# Лабораторная работа No 5.

Королёв Иван

# Содержание

# Список иллюстраций

# 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 Выполнение лабораторной работы

## 2.1 SetUID бит

1. Из-под пользователя `guest` создадим файл `simpleid.c`

```c
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

int main () {
        uid_t uid = geteuid();
        gid_t gid = getegid();
        printf ("uid=%d, gid=%d\n", uid, gid);
        return 0;
}
```

Рис. 2.1: Исходный код файла simpleid.c

2. Скомпилируем, запустим и сравним результат с выводом команды `id`. User ID и Group ID совпадают.

```
[guest@batagiev ~]$ gcc simpleid.c -o simpleid
[guest@batagiev ~]$ ./simpleid
uid=1001, gid=1002
[guest@batagiev ~]$ id
uid=1001(guest) gid=1002(guest) groups=1002(guest)
:unconfined_t:s0-s0:c0.c1023
[guest@batagiev ~]$
```

Рис. 2.2: Результат

3. Перепишем программу, чтобы она возвращала нам `e_uid`/`e_gid` и действительный `uid`/`gid`. `getgid` возвращает действительный идентификатор группы текущего процесса. `getegid` возвращает эффективный идентификатор группы текущего процесса. Действительный идентификатор соответствует идентификатору вызывающего процесса. Эффективный идентификатор соответствует биту `setuid` на исполняемом файле.

```c
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, gid=%d\n", e_uid, gid);
        printf("real_uid=%d, real_gid=%d\n", real_uid, real_gid);
        return 0;
}
```

Рис. 2.3: Измененный исходный код программы simpleid.c

4. Поменяем пользователя-обладателя исполняемого файла и добавить бит `setuid`.

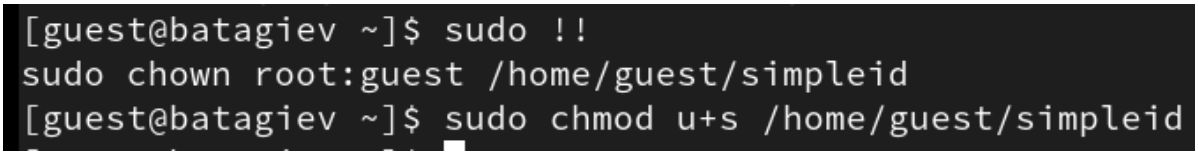


Рис. 2.4: Изменение прав

5. Проверим изменения.

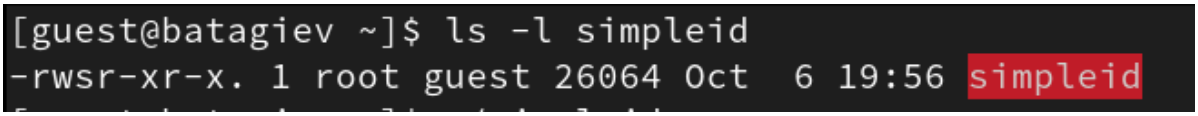


Рис. 2.5: ls -l

6. Запустим исполняемый файл.

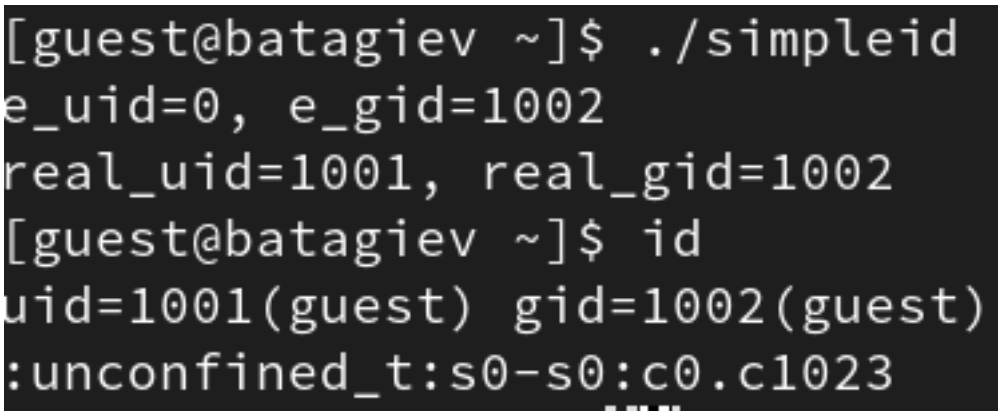


Рис. 2.6: Результат

7. Проведем те же манипуляции над файлов, но теперь для группы.

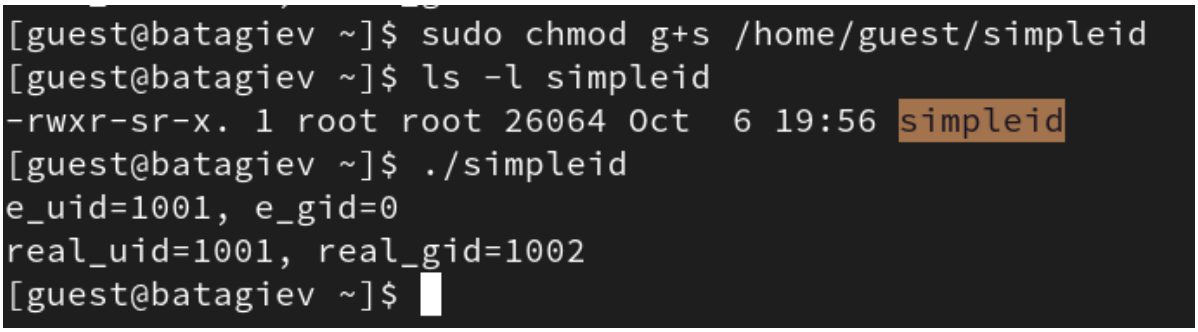


Рис. 2.7: Результат

8. Создадим программу `readfile.c`

```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;
}
```

