

Отчет по лабораторной работе 5

**Дискреционное разграничение прав в Linux. Исследование влияния
дополнительных атрибутов**

Шалыгин Георгий Эдуардович

Содержание

1	Цель работы	5
2	Теоретическое введение	6
2.0.1	Изменение владельца	7
2.1	Использование chmod	7
3	Выполнение лабораторной работы	9
4	Выводы	15
	Список литературы	16

Список иллюстраций

3.1	simpleid	9
3.2	run simpleid	9
3.3	id	9
3.4	simpleid2	10
3.5	simpleid2 with u+s	10
3.6	readfile.c	11
3.7	Неудача	11
3.8	Проверка прав доступа для новых атрибутов	11
3.9	run readfile	12
3.10	etc/shadow	12
3.11	просмотр атрибутов	12
3.12	просмотр атрибутов	13
3.13	проверка доступа	13
3.14	проверка доступа	13
3.15	удаление стики-бита	14
3.16	атрибуты	14
3.17	проверка доступа	14

Список таблиц

1 Цель работы

Изучение механизмов изменения идентификаторов, применения SetUID- и Sticky-битов. Получение практических навыков работы в консоли с дополнительными атрибутами. Рассмотрение работы механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

2 Теоретическое введение

В Linux, как и в любой многопользовательской системе, абсолютно естественным образом возникает задача разграничения доступа субъектов — пользователей к объектам — файлам дерева каталогов.

Один из подходов к разграничению доступа — так называемый дискреционный (от англ. *discretion* — чье-либо усмотрение) — предполагает назначение владельцев объектов, которые по собственному усмотрению определяют права доступа субъектов (других пользователей) к объектам (файлам), которыми владеют.

Дискреционные механизмы разграничения доступа используются для разграничения прав доступа процессов как обычных пользователей, так и для ограничения прав системных программ в (например, служб операционной системы), которые работают от лица псевдопользовательских учетных записей.

В Linux у каждого файла и каждого каталога есть два владельца: пользователь и группа.

Эти владельцы устанавливаются при создании файла или каталога. Пользователь, который создаёт файл становится владельцем этого файла, а первичная группа, в которую входит этот же пользователь, так же становится владельцем этого файла. Чтобы определить, есть ли у вас как у пользователя права доступа к файлу или каталогу, оболочка проверяет владение ими.

Это происходит в следующем порядке:

1. Оболочка проверяет, являетесь ли вы владельцем файла, к которому вы хотите получить доступ. Если вы являетесь этим владельцем, вы получаете разрешения и оболочка прекращает проверку.

2. Если вы не являетесь владельцем файла, оболочка проверит, являетесь ли вы участником группы, у которой есть разрешения на этот файл. Если вы являетесь участником этой группы, вы получаете доступ к файлу с разрешениями, которые для группы установлены, и оболочка прекратит проверку.
3. Если вы не являетесь ни пользователем, ни владельцем группы, вы получаете права других пользователей (Other).

Чтобы увидеть текущие назначения владельца, вы можете использовать команду **ls -l**. Эта команда показывает пользователя и группу-владельца.

Подробнее в [1].

2.0.1 Изменение владельца

Чтобы применить соответствующие разрешения, первое, что нужно учитывать, это владение. Для этого есть команда **chown**. Синтаксис этой команды несложен для понимания:

```
chown кто что
```

Например, следующая команда меняет владельца каталога `/home/account` на пользователя `linda`:

```
chown linda /home/account
```

2.1 Использование **chmod**

Для управления правами используется команда **chmod**. При использовании **chmod** вы можете устанавливать разрешения для пользователя (user), группы (group) и других (other). Вы можете использовать эту команду в двух режимах: относительный режим и абсолютный режим. В абсолютном режиме три цифры используются для установки основных разрешений.

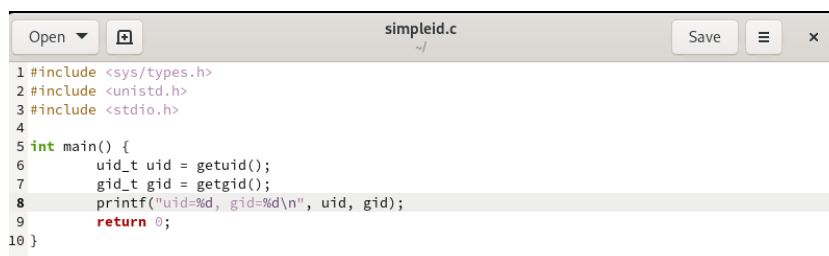
При настройке разрешений рассчитайте необходимое вам значение. Если вы хотите установить чтение, запись и выполнение для пользователя, чтение и выполнение для группы, а также чтение и выполнение для других в файле /somefile, то вы используете следующую команду **chmod**:

```
chmod 755 /somefile
```

Подробнее в [2].

