

Отчет по лабораторной работе №5

по дисциплине: Информационная безопасность

Ким Михаил Алексеевич

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
3.1	Термины	6
4	Выполнение лабораторной работы	7
4.1	Создание программы	7
4.2	Исследование Sticky-бита	15
5	Анализ результатов	19
6	Выводы	20
	Список литературы	21

Список иллюстраций

4.1	Программа simpleid.c	8
4.2	Компиляция и выполнение программы simpleid.c	8
4.3	Программа simpleid2.c	9
4.4	Компиляция и выполнение программы simpleid2.c	9
4.5	Изменение владельца файла и добавление SetUID-бита	10
4.6	Сравнение ./simpleid2 и id при установленном SetUID-бите	10
4.7	Сравнение ./simpleid2 и id при установленном SetGID-бите	10
4.8	Программа readfile.c	12
4.9	Смена владельца и прав файла readfile.c	12
4.10	Смена владельца и прав файла readfile	13
4.11	Чтение файлов при помощи программы readfile. 1	14
4.12	Чтение файлов при помощи программы readfile. 2	15
4.13	Просмотр атрибутов директории /tmp. Изменение атрибутов файла /tmp/file01.txt	16
4.14	Попытка изменить файл /tmp/file01.txt от имени пользователя guest2	16
4.15	Попытка удалить файл /tmp/file01.txt от имени пользователя guest2	17
4.16	Попытка изменить и удалить файл /tmp/file01.txt от имени пользо- вателя guest2 при отсутствии Sticky-бита на директории /tmp . . .	17
4.17	Возвращение Sticky-бита директории /tmp	18

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 Задание

1. Произвести эксперименты с дополнительными атрибутами прав доступа.

3 Теоретическое введение

3.1 Термины

- Терминал (или «Bash», сокращение от «Bourne-Again shell») — это программа, которая используется для взаимодействия с командной оболочкой. Терминал применяется для выполнения административных задач, например: установку пакетов, действия с файлами и управление пользователями. [1]
- Права доступа определяют, какие действия конкретный пользователь может или не может совершать с определенными файлами и каталогами. [2]
- В UNIX-системах, кроме стандартных прав доступа, существуют также дополнительные или специальные атрибуты файлов, которые поддерживает файловая система. [3]
- Setuid – это бит разрешения, который позволяет пользователю запускать исполняемый файл с правами владельца этого файла. Другими словами, использование этого бита позволяет нам поднять привилегии пользователя в случае, если это необходимо.
- Sticky Bit - в случае, если этот бит установлен для папки, то файлы в этой папке могут быть удалены только их владельцем. [4]

