

# Основы информационной безопасности. Лабораторная работа № 5

Дискреционное разграничение прав в Linux. Исследование влияния дополнительных атрибутов

---

Сунгурова Мариян М.

05.10.2024

Российский Университет дружбы народов

## Информация

---

- Сунгурова Мариян М.
- студентка группы НКНбд-01-21
- Российский университет дружбы народов

## Вводная часть

---

Целью данной лабораторной работы является изучение механизмов изменения идентификаторов, применения SetUID- и Sticky-битов, а также получение практических навыков работы в консоли с дополнительными атрибутами и рассмотрение работы механизма смены идентификатора процессов пользователей, и влияние бита Sticky на запись и удаление файлов.

При работе с командой `chmod` важно понимать основные права доступа, которые назначают файлам или каталогам. В Linux используется три основных типа прав доступа:

- Чтение (Read) — обозначается буквой «r». Предоставляет возможность просматривать содержимое файла или каталога.
- Запись (Write) — обозначается буквой «w». Позволяет создавать, изменять и удалять файлы внутри каталога, а также изменять содержимое файла.
- Выполнение (Execute) — обозначается буквой «x». Дает разрешение на выполнение файла или на вход в каталог.

Каждый из указанных выше типов прав доступа может быть назначен трем группам пользователей:

- Владелец (Owner) — пользователь, который является владельцем файла или каталога.
- Группа (Group) — группа пользователей, к которой принадлежит файл или каталог.
- Остальные пользователи (Others) — все остальные пользователи системы.

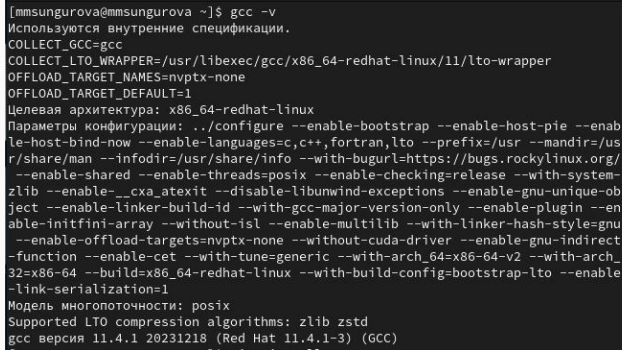
Комбинация этих базовых прав доступа для каждой из групп пользователей определяет полный набор прав доступа для файла или каталога.

## Выполнение лабораторной работы

---



Проверим установлен ли компилятор gcc, а также отключим SELinux(рис. (fig:001?))



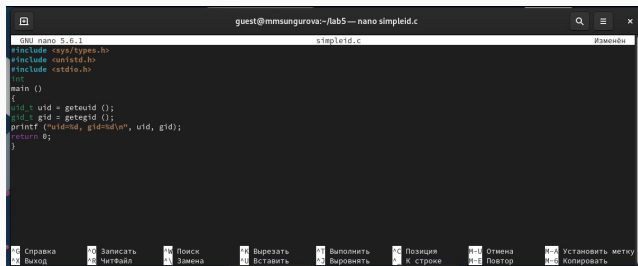
```
[mmsungurova@mmsungurova ~]$ gcc -v
Используются внутренние спецификации.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/libexec/gcc/x86_64-redhat-linux/11/lto-wrapper
OFFLOAD_TARGET_NAMES=nvptx-none
OFFLOAD_TARGET_DEFAULT=1
Целевая архитектура: x86_64-redhat-linux
Параметры конфигурации: ../configure --enable-bootstrap --enable-host-pie --enable-host-bind-now --enable-languages=c,c++,fortran,lto --prefix=/usr --mandir=/usr/share/man --infodir=/usr/share/info --with-bugurl=https://bugs.rockylinux.org/ --enable-shared --enable-threads=posix --enable-checking=release --with-system-zlib --enable-__cxa_atexit --disable-libunwind-exceptions --enable-gnu-unique-object --enable-linker-build-id --with-gcc-major-version-only --enable-plugin --enable-initfini-array --without-isl --enable-multilib --with-linker-hash-style=gnu --enable-offload-targets=nvptx-none --without-cuda-driver --enable-gnu-indirect-function --enable-cet --with-tune=generic --with-arch_64=x86-64-v2 --with-arch_32=x86-64 --build=x86_64-redhat-linux --with-build-config=bootstrap-lto --enable-link-serialization=1
Модель многопоточности: posix
Supported LTO compression algorithms: zlib zstd
gcc версия 11.4.1 20231218 (Red Hat 11.4.1-3) (GCC)
```

