.........".."
0x00000020  00 00 08 00 01 00 00 00 : 0A F3 02 00 04 00 00 00  ................
0x00000030  00 00 00 00 00 00 00 00 : 00 04 00 00 00 CC 12 00  ................
0x00000040  00 04 00 00 55 00 00 00 : 00 78 13 00 00 00 00 00  ....U....x......
0x00000050  00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00  ................
0x00000060  00 00 00 00 80 87 68 6A : 00 00 00 00 00 00 00 00  ......hj........
0x00000070  00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00  ................
0x00000080  1C 00 00 00 60 CA CD 61 : 00 00 00 00 84 BD D6 9E  ....`..a........
0x00000090  94 0F 37 4B 38 0C FB 34 : 00 00 00 00 00 00 00 00  ..7K8..4........
0x000000A0  00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00  ................
0x000000B0  00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00  ................
0x000000C0  00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00  ................
0x000000D0  00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00  ................
0x000000E0  00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00  ................
0x000000F0  00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00  ................
```

Рис. П.5.4. Вывод шестнадцатеричного представления индексного дескриптора

Вывод информации о файловой системе в целом представлен на рис. П.5.5. Команда **extview –s /dev/sda5**.

```
      Информация о файловой системе:
Число индексных дескрипторов:       605024
Число блоков в файловой системе:    2415766
Число блоков для суперпользователя: 120788
Число свободных блоков:             1247814
Число свободных индексных дескрипторов:   378130
Номер первого блока с данными:            0
Размер блока данных:                4096
Размер фрагмента:                   4096
Блоков в группе:                    32768
Фрагментов в группе:                32768
```

```
Индексных дескрипторов в группе:    8176
Время создания:                     Mon Nov  2 16:24:05 2009
Последнее монтирование:             Mon Feb 22 22:29:25 2010
Время последней записи:             Mon Feb 15 18:17:16 2010
Время последней проверки ФС:        Sat Feb 13 18:03:59 2010
Число монтирований:                 18
Предельное число монтирований:      29
Магическое число:                   0xEF53
Флаги текущего состояния:           0x0001
Флаги обработки ошибок:             0x0001
UID по умолчанию:                   0
GID по умолчанию:                   0
Первый индексный дескриптор:        11
Размер индексного дескриптора:      256
Флаги COMPAT:                       0x0000003C
Флаги INCOMPAT:                     0x00000246
Флаги ROCOMPAT:                     0x0000007B
Индексный дескриптор журнала:       8
Размер гибкой группы блоков:        4
```

Рис. П.5.5. Вывод информации о файловой системе

На рис. П.5.6 представлен вывод программой информации о журнальном суперблоке, команда **`extview –j 0 /dev/sda5`**

```
Первый блок журнала:          1081344
Номер последовательности:     0
Тип блока:                    Журнальный суперблок v.2
Размер блоков журнала:        4096
Число блоков в журнале:       32768
Первый блок с информацией:    1
Первый ожидаемый ID:          91310
Блок начала журнала:          1
Код ошибки:                   0
Флаги COMPAT:                 0x00000000
Флаги INCOMPAT:               0x00000001
Флаги ROCOMPAT:               0x00000000
Журнальный UUID:              AACA0A90-3A1241B8-9F3DC82B-1B48680C
Число ФС для этого журнала:   1
Копия суперблока в блоке:     0
```

Рис. П.5.6. Вывод информации о журнальном суперблоке

Вывод журнального блока, который не является дескриптором, представлен на рис. П.5.7. В команду добавлена опция «B», благодаря чему выведен весь блок данных, а не только информация о предыдущем и следующем блоках-дескрипторах. Команда **extview –j 1805 /dev/sda5 B**

```
Первый блок журнала:       1081344
Тип блока:                 Обычный блок
  Предыдущий дескриптор:          Следующий дескриптор:
  Блок 1800                       Блок 1810

0x0070D000  A1 1F 02 00 0C 00 01 02 : 2E 00 00 00 A0 1F 02 00  ................
0x0070D010  0C 00 02 02 2E 2E 00 00 : BA 16 02 00 18 00 0C 01  ................
0x0070D020  73 6D 62 2D 6E 65 74 77 : 6F 72 6B 3A 51 32 55 55  smb-network:Q2UU
0x0070D030  FD 01 02 00 14 00 09 01 : 63 6F 6D 70 75 74 65 72  ........computer
```

Рис. П.5.7. Фрагмент вывода журнального блока

Вывод информации о группах блоков представлен на рис. П.5.8. Команда **extview –g /dev/sda5**