3 Выполнение лабораторной работы

1. От имени пользователя guest2 создадим программу simpleid.c: (fig. 3.1).

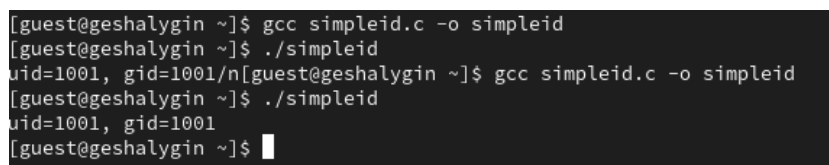


```
1 #include <sys/types.h>
2 #include <unistd.h>
3 #include <stdio.h>
4
5 int main() {
6     uid_t uid = getuid();
7     gid_t gid = getgid();
8     printf("uid=%d, gid=%d\n", uid, gid);
9     return 0;
10 }
```

Рис. 3.1: simpleid

2. Скомпилируем программу, что файл программы создан: gcc simpleid.c -o simpleid (fig:002).

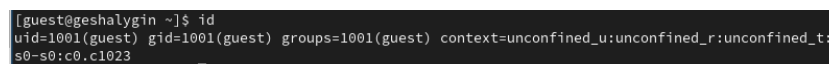
Выполним программу simpleid и id, в результате получаем правильные значения (fig:002).



```
[guest@geshalygin ~]$ gcc simpleid.c -o simpleid
[guest@geshalygin ~]$ ./simpleid
uid=1001, gid=1001/n[guest@geshalygin ~]$ gcc simpleid.c -o simpleid
[guest@geshalygin ~]$ ./simpleid
uid=1001, gid=1001
[guest@geshalygin ~]$
```

Рис. 3.2: run simpleid

3. Сравнение вывода с id (fig. 3.3).



```
[guest@geshalygin ~]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:
s0-s0:c0.c1023
```

Рис. 3.3: id

4. Усложним программу, добавив вывод действительных идентификаторов.

Скомпилируем и запустим simpleid2.c (fig. 3.4).

```
[guest@geshalygin ~]$ gcc simpleid.c -o simpleid2
[guest@geshalygin ~]$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@geshalygin ~]$
```

Рис. 3.4: simpleid2

5. От имени суперпользователя выполним команды (fig. 3.5). Изменим владельца и добавим uid бит.

Выполним проверку правильности установки новых атрибутов и смены владельца файла simpleid2: `ls -l simpleid2 11`.

Запустите simpleid2 и id: `./simpleid2 id`

Результаты совпадают.

```
[root@geshalygin guest]# chown root:guest /home/guest/simpleid
[root@geshalygin guest]# chmod u+s /home/guest/simpleid2
[root@geshalygin guest]# ls -l simpleid2
-rwsr-xr-x. 1 guest guest 26064 Oct  5 01:08 simpleid2
[root@geshalygin guest]# ./simpleid2
e_uid=1001, e_gid=0
real_uid=0, real_gid=0
[root@geshalygin guest]# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@geshalygin guest]#
```

Рис. 3.5: simpleid2 with u+s

6. Создадим программу readfile.c: (fig. 3.6).

```

1 #include <fcntl.h>
2 #include <stdio.h>
3 #include <sys/stat.h>
4 #include <unistd.h>
5
6 int main(int argc, char* argv[]) {
7     unsigned char buffer[16];
8     size_t bytes_read;
9     int i;
10
11     int fd = open(argv[1], O_RDONLY);
12     do {
13         bytes_read = read(fd, buffer, sizeof(buffer));
14         for (i=0; i<bytes_read;i++) printf("%c", buffer[i]);
15     }
16     while (bytes_read == sizeof(buffer));
17     close(fd);
18     return 0;
19 }

```

Рис. 3.6: readfile.c

7. Скомпилируем (fig. 3.7).

```

[root@geshalygin guest]# gcc readfile.c -o readfile
[root@geshalygin guest]# chown root:root readfile
[root@geshalygin guest]# chmod 400 readfile