Рис. 1: Подготовка лабораторного стенда

```
[mmsungurova@mmsungurova ~]$ whereis gcc
gcc: /usr/bin/gcc /usr/lib/gcc /usr/libexec/gcc /usr/share/man/man1/gcc.1.gz /usr/share/info/gcc.info.gz
[mmsungurova@mmsungurova ~]$ whereis g++
g++: /usr/bin/g++ /usr/share/man/man1/g++.1.gz
[mmsungurova@mmsungurova ~]$
```

Рис. 2: Подготовка лабораторного стенда

Войдем в систему от имени пользователя guest и создадим программу simpleid.c, которая выводит идентификатор пользователя и группы(рис. (fig:002?))



```
GNU nano 5.6.1 simpleid.c
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
int
main ()
{
uid_t uid = getuid ();
gid_t gid = getgid ();
printf ("uid=%d, gid=%d\n", uid, gid);
return 0;
}
```

Справка    Записать    Поиск    Вырезать    Выполнить    Позиция    Отмена    Установить метку  
Выход    Читфайл    Замена    Вставить    Выровнять    к строке    Повтор    И-д копировать

Рис. 3: Текст программы simpleid.c

Теперь скомпилируем программу с помощью gcc, затем, запустив её, увидим, что она выводит идентификаторы пользователя и группы 1001 и 1001 для guest, что совпадает с выводом команды id (рис. (fig:003?))

```
[guest@msungurova lab5]$ nano simpleid.c
[guest@msungurova lab5]$ gcc simpleid.c -o simpleid
[guest@msungurova lab5]$ ls
simpleid  simpleid.c
[guest@msungurova lab5]$ ./simpleid
uid=1001, gid=1001
[guest@msungurova lab5]$ id
uid=1001(guest) gid=1001(guest) rpynm=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-c0.c1023
[guest@msungurova lab5]$
```

Рис. 4: Запуск программы simpleid

Усложним программу, добавив вывод действительных идентификаторов(рис. (fig:004?)).



```
guest@mmsungurova:~/.lab5 — nano simpleid2.c
GNU nano 5.6.1 simpleid2.c
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
int
main ()
{
    uid_t real_uid = getuid ();
    uid_t e_uid = geteuid ();
    gid_t real_gid = getgid ();
    gid_t e_gid = getegid ();
    printf ("e_uid=%d, e_gid=%d\n", e_uid, e_gid);
    printf ("real_uid=%d, real_gid=%d\n", real_uid,
    ,+ real_gid);
    return 0;
}
```

Bottom status bar: Ctrl-S Справка, Ctrl-X Выход, Ctrl-O Записать, Ctrl-F Чить файл, Ctrl-W Поиск, Ctrl-R Замена, Ctrl-D Вырезать, Ctrl-V Вставить, Ctrl-E Выполнить, Ctrl-Y Выводить, Ctrl-L Позиция, Ctrl-K К строке, Ctrl-U Отмена, Ctrl-\_ Повтор, Ctrl-I Установить метку, Ctrl-C Копировать

