Факультет компьютерных технологий и прикладной математики

Кафедра вычислительных технологий

02.03.02

Информационная безопасность

Лабораторная работа № 4

Тема: Разграничение прав доступа. Файловые подсистемы.

Цель работы

Изучение механизмов управления доступа к ресурсам, прав доступа. Постижение понятия пользователя и группы. Приобретение практических навыков управления пользователями при помощи консольных утилит. Приобретение навыков работы с правами пользователей и правами на файлы, каталоги при помощи консольных утилит.

Получение теоретических и практических навыков работы с таблицами разделов (MBR и GPT), создания разделов и файловых систем.

Указания к работе

Вначале студенты изучают теоретическую часть. Далее каждый студент должен выполнить задания, а также ответить на вопросы к лабораторной работе. За проделанную работу студент может получить оценку от «неудовлетворительно» до «отлично». Для получения оценки «удовлетворительно» студент должен выполнить ВСЕ задания к лабораторной работе. Оценка «хорошо» ставится, если студент ответил на ВСЕ вопросы к лабораторной работе. Оценку «отлично» студент получает, если подготовлен отчёт по лабораторной работе.

ОЦЕНКУ ЗА ПРОДЕЛАННУЮ РАБОТУ МОЖНО ПОВЫСИТЬ ДО СДАЧИ СЛЕДУЮЩЕЙ ЛАБОРАТОРНОЙ РАБОТЫ.

Теоретическая часть

У каждого объекта (файла) есть уникальное имя, по которому к нему можно обращаться, и конечный набор операций, которые процессы могут

выполнять в отношении этого объекта. Файлу свойственны операции read, write и execute.

Совершенно очевидно, что нужен способ запрещения процессам доступа к тем объектам, к которым у них нет прав доступа. Более того, этот механизм должен также предоставлять возможность при необходимости ограничивать процессы поднабором разрешенных операций. Например, процессу A может быть дано право проводить чтение данных из файла F, но не разрешено вести запись в этот файл.

Права доступа означают разрешение на выполнение той или иной операции (чтение, записи, исполнения).

Когда пользователь входит в систему, его оболочка получает UID и GID (UID – идентификатор пользователя, GID – идентификатор группы), которые содержатся в его записи в файле паролей, и они наследуются всеми его дочерними процессами. Представляя любую комбинацию (UID, GID), можно составить полный список всех объектов (файлов, включая устройства ввода-вывода, которые представлены в виде специальных файлов и т.д.), к которым процесс может обратиться с указанием возможного типа доступа (чтение, запись, исполнение).

Два процесса с одинаковой комбинацией (UID, GID) будут иметь абсолютно одинаковый доступ к одинаковому набору объектов. Процессы с различающимися значениями (UID, GID) будут иметь доступ к разным наборам файлов, хотя, может быть, и со значительным перекрытием этих наборов.

SUID (Set User ID)

Атрибут исполняемого файла, позволяющий запустить его с правами владельца. В операционных системах Linux приложение запускается с правами пользователя, запустившего указанное приложение. Это обеспечивает дополнительную безопасность т.к. процесс с правами пользователя не сможет получить доступ на запись к важным системным файлам, например /etc/passwd, который принадлежит суперпользователю root.

Если на исполняемый файл установлен бит suid, то при выполнении эта программа автоматически меняет «эффективный userID» на идентификатор того пользователя, который является владельцем этого файла. То есть, не зависимо от того кто запускает эту программу, она при выполнении имеет права хозяина этого файла.

SGID (Set Group ID)

Аналогичен SUID, но относиться к группе. При этом, если для каталога установлен бит SGID, то создаваемые в нем объекты будут получать группу владельца каталога, а не пользователя.

Практические примеры

Узнать права на файл/директорию

```
sit@ubuntu:~$ ls -l /bin/ls
-rwxr-xr-x 1 root root 129280 Feb 18  2016 /bin/ls
```

Права доступа состоят из трех троек символов. Первая тройка представляет права владельца файла, вторая представляет права группы файла и третья права всех остальных пользователей.

В нашем случае это:

- «rwx» – Права владельца файла;
- «r-x» – Права группы файла;
- «r-x» – Права всех остальных на файл.

Символ «r» означает, что чтение (просмотр данных содержащихся в файле) разрешено, «w» означает запись (изменение, а также удаление данных) разрешено и «x» означает исполнение (запуск программы разрешен).