```

Рис. 3.7: Неудача

8. Сменим владельца у файла readfile.c и права так, чтобы только суперпользователь (root) мог прочитать его, а guest не мог. (fig. 3.8). Проверим что guest не имеет доступ к файлу.

```

[root@geshalygin guest]# chown root:root readfile.
[root@geshalygin guest]# chmod 400 readfile.c
[root@geshalygin guest]# su guest
[guest@geshalygin ~]$ cat readfile.c
cat: readfile.c: Permission denied

```

Рис. 3.8: Проверка прав доступа для новых атрибутов

9. Сменим у программы readfile владельца и установите SetU'D-бит. Проверим, может ли программа readfile прочитать файл readfile.c? Как видим, может(fig. 3.9).

```
[guest@gshalygin ~]$ gcc readfile.c -o readfile  
[guest@gshalygin ~]$ su  
Password:  
[root@gshalygin guest]# chown root:root readfile.c  
[root@gshalygin guest]# chmod 400 readfile.c  
[root@gshalygin guest]# su guest  
[guest@gshalygin ~]$ chmod u+s readfile  
[guest@gshalygin ~]$ ./readfile readfile.c  
cSHELL=/bin/bashSESSION_MANAGER=local/unix:/tmp/.ICE-unix/2844,unix/user:/tmp/.ICE-unix/2844COLORTERM=truecolorHISTCONTROL=ignoredupsXDG_MEN_PREFIX=gnome-HISTSIZE=1000HOSTNAME=gshalygin.localdomainSSH_AUTH_SOCK=/run/user/1090/keyring/sshXMODIFIERS=@im=ibusDESKTOP_SESSION=gnomePWD=/home/guestLOGNAME=guestXDG_SESSION_DESKTOP=gnomeXDG_SESSION_TYPE=waylandSYSTEMD_EXEC_PTD=2862GDM_LANG=en_US.UTF-8HOME=/home/guestUSER=gshalyginLANG=en_US.UTF-8_COLORS=rs=0:di=01;34;ln=01;36;mh=00;pi=40;33:sf=01;35;do=01;35;bd=04;33;oi=cd=04;33;o=1;r=04;31;l=mi=01;37;4l=su=37;4j=sg=30;43;ca=30;41;tw=30;42;ow=34;42;st=37;44;ex=01;32;*tar=01;31*;tgz=01;31*;arc=01;31*;arj=01;31*;tar.gz=01;31*;lha=01;31*;Lzh=01;31*;lzma=01;31*;tlz=01;31*;txz=01;31*;tzo=01;31*;tTz=01;31*;zip=01;31*;z=01;31*;dz=01;31*;gz=01;31*;lrz=01;31*;lz=01;31*;lzo=01;31*;xz=01;31*;xzt=01;31*;zst=01;31*;bz2=01;31*;bz=01;31*;tbz=01;31*;deb=01;31*;rpm=01;31*;jar=01;31*;war=01;31*;ear=01;31*;sar=01;31*;rar=01;31*;alz=01;31*;ace=01;31*;zoo=01;31*;cptio=01;31*;7z=01;31*;rzip=01;31*;cab=01;31*;wim=01;31*;swm=01;31*;dwm=01;31*;esd=01;31*;jpg=01;35*;jpeg=01;35*;mjpeg=01;35*
```

Рис. 3.9: run readfile

10. Проверим, может ли программа readfile прочитать файл /etc/shadow (fig. 3.10).

[illegible]

Рис. 3.10: etc/shadow

11. Выясним, установлен ли атрибут Sticky на директории /tmp (fig. 3.11).

```
[guest@geshalygin ~]$ ls -l / | grep tmp
drwxrwxrwt. 16 root root 4096 Oct  5 06:46 tmp
```

Рис. 3.11: просмотр атрибутов

12. От имени пользователя guest создадим файл file01.txt в директории /tmp

со словом test. Просмотрим атрибуты у только что созданного файла и разрешим чтение и запись для категории пользователей «все остальные» (fig. 3.12).

```
[guest@geshalygin ~]$ echo "test"> /tmp/file01.txt
[guest@geshalygin ~]$ ls -l tmp/file01.txt
ls: cannot access 'tmp/file01.txt': No such file or directory
[guest@geshalygin ~]$ ls -l /tmp/file01.txt
-rw-r--r--. 1 guest guest 5 Oct  5 06:48 /tmp/file01.txt
[guest@geshalygin ~]$ chmod o+rw /tmp/file01.txt
[guest@geshalygin ~]$ ls -l /tmp/file01.txt
-rw-r--rw-. 1 guest guest 5 Oct  5 06:48 /tmp/file01.txt
[guest@geshalygin ~]$
```