Рис. 5: Текст программы simpleid2.c

Теперь скомпилируем программу с помощью gcc, затем, запустив её, увидим, что она выводит идентификаторы пользователя и группы 1001 и 1001 для guest, что совпадает с выводом команды id(рис. (fig:005?)).

```
[guest@mmsungurova lab5]$ nano simpleid2.c
[guest@mmsungurova lab5]$ gcc simpleid2.c -o simpleid2
[guest@mmsungurova lab5]$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@mmsungurova lab5]$
```

Рис. 6: Запуск программы simpleid2

От имени суперпользователя изменим владельца файла /home/guest/simpleid2 и установим SetUID-бит. Проверим корректность установленных прав и опять запустим simpleid2(рис. (fig:006?)).

```
[guest@mmsungurova lab5]$ ls -l
итого 56
-rwxr-xr-x. 1 guest guest 24384 окт  5 13:48 simpleid
-rwsr-xr-x. 1 guest guest 24488 окт  5 13:53 simpleid2
-rw-r--r--. 1 guest guest   302 окт  5 13:53 simpleid2.c
-rw-r--r--. 1 guest guest   175 окт  5 13:48 simpleid.c
[guest@mmsungurova lab5]$
```

Рис. 7: Изменение владельца и запуск программы simpleid2 с установленным SetUID-битом

Прделаем аналогичные действия относительно SetGID-бита(рис. (fig:007?)):

```
[guest@mmsungurova lab5]$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@mmsungurova lab5]$ id
uid=1001(guest) gid=1001(guest) rpyны=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@mmsungurova lab5]$
```

Рис. 8: Запуск программы simpleid2 с установленным SetGID-битом



## Выполнение лабораторной работы

Создадим программу для чтения файлов readfile.c(рис. (fig:008?)):

```
[root@mmsungurova lab5]# chown root:guest readfile.c
[root@mmsungurova lab5]# chmod 700 readfile.c
[root@mmsungurova lab5]# cat readfile.c
#include <fcntl.h>
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
int
main (int argc, char* argv[])
{
    unsigned char buffer[16];
    size_t bytes_read;
    int i;
    int fd = open (argv[1], O_RDONLY);
    do
    {
        bytes_read = read (fd, buffer, sizeof (buffer));
        for (i =0; i < bytes_read; ++i) printf("%c", buffer[i]);
    }
    while (bytes_read == sizeof (buffer));
    close (fd);
    return 0;
}
[root@mmsungurova lab5]# exit
Выход
[guest@mmsungurova lab5]$ ls
readfile readfile.c simpleid simpleid2 simpleid2.c simpleid.c
[guest@mmsungurova lab5]$ cat readfile.c
cat: readfile.c: Отказано в доступе
```

Скомпилируем её и сменим владельца у файла с текстом программы, затем изменим права так, чтобы только суперпользователь (root) мог прочитать его, и проверим корректность настроек(рис. (fig:008?)):

## Выполнение лабораторной работы

Сменим у программы readfile владельца и установим SetUID-бит. Теперь эта программа может прочитать файл readfile.c даже с пользователя guest, также она может прочитать файл /etc/shadow, владельцем которого guest также не является, так как программа readfile теперь имеет все права пользователя root(рис. (fig:010?)):

```
[guest@mmsungurova lab5]$ su -
Пароль:
[root@mmsungurova ~]# ls
anaconda-ks.cfg
[root@mmsungurova ~]# cd /home/guest/lab5/
[root@mmsungurova lab5]# chown root:guest readfile
[root@mmsungurova lab5]# chmod u+s readfile
[root@mmsungurova lab5]# exit
выход
[guest@mmsungurova lab5]$ ./readfile readfile.c
#include <fcntl.h>
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
int
main (int argc, char* argv[])
{
    unsigned char buffer[16];
    size_t bytes_read;
    int i;
    int fd = open (argv[1], O_RDONLY);
    do
    {
        bytes_read = read (fd, buffer, sizeof (buffer));
        for (i = 0; i < bytes_read; ++i) printf("%c", buffer[i]);
    }
    while (bytes_read == sizeof (buffer));
    close (fd);
    return 0;
}
[guest@mmsungurova lab5]$ ./readfile /etc/shadow
root:$6$1jG1/SvuBza3nXaR$57dCtb4.IE1fdTPQp1Dx9q1f6vxxb6hFIXCnVg5NpLZJkwsyqV2WlFa/BuwCDYzHVk8nEP935EjRb2FnabH00::0:99999:7:::
bin:*:19820:0:99999:7:::
daemon:*:19820:0:99999:7:::
adm:*:19820:0:99999:7:::
lp:*:19820:0:99999:7:::
sync:*:19820:0:99999:7:::
nobody:*:19820:0:99999:7:::
```