Рис. 2.8: Исходный код файла readfile.c

9. Изменим права для файла `readfile.c`. Проверим изменения.

```
[guest@batagiev ~]$ sudo chown root:root readfile.c
[guest@batagiev ~]$ sudo chmod 700 readfile.c
[guest@batagiev ~]$ cat readfile.c
cat: readfile.c: Permission denied
[guest@batagiev ~]$
```

Рис. 2.9: Измение прав файла readfile.c

10. Сменим владельца файла `readfile`, а также добавим `setuid` бит.

```
[guest@batagiev ~]$ sudo chown root:guest ./readfile
[guest@batagiev ~]$ sudo chmod u+s ./readfile
[guest@batagiev ~]$
```

Рис. 2.10: владелец и setuid бит

11. Проверим чтения разных файлов. Например `readfile.c` и `/etc/shadow`.

```
[guest@batagiev ~]$ ./readfile ./readfile.c
#include <fcntl.h>
#include <stdio.h>
```

Рис. 2.11: readfile.c

```
[guest@batagiev ~]$ ./readfile /etc/shadow
root:$6$l0UHCDykMrMqVr5v$7LfXbb7V8ZOiTqEaEuYSO
y/ZZJAAZoWAeGzIFKYqcqjlYRk/::0:99999:7:::
bin:*:19469:0:99999:7:::
```

Рис. 2.12: /etc/shadow

## 2.2 Sticky бит

1. Проверим установлен ли атрибут Sticky на директорию `/tmp`.

```
[guest@batagiev ~]$ ls -l / | grep tmp
drwxrwxrwt.  15 root root 4096 Oct  6 20:07 tmp
```

Рис. 2.13: /tmp

2. Создадим файл `/tmp/file01.txt`. Добавим для остальных пользователей права на чтение и запись.

```
[guest@batagiev ~]$ echo "test" > /tmp/file01.txt
[guest@batagiev ~]$ ls -l /tmp/file01.txt
-rw-r--r--. 1 guest guest 5 Oct  6 20:14 /tmp/file01.txt
[guest@batagiev ~]$ chmod o+rw /tmp/file01.txt
[guest@batagiev ~]$ ls -l /tmp/file01.txt
-rw-r--rw-. 1 guest guest 5 Oct  6 20:14 /tmp/file01.txt
```

Рис. 2.14: /tmp/file01.txt

3. Попробуем прочитать содержимое файла от пользователя `guest2`.

```
[guest2@batagiev ~]$ cat /tmp/file01.txt
test
```

Рис. 2.15: guest2

4. Допишем в конец файла новый текст.

```
[guest2@batagiev ~]$ echo "test2" >> /tmp/file01.txt
-bash: /tmp/file01.txt: Permission denied
[guest2@batagiev ~]$
```

Рис. 2.16: guest2

5. А также удалим этот файл.

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

Рис. 2.17: guest2

6. Удалим sticky бит директории `/tmp`.

```
[guest2@batagiev ~]$ su -
Password:
[root@batagiev ~]# chmod -t /tmp/
[root@batagiev ~]#
```

Рис. 2.18: /tmp

7. Повторим предыдущие действия по изменению файла.

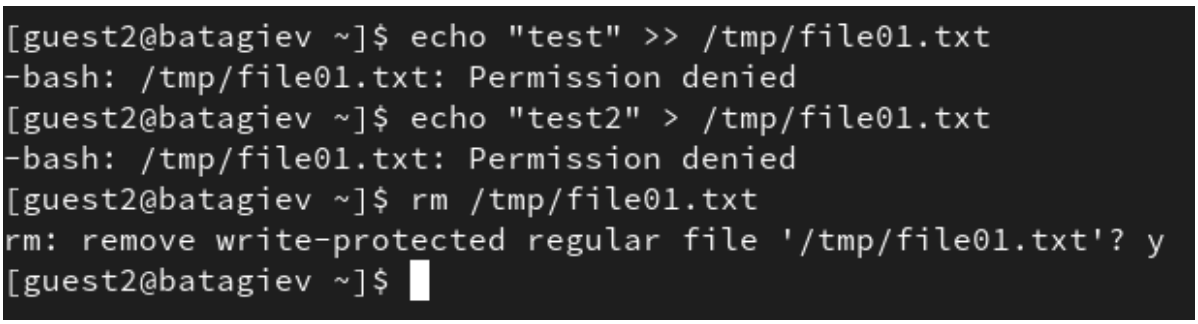
[guest2@batagiev ~]$ echo "test" >> /tmp/file01.txt
-bash: /tmp/file01.txt: Permission denied
[guest2@batagiev ~]$ echo "test2" > /tmp/file01.txt
-bash: /tmp/file01.txt: Permission denied
[guest2@batagiev ~]$ rm /tmp/file01.txt
rm: remove write-protected regular file '/tmp/file01.txt'? y
[guest2@batagiev ~]$

Рис. 2.19: /tmp

# 3 Выводы

В результате выполнения работы я выполнил цели работы.