```
Число групп блоков: 74
Группа 0:   карта блоков: 591, карта дескрипторов: 607,
            таблица индексных дескрипторов: 623,
            свободных блоков: 3436, свободных дескрипторов: 0,
            директорий: 4, неиспользованных дескрипторов: 0.
            Контрольная сумма: 0x3462

Группа 1:   карта блоков: 592, карта дескрипторов: 608,
            таблица индексных дескрипторов: 1134,
            свободных блоков: 4107, свободных дескрипторов: 32,
            директорий: 753, неиспользованных дескрипторов: 0.
            Контрольная сумма: 0x9185

и т. д.
```

Рис. П.5.8. Фрагмент вывода информации о группах блоков

На рис. П.5.9 представлен вывод программой указанного блока данных. Команда **extview –b 590848 /dev/sda5**. В этом блоке содержится фрагмент HTML-файла.

```
0x90400000  3C 21 44 4F 43 54 59 50 : 45 20 48 54 4D 4C 20 50  <!DOCTYPE HTML P
0x90400010  55 42 4C 49 43 20 22 2D : 2F 2F 57 33 43 2F 2F 44  UBLIC "-//W3C//D
0x90400020  54 44 20 48 54 4D 4C 20 : 34 2E 30 31 20 54 72 61  TD HTML 4.01 Tra
0x90400030  6E 73 69 74 69 6F 6E 61 : 6C 2F 2F 45 4E 22 3E 0A  nsitional//EN">.
0x90400040  0A 3C 68 74 6D 6C 3E 0A : 20 20 3C 68 65 61 64 3E  .<html>.  <head>
0x90400050  0A 20 20 20 20 3C 6D 65 : 74 61 0A 20 20 20 20 20  .    <meta.
0x90400060  6E 61 6D 65 3D 22 67 65 : 6E 65 72 61 74 6F 72 22  name="generator"
0x90400070  0A 20 20 20 20 20 63 6F : 6E 74 65 6E 74 3D 0A 20  .     content=.
0x90400080  20 20 20 22 48 54 4D 4C : 20 54 69 64 79 20 66 6F     "HTML Tidy fo
0x90400090  72 20 4C 69 6E 75 78 2F : 78 38 36 20 28 76 65 72  r Linux/x86 (ver
0x904000A0  73 20 31 73 74 20 4A 75 : 6C 79 20 32 30 30 32 29  s 1st July 2002)
```

Рис. П.5.9. Фрагмент шестнадцатеричного представления блока данных

Вывод директорий по индексному дескриптору приведен на рис. П.5.10, в том числе указываются удаленные и скрытые файлы. Команда **extview –d 131360 /dev/sda5**

```
Первый блок директории: 532637
      131360, . (dir)
      287, .. (dir)
      136095, apelsin.jpg
      139264, Снимки (dir)
      138745, Шихан.jpg
      172731, 461920.jpg
      138306, 459436.jpg
      138690, Невьянск.jpg
      138862, Верхотурье.jpg
      172743, dir.png
      172771, joursup.png
      137613, lost.png
      171414, lde-super.png
      160758, before.png - удален
      160892, after.png
      161424, dirview.png - удален
```

```
      160807, journal.png
      160809, find.png - удален
      161388, helpp.png
      163184, tree.png
      172812, jourblock.png
```

Рис. П.5.10. Вывод записей в директории

На рис. П.5.11 представлен вывод программой списка удаленных индексных дескрипторов. Перед выполнением поиска программа запрашивает время, после которого следует искать дескрипторы. Команда **`extview –l /dev/sda5`**

```
Введите время удаления, после которого искать файлы
1266850000
Mon Feb 22 19:46:40 2010
Список удаленных индексных дескрипторов:
1     178952, удален Mon Feb 22 23:14:42 2010
2     179092, удален Mon Feb 22 23:14:42 2010
3     237127, удален Mon Feb 22 23:14:42 2010
4     237133, удален Mon Feb 22 23:14:42 2010
Файлы, удаленные после: (1266850000) Mon Feb 22 19:46:40 2010
```

Рис. П.5.11. Список недавно удаленных файлов