Таким образом, если в целом посмотреть на права мы увидим, что кому угодно разрешено читать содержимое и исполнять этот файл, но только владельцу (root) разрешено как-либо модифицировать этот файл. Иными словами, нормальным пользователям разрешено копировать содержимое этого файла, то только root может изменять или удалять его.

Определение текущего пользователя и групп в которых он состоит

Перед тем, как изменять владельца или группу которой принадлежит файл, необходимо уметь определять текущего пользователя и группу к которой он принадлежит. Чтобы узнать под каким пользователем вы работаете, наберите whoami:

```
sit@ubuntu:~$ whoami
sit
```

Для определения в каких группах состоит пользователь sit, необходимо воспользоваться командой groups:

```
sit@ubuntu:~$ groups
sit adm cdrom sudo dip plugdev lxd lpadmin sambashare
```

Из этого примера видно, что пользователь sit состоит в группах sit, adm, cdrom, sudo, dip, plugdev, lxd, lpadmin, sambashare. Если вы хотите посмотреть, в каких группах состоит другой пользователь, то передайте его имя в качестве аргумента.

```
sit@ubuntu:~$ groups root
root : root
```

Изменение пользователя и группы владельца

Чтобы изменить владельца или группу файла (или другого объекта) используется команды chown или chgrp соответственно. Сначала нужно передать имя группы или владельца, а потом список файлов.

```
chown sit /home/sit/itmo.txt
chgrp sit /home/sit/itmo.txt
```

Можно также изменять пользователя и группу одновременно используя команду chown в другой форме:

```
chown sit:sit /home/sit/itmo.txt
```

Вы не можете использовать команду chown без прав суперпользователя, но chgrp может быть использована всеми, чтобы изменить группу-владельца файла на ту группу, к которой они принадлежат.

Знакомство с chmod

chown и chgrp используются для изменения владельца и группы объекта файловой системы, но кроме них существует и другая программа, называемая chmod, которая используется для изменения прав доступа на чтение, запись и исполнение, которые мы видим в выводе команды ls -l. Команда chmod использует два и более аргументов: метод, описывающий как именно необходимо изменить права доступа с последующим именем файла или списком файлов, к которым необходимо применить эти изменения:

```
chmod +x /home/sit/itmo.sh
```

В примере выше в качестве метода указано +x. Как можно догадаться, метод +x указывает chmod, что файл необходимо сделать исполняемым для пользователя, группы и для всех остальных. Если мы решим отнять все права на исполнение файла, то сделаем вот так:

```
chmod +x /home/sit/itmo.sh
```

Разделение между пользователем, группой и всеми остальными

Часто бывает удобно изменить только один или два набора прав доступа за раз. Чтобы сделать это, просто необходимо использовать специальный символ для обозначения набора прав доступа, который необходимо изменить, со знаком «+»» или «-» перед ним. Символ «u» для пользователя, «g» для группы и «o» для остальных пользователей.

```
chmod go-w /home/sit/itmo.sh
```

Данный пример удаляет право на запись для группы и всех остальных пользователей, но оставляет права владельца нетронутыми.

Числовые режимы

Существует еще один достаточно распространенный способ указания прав: использование четырехзначных восьмеричных чисел. Этот синтаксис, называется числовым синтаксисом прав доступа, где каждая цифра представляет тройку разрешений. Например, в 0777, 777 устанавливают флаги для владельца, группы, и остальных пользователей. Ниже таблица показывающая как транслируются права доступа на числовые значения.

Режим	Число
rwx	7
rw-	6
r-x	5
r--	4
-wx	3
-w-	2
--x	1
---	0

umask

Когда процесс создает новый файл, он указывает, какие права доступа нужно задать для данного файла. Зачастую запрашиваются права 0666 (чтение и запись всеми), что дает больше разрешений, чем необходимо в большинстве случаев. К счастью, каждый раз, когда в Linux создается новый файл, система обращается к параметру, называемому umask. Система использует значение umask чтобы понизить изначально задаваемые разрешения на что-то более разумное и безопасное. Вы можете просмотреть текущие настройки umask набрав umask в командной строке:

```
sit@ubuntu:~$ umask
0002
```

В Linux-системах значением по умолчанию для umask является 0022, что позволяет другим читать ваши новые файлы (если они могут до них добраться), но не изменять их. Чтобы автоматически обеспечивать больший уровень защищенности для создаваемых файлов, можно изменить настройки umask:

```
sit@ubuntu:~$ umask 0077
```

Такое значение umask приведет к тому, что группа и прочие не будут иметь совершенно никаких прав доступа для всех, вновь созданных файлов.