Проверим, что установлен атрибут Sticky на директории /tmp(в конце стоит t). Затем от имени пользователя guest создадим файл file01.txt в директории /tmp со словом test, затем посмотрим атрибуты у только что созданного файла и разрешим чтение и запись для категории пользователей «все остальные». После этого от пользователя guest2 попробуем дозаписать в этот файл новое слово, однако получим отказ, также нам отказано в перезаписи и удалении этого файла. Если же убрать Sticky бит, то нам будет разрешено удаление этого файла(рис. (fig:011?))

# Выполнение лабораторной работы

```
[guest@mmsungurova lab5]$ ls -l / | grep tmp
drwxrwxrwt. 16 root root 4096 окт  5 14:10 tmp
[guest@mmsungurova lab5]$ echo "test" > /tmp/file01.txt
[guest@mmsungurova lab5]$ ls -l /tmp/file01.txt
-rw-r--r--. 1 guest guest 5 окт  5 14:13 /tmp/file01.txt
[guest@mmsungurova lab5]$ chmod o+rw /tmp/file01.txt
[guest@mmsungurova lab5]$ ls -l /tmp/file01.txt
-rw-r--r--. 1 guest guest 5 окт  5 14:13 /tmp/file01.txt
[guest@mmsungurova lab5]$ su guest2
Пароль:
[guest2@mmsungurova lab5]$ cat /tmp/file01.txt
test
[guest2@mmsungurova lab5]$ echo "test2" > /tmp/file01.txt
bash: /tmp/file01.txt: Отказано в доступе
[guest2@mmsungurova lab5]$ cat /tmp/file01.txt
test
[guest2@mmsungurova lab5]$ echo "test3" > /tmp/file01.txt
bash: /tmp/file01.txt: Отказано в доступе
[guest2@mmsungurova lab5]$ cat /tmp/file01.txt
test
[guest2@mmsungurova lab5]$ rm /tmp/file01.txt
rm: невозможно удалить '/tmp/file01.txt': Нет такого файла или каталога
[guest2@mmsungurova lab5]$ su -
Пароль:
[root@mmsungurova ~]# chmod -t /tmp
[root@mmsungurova ~]# exit
Выход
[guest2@mmsungurova lab5]$ ls -l / | grep tmp
drwxrwxrwx. 16 root root 4096 окт  5 14:17 tmp
[guest2@mmsungurova lab5]$ cat /tmp/file01.txt
test
[guest2@mmsungurova lab5]$ echo "test2" > /tmp/file01.txt
bash: /tmp/file01.txt: Отказано в доступе
[guest2@mmsungurova lab5]$ cat /tmp/file01.txt
test
[guest2@mmsungurova lab5]$ echo "test3" > /tmp/file01.txt
bash: /tmp/file01.txt: Отказано в доступе
[guest2@mmsungurova lab5]$ cat /tmp/file01.txt
test
[guest2@mmsungurova lab5]$ rm /tmp/file01.txt
rm: невозможно удалить '/tmp/file01.txt': Нет такого файла или каталога
[guest2@mmsungurova lab5]$ su -
Пароль:
[root@mmsungurova ~]# chmod +t /tmp
[root@mmsungurova ~]# exit
Выход
```

## Выводы

---

В результате выполнения данной лабораторной работы были рассмотрены:

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