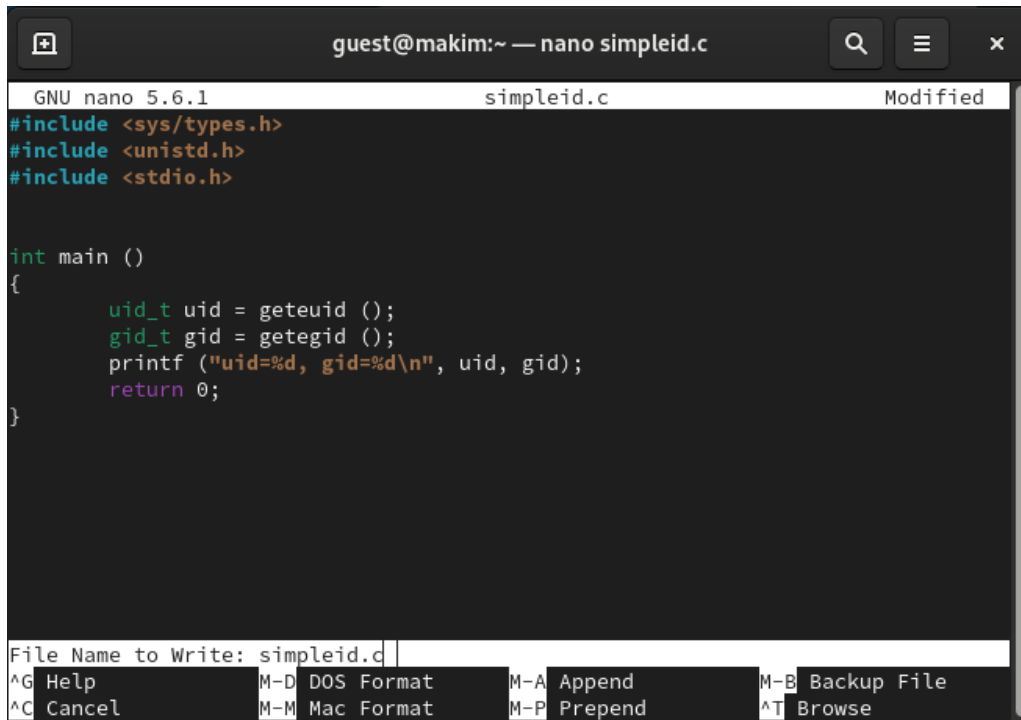
4 Выполнение лабораторной работы

4.1 Создание программы

1. Создадим программу `simpleid.c` от имени пользователя `guest` командой `nano simpleid.c` (рис. 4.1).

```
#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;
}
```



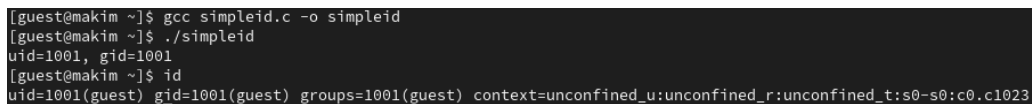
```
guest@makim:~ — nano simpleid.c
GNU nano 5.6.1 simpleid.c Modified
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

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

File Name to Write: simpleid.c
^G Help      M-D DOS Format  M-A Append     M-B Backup File
^C Cancel    M-M Mac Format  M-P Prepend    ^T Browse
```

Рис. 4.1: Программа simpleid.c

2. Скомпилируем и выполним программу. Сравним полученный результат с выполнением системной команды `id`: информация совпадает (рис. 4.2).



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

Рис. 4.2: Компиляция и выполнение программы simpleid.c

3. Усложним программу, добавив вывод действительных идентификаторов. Назовем программу `simpleid2.c` (рис. 4.3).

```
#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;
}

```

Рис. 4.3: Программа simpleid2.c

4. Скомпилируем и выполним программу (рис. 4.4).

```

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

```

Рис. 4.4: Компиляция и выполнение программы simpleid2.c

5. От имени суперпользователя выполните команды: `chown root:guest /home/guest/simpleid2` - изменяем владельца файла `simpleid2` с `guest` на `root` с указанием группы; `chmod u+s /home/guest/simpleid2` - добавляем на файл SetUID-бит (рис. 4.5).

```
[root@makim ~]# chown root:guest /home/guest/simpleid2
[root@makim ~]# chmod u+s /home/guest/simpleid2
```

Рис. 4.5: Изменение владельца файла и добавление SetUID-бита

6. Выполним проверку правильности установки новых атрибутов и смены владельца файла `simpleid2` командой `ll simpleid2`. Запустим `simpleid2` и `id` командами: `./simpleid2` и `id`. Сравним результаты: `e_uid` у нас изменилось (рис. 4.6).

```
[guest@makim ~]$ ll simpleid2
-rwsr-xr-x. 1 root guest 26064 Oct  7 16:54 simpleid2
[guest@makim ~]$ ./simpleid2
e_uid=0, e_gid=1001
real_uid=1001, real_gid=1001
[guest@makim ~]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Рис. 4.6: Сравнение `./simpleid2` и `id` при установленном SetUID-бите

7. Прделаем аналогичные действия относительно SetGID-бита. Как видим, `e_uid` у нас снова изменился (рис. 4.7).