В отличие от «обычного» назначения прав доступа к файлу, umask задает какие права доступа должны быть отключены. Снова посмотрим на таблицу соответствия значений чисел и методов:

Режим	Число
rwx	7
rw-	6
r-x	5
r--	4
-wx	3
-w-	2
--x	1
---	0

Воспользовавшись этой таблицей мы видим, что последние три знака в 0077 обозначают rwxrwx. umask показывает системе, какие права доступа отключить. Совместив первое и второе становится видно, что все права для группы и остальных пользователей будут отключены, в то время как права владельца останутся нетронутыми.

Изменение suid и sgid

Способ установки и удаления битов suid и sgid чрезвычайно прост. Чтобы задать бит suid:

```
chmod u+s /home/sit/itmo.sh
```

Чтобы задать бит sgid:

```
chmod g+s /home/sit/itmo/
```

Определение первого знака прав доступа

Он используется для задания битов sticky, suid и sgid совместно с правами доступа:

suid	sgid	sticky	режим
on	on	on	7

```
on    on    off   6
on    off   on    5
on    off   off   4
off   on    on    3
off   on    off   2
off   off   on    1
off   off   off   0
```

Ниже приведен пример того, как использовать четырехзначный режим для установки прав доступа на директорию.

```
sit@ubuntu:~$ chmod 4775 /home/sit/itmo
sit@ubuntu:~$ ls -l /home/sit/itmo
-rwsrwxr-x 1 sit sit 0 Sep  9 12:42 /home/sit/itmo
```

Консольные команды

- id <печать идентификатора пользователя>.
- chgrp <изменить группу файла>.
- chown <изменить владельца и группу файлов>.
- chmod <изменить права доступа к файлу>.
- usermod <изменение параметров учетной записи пользователя>.
- useradd <создание нового пользователя>.
- userdel <удаление пользователя>.
- whoami <определение текущего пользователя>.
- umask <определение или установление маски прав доступа для вновь создаваемых файлов>.
- sudo su <получение прав суперпользователя>.
- groups <определение к каким группам принадлежит пользователь>.

Консольные команды

- fdisk <параметры> – Консольная программа для управления дисками (Работает только с MBR).

• parted <параметры> – Консольная программа для управления дисками (Работает как с MBR, так и с GPT).

• dd <параметры> – Консольная программа копирования данных.

• mkfs.<тип файловой системы> <раздел диска> – Класс консольных команд создания файловых систем на разделах.

• mount -t <тип файловой системы> <раздел диска> <точка монтирования> – Консольная программа монтирования разделов жесткого диска.

Диск делится на разделы. Как именно диск делится на разделы, определяется таблицей разделов. Таблицы разделов бывают двух типов : MBR и GPT.

Структура MBR

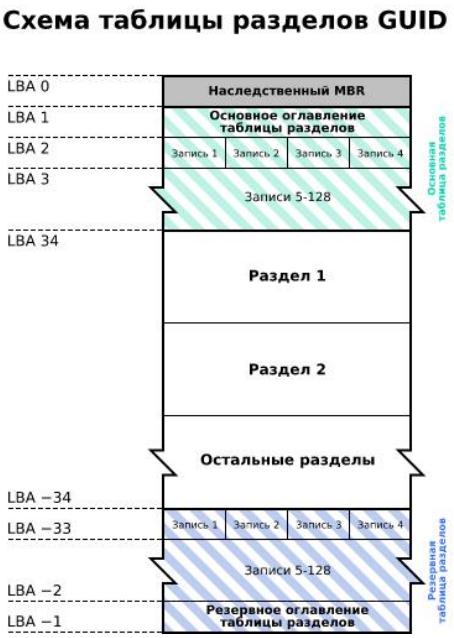
Первые 512 байт (первый сектор диска) главного устройства хранения данных занимает MBR (Master Boot Record). В состав MBR входит 446 байт кода загрузчика, четыре записи по 16 байт – это таблица разделов, 2 байта сигнатуры. Таблица разделов может состоять из первичных разделов (до 4) и логических разделов(до 128).

