

# Дискреционное разграничение прав в Linux. Исследование влияния дополнительных атрибутов

---

Кашкин Иван Евгеньевич

5 октября, 2024, Москва, Россия

Российский Университет Дружбы Народов

# Цели и задачи

---

- SUID - разрешение на установку идентификатора пользователя. Это бит разрешения, который позволяет пользователю запускать исполняемый файл с правами владельца этого файла.
- SGID - разрешение на установку идентификатора группы. Принцип работы очень похож на SUID с отличием, что файл будет запускаться пользователем от имени группы, которая владеет файлом.

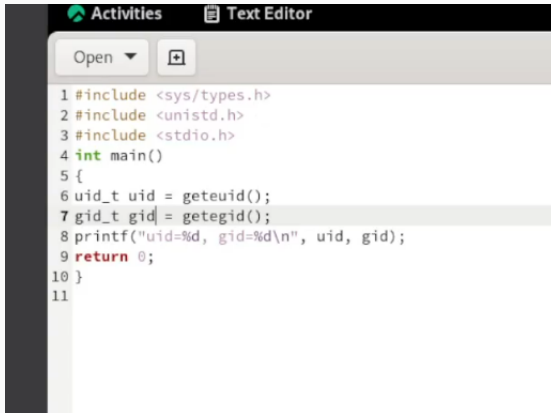
## Цель лабораторной работы

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.

# **Выполнение лабораторной работы**

---

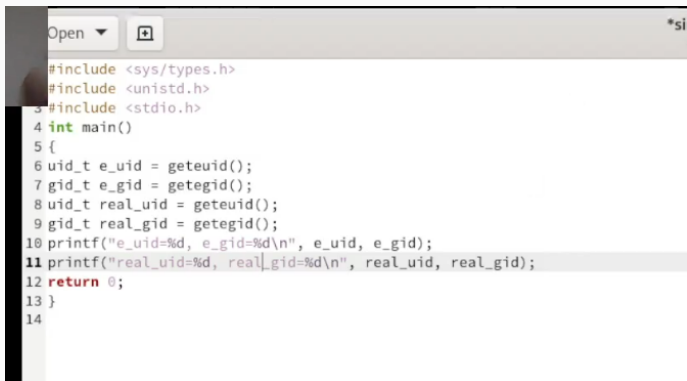
# Программа simpleid


A screenshot of a Linux desktop environment. At the top, there is a black bar with the text "Activities" and "Text Editor". Below this, there is a window titled "Text Editor" with a menu bar containing "Open" and a "+" icon. The main area of the window displays the source code of the simpleid program. The code is as follows:

```
1 #include <sys/types.h>
2 #include <unistd.h>
3 #include <stdio.h>
4 int main()
5 {
6     uid_t uid = geteuid();
7     gid_t gid = getegid();
8     printf("uid=%d, gid=%d\n", uid, gid);
9     return 0;
10 }
11
```

**Figure 1:** результат программы simpleid

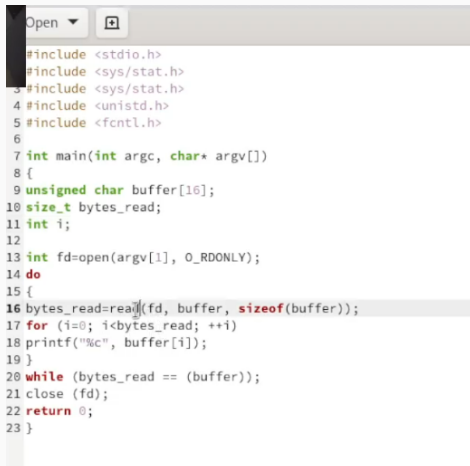
# Программа simpleid2



```
Open ▾  *sir
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
4 int main()
5 {
6     uid_t e_uid = geteuid();
7     gid_t e_gid = getegid();
8     uid_t real_uid = geteuid();
9     gid_t real_gid = getegid();
10    printf("e_uid=%d, e_gid=%d\n", e_uid, e_gid);
11    printf("real_uid=%d, real_gid=%d\n", real_uid, real_gid);
12    return 0;
13 }
14
```

**Figure 2:** результат программы simpleid2

# Программа readfile

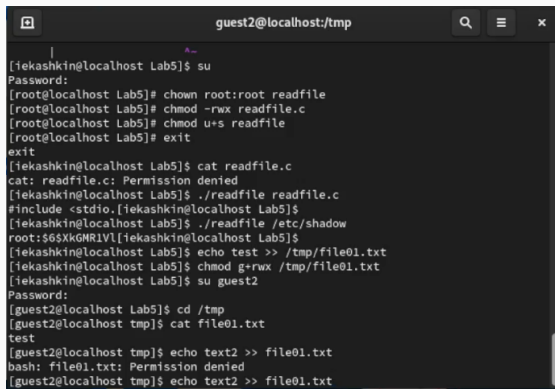


```
1 #include <stdio.h>
2 #include <sys/stat.h>
3 #include <sys/stat.h>
4 #include <unistd.h>
5 #include <fcntl.h>
6
7 int main(int argc, char* argv[])
8 {
9     unsigned char buffer[16];
10    size_t bytes_read;
11    int i;
12
13    int fd=open(argv[1], O_RDONLY);
14    do
15    {
16        bytes_read=read(fd, buffer, sizeof(buffer));
17        for (i=0; i<bytes_read; ++i)
18            printf("%c", buffer[i]);
19    }
20    while (bytes_read == (buffer));
21    close (fd);
22    return 0;
23 }
```

**Figure 3:** результат программы readfile



# Исследование Sticky-бита



```
guest2@localhost:/tmp
[iekashkin@localhost Lab5]$ su
Password:
[root@localhost Lab5]# chown root:root readfile
[root@localhost Lab5]# chmod -rwx readfile.c
[root@localhost Lab5]# chmod u+s readfile
[root@localhost Lab5]# exit
exit
[iekashkin@localhost Lab5]$ cat readfile.c
cat: readfile.c: Permission denied
[iekashkin@localhost Lab5]$ ./readfile readfile.c
#include <stdio.h>[iekashkin@localhost Lab5]$
[iekashkin@localhost Lab5]$ ./readfile /etc/shadow
root:$6$XkGMR1Vl[iekashkin@localhost Lab5]$
[iekashkin@localhost Lab5]$ echo test >> /tmp/file01.txt
[iekashkin@localhost Lab5]$ chmod g+rwX /tmp/file01.txt
[iekashkin@localhost Lab5]$ su guest2
Password:
[guest2@localhost Lab5]$ cd /tmp
[guest2@localhost tmp]$ cat file01.txt
test
[guest2@localhost tmp]$ echo text2 >> file01.txt
bash: file01.txt: Permission denied
[guest2@localhost tmp]$ echo text2 >> file01.txt
```

Figure 4: исследование Sticky-бита

## **Выводы**

---

Изучили механизмы изменения идентификаторов, применения SetUID- и Sticky-битов. Получили практические навыки работы в консоли с дополнительными атрибутами. Также мы рассмотрели работу механизма смены идентификатора процессов пользователей и влияние бита Sticky на запись и удаление файлов.