Рис. 3.12: просмотр атрибутов

13. От пользователя guest2 (не являющегося владельцем) попробуем прочитав файл /tmp/file01.txt. Доступ открыт. Дозаписать в файл уже нельзя (fig. 3.13).

```
[guest@geshalygin ~]$ su guest2
Password:
[guest2@geshalygin guest]$ cat /tmp/file01.txt
test
[guest2@geshalygin guest]$ echo "test2" > /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@geshalygin guest]$
```

Рис. 3.13: проверка доступа

14. Проверьте содержимое файла. Дозапись и удаление невозможны (fig. 3.14).

```
[guest2@geshalygin guest]$ cat /tmp/file01.txt
test
[guest2@geshalygin guest]$ rm /tmp/file01.txt
rm: remove write-protected regular file '/tmp/file01.txt'? y
rm: cannot remove '/tmp/file01.txt': Operation not permitted
[guest2@geshalygin guest]$
```

Рис. 3.14: проверка доступа

15. Выполним команду, снимающую sticky-бит (fig. 3.15).

```
[guest2@geshalygin guest]$
[guest2@geshalygin guest]$ su -
Password:
[root@geshalygin ~]# chmod -t /tmp
[root@geshalygin ~]# exit
logout
[guest2@geshalygin guest]$
```

Рис. 3.15: удаление стики-бита

16. Выведем атрибуты (fig. 3.16).

```
[root@geshalygin ~]# chmod -t /tmp
[root@geshalygin ~]# exit
logout
[guest2@geshalygin guest]$ ls -l / | grep tmp
drwxrwxrwx. 16 root root 4096 Oct  5 06:52 tmp
[guest2@geshalygin guest]$
```

Рис. 3.16: атрибуты

17. Теперь дозапись недоступна, удаление доступно (fig. 3.17).

```
[guest2@geshalygin guest]$ echo "test2" > /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@geshalygin guest]$ cat /tmp/file01.txt
test
[guest2@geshalygin guest]$ rm /tmp/file01.txt
rm: remove write-protected regular file '/tmp/file01.txt'? y
[guest2@geshalygin guest]$ ls /tmp
dbus-xmABtX2BT4
systemd-private-6e0e6cdf6abd4005b5b03329d23d2a3c-chrond.service-iMqU2F
systemd-private-6e0e6cdf6abd4005b5b03329d23d2a3c-colord.service-8XxdJc
systemd-private-6e0e6cdf6abd4005b5b03329d23d2a3c-dbus-broker.service-ZPz3x5
systemd-private-6e0e6cdf6abd4005b5b03329d23d2a3c-geoclue.service-TEaNiC
systemd-private-6e0e6cdf6abd4005b5b03329d23d2a3c-ModemManager.service-3C0xr2
systemd-private-6e0e6cdf6abd4005b5b03329d23d2a3c-power-profiles-daemon.service-oiuDDj
systemd-private-6e0e6cdf6abd4005b5b03329d23d2a3c-rtkit-daemon.service-YbtxPg
systemd-private-6e0e6cdf6abd4005b5b03329d23d2a3c-switcheroo-control.service-30Xgvt
systemd-private-6e0e6cdf6abd4005b5b03329d23d2a3c-systemd-logind.service-Xc168Z
systemd-private-6e0e6cdf6abd4005b5b03329d23d2a3c-upower.service-ySHieR
[guest2@geshalygin guest]$
```

Рис. 3.17: проверка доступа

4 Выводы

В результате выполнения работы мы изучили механизмы изменения идентификаторов, применения SetUID- и Sticky-битов. Получили практические навыки работы в консоли с дополнительными атрибутами. Рассмотрели работу механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

Список литературы

1. Кетов Д.В. Внутреннее устройство Linux. BHV, 2017. 124 с.
2. Л. М. Ухлинов. Управление доступом в ОС GNU /Linux . ОКБ САПР», Москва, Россия, 2010.