Структура GPT

GUID Partition Table, аббр. GPT – стандарт формата размещения таблиц разделов на физическом жестком диске. Он является частью расширяемого микропрограммного интерфейса (англ. Extensible Firmware Interface, EFI) – стандарта, предложенного Intel на смену BIOS. EFI использует GPT там, где BIOS использует главную загрузочную запись (англ. Master Boot Record, MBR). В GPT нет собственной программы-загрузчика, вместо этого он работает в паре с EFI. Внутри GPT используется адресация логических блоков LBA, которая абстрагирована от физики устройства (в отличие от CHS – «Цилиндр-Головка-Сектор»). Каждый логический блок занимает 512 байт. LBA 0 – первые 512 байт диска, LBA 1 – следующие, и так далее. Отрицательные значения LBA означают смещение в блоках с конца диска. Последний блок имеет смещение «-1» (LBA -1).

Структура

Название	Адрес	Описание
Наследственный MBR	LBA 0	Первые 512 байт диска отведены под "фейковый MBR". В нём из записей есть только индентификатор диска, стандартная сигнатура **0x55AA** в конце и единственный фейковый раздел типа **0xEE** (указание, что используется GPT), внутри которого находится настоящая разметка диска и все пользовательские данные. Остальное забито нулями, кода загрузчика нет. Наследственный MBR служит для предотвращения потери данных из-за программ, которые не понимают GPT.
Основная таблица разделов GPT	LBA 1	Оглавление таблицы разделов. Содержит GUID диска, адреса основной и резервной таблиц и данные о размере и количестве записей о разделах (стандартно — 128 штук). И контрольную сумму, которую проверяет EFI. "Благодаря" этой контрольной сумме **ручное редактирование разделов GPT невозможно**.
	LBA 2 ... LBA 33	Записи данных о разделах. Каждая запись занимает 128 байт, то есть в один LBA вмещается 4 записи. Первые 16 байт записи — GUID типа раздела, следующие 16 байт — его UUID, уникальный идентификатор, остальное место занимает информация о его границах и атрибутах.
Данные	LBA 34 ... LBA *	Собственно, содержимое разделов.
Резервная таблица разделов GPT	LBA -33 ... LBA -2	Полная копия описания разделов
	LBA -1	Полная копия оглавления.



На данный момент наиболее распространенной схемой разбиения дисков является MBR. Но с развитием средств хранения данных и их объемов, возможностей MBR становится недостаточно. Это связанно с невозможностью обеспечивать доступ к разделу диска емкостью более чем 2.2 TB. На сегодняшний день уже доступны диски емкостью более 6 TB, а так же, применяются различные технологии по объединению дисков в массивы, такие как RAID и LVM. Таким образом, применение схемы разбиения дисков на основе GPT становится все более актуальным.

Процесс загрузки

Процесс загрузки компьютера является многоступенчатым процессом, и начинается он с инициализации системных устройств набором микропрограмм, называемых BIOS (Basic Input/Output System), которые выполняются при старте системы. После того, как BIOS успешно проверит системные устройства, идет процесс поиска загрузчика в MBR устройств хранения (CD/DVD диски, USB диск, HDD, SSD и др.) или на первом разделе устройства. После того, как загрузчик получил управление, он получает таблицу разделов и готовит к загрузке операционную систему. В семействе загрузчиков GNU/Linux яркими представителями являются GRUB и LILO. В них MBR состоит из небольшой части ассемблерного кода. Стандартный

загрузчик Windows/DOS в состоянии проверить только активный раздел, считать несколько секторов с этого раздела и затем передать управление операционной системе. Он не в состоянии загрузить Linux, так как не наделен необходимым функционалом. GRand Unified Bootloader (GRUB) – это стандартный загрузчик для операционных систем семейства GNU/Linux, и всем пользователям рекомендуется по умолчанию установить его в MBR, для того чтобы иметь возможность загружать операционную систему с любого раздела, первичного или логического.

Пример работы с MBR

Существует специальный набор команд для работы с MBR. Так как он расположен на диске, то может быть сохранен и, в случае необходимости, восстановлен.

- dd if=/dev/sda of=/path/mbr-backup bs=512 count=1 – Для создания резервной копии MBR.
- dd if=/path/mbr-backup of=/dev/sda bs=512 count=1 – Для восстановления MBR.
- dd if=/dev/sda of=/path/mbr-boot-code bs=446 count=1 – Для сохранения только загрузочного кода.
- dd if=/dev/sda of=/path/mbr-part-table bs=1 count=66 skip=446 – Для сохранения только таблицы разделов.
- dd if=/path/mbr-backup of=/dev/sda bs=446 count=1 – Для восстановление загрузочного кода из файла mbr-backup.
- dd if=/path/mbr-backup of=/dev/sda bs=1 skip=446 seek=466 count=66 – Для восстановления только таблицы разделов.

dd if=/dev/zero of=/dev/sda bs=446 count=1 – Для очистки MBR, но при этом оставить таблицу разделов.

Задания к лабораторной работе

Часть 1

1. Войдите под терминалом в пользователя «sit».

2. Посмотрите какой идентификатор получил пользователь, используя команду id.

3. Посмотрите права доступа на домашний каталог, используя команду ls.

4. Создайте файл с маской 0077 используя umask.

5. Попробуйте прочитать его содержимое используя команду cat.

6. Запишите текстовую информацию в файл, используя консольный текстовый редактор vi или nano.

7. Проверьте права на файл, и прочитайте его содержимое.

8. Создайте каталог.

9. Установите права записи для группы пользователей на данный каталог.

10. Проверьте в какие группы входит пользователь.

11. Создайте несколько файлов в каталоге.

12. Ознакомьтесь как удалить пользователя вместе с содержимым его домашнего каталога из справочной документации.

13. Удалите пользователя вместе с его домашним каталогом.

Часть 2

14. Добавьте в виртуальную машину с операционной системой Linux виртуальный жесткий диск (делается это в настройках виртуальной машины).

15. Запустите виртуальную машину с операционной системой Linux.

16. Ознакомьтесь с командой fdisk и ее возможностями из справочной документации.

17. Создайте таблицу разделов (3 первичных и 1 логический) с помощью команды fdisk на **добавленном** виртуальном диске (обычно это диск /dev/sdb).

18. Запишите изменения на диск

19. Проверьте факт создания разделов используя команду fdisk. (Так же, создание разделов можно проверить используя команду ls /dev/sd*)

20. Отформатируйте созданные разделы в файловую систему ext4.

21. Ознакомьтесь с командами mount и umount и их возможностями из справочной документации.

22. Смонтируйте созданные разделы и создайте там произвольные файлы.

23. Сделайте резервную копию MBR с помощью утилиты DD.

24. Сотрите таблицу разделов MBR с помощью утилиты DD.

25. Восстановите MBR с помощью утилиты DD.

26. Смонтируйте разделы и проверьте целостность данных.

27. Отмонтируйте разделы.

Вопросы к лабораторной работе

Часть 1

1. Какой uid у пользователя «sit»? В какие группы он входит?

2. Почему попытка удалить пользователя не удалась, и что нужно сделать для его удаления?

3. Какие права доступа установлены на домашний каталог пользователя «sit»?

4. Как рекурсивно изменить права доступа на файлы в каталоге?

5. Как можно осуществлять переключение между пользователями в рамках одного терминала?

6. Как удалить пользователя при этом сохранив его домашний каталог и данные внутри него?

7. Какое значение umask нужно установить, чтобы владелец и группа имели право на чтение, запись и исполнение, а все остальные пользователи не имели никаких прав?

8. Как рекурсивно снять все suid биты с файлов в каталоге?

9. Как разрешить программе (файлу) исполняться?

10. Что такое бит sticky? Для чего он предназначен?

11. Зачем нужны uid и gid?

12. Почему uid пользователя задается больше 1000?

Часть 2

13. Что записано в первом секторе главной загрузочной записи MBR?

14. Функциональное назначение MBR и GPT?

15. Структура GPT.

16. Какое максимальное количество первичных разделов можно создать при использовании таблицы разделов MBR?

17. Какое максимальное количество первичных разделов можно создать при использовании таблицы разделов GPT?

18. Как сохранить информацию о структуре MBR?

19. Как создать 10 разделов с файловой системой ext3 на диске в таблице разделов MBR?

20. Как стереть код загрузчика в MBR?

21. Как можно смонтировать раздел диска с файловой системой в режиме только для чтения?

22. Как можно осуществить восстановление GPT разделов в случае сбоев?