```
[guest@makim ~]$ ll simpleid2
-rwsr-xr-x. 1 root guest 26064 Oct  7 16:54 simpleid2
[guest@makim ~]$ su -
Password:
[root@makim ~]# chmod u-s /home/guest/simpleid2
[root@makim ~]# ll /home/guest/simpleid2
-rwxr-xr-x. 1 root guest 26064 Oct  7 16:54 /home/guest/simpleid2
[root@makim ~]# chmod g+s /home/guest/simpleid2
[root@makim ~]# ll /home/guest/simpleid2
-rwxr-sr-x. 1 root guest 26064 Oct  7 16:54 /home/guest/simpleid2
[root@makim ~]# exit
logout
[guest@makim ~]$ ll simpleid2
-rwxr-sr-x. 1 root guest 26064 Oct  7 16:54 simpleid2
[guest@makim ~]$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@makim ~]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Рис. 4.7: Сравнение `./simpleid2` и `id` при установленном SetGID-бите

8. Создадим программу `readfile.c` (рис. 4.8).

```
#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;
}
```

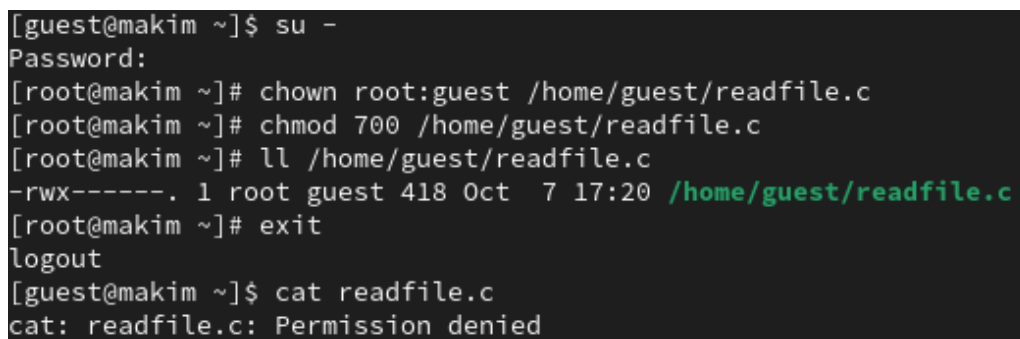


```
guest@makim:~ — nano readfile.c
GNU nano 5.6.1 readfile.c Modified
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

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;
}
File Name to Write: readfile.c
^G Help      M-D DOS Format  M-A Append     M-B Backup File
^C Cancel    M-M Mac Format  M-P Prepend    ^T Browse
```

Рис. 4.8: Программа readfile.c

9. Сменим владельца у файла readfile.c и изменим права так, чтобы только суперпользователь (root) мог прочитать его, а guest не мог. Проверим, что пользователь guest не может прочитать файл readfile.c. (рис. 4.9).



```
[guest@makim ~]$ su -
Password:
[root@makim ~]# chown root:guest /home/guest/readfile.c
[root@makim ~]# chmod 700 /home/guest/readfile.c
[root@makim ~]# ll /home/guest/readfile.c
-rwx-----. 1 root guest 418 Oct  7 17:20 /home/guest/readfile.c
[root@makim ~]# exit
logout
[guest@makim ~]$ cat readfile.c
cat: readfile.c: Permission denied
```

Рис. 4.9: Смена владельца и прав файла readfile.c

10. Сменим у программы readfile владельца и установим SetUD-бит (рис. 4.10).

```

[guest@makim ~]$ ll
total 100
drwxr-xr-x. 2 guest guest    6 Sep 16 18:29 Desktop
drwx-----, 2 guest guest   19 Sep 30 18:06 dir1
drwxr-xr-x. 2 guest guest    6 Sep 16 18:29 Documents
drwxr-xr-x. 2 guest guest    6 Sep 16 18:29 Downloads
drwxr-xr-x. 2 guest guest    6 Sep 16 18:29 Music
drwxr-xr-x. 2 guest guest  100 Sep 16 21:21 Pictures
drwxr-xr-x. 2 guest guest    6 Sep 16 18:29 Public
-rwxr-xr-x. 1 guest guest 26008 Oct  7 17:27 readfile
-rwx-----, 1 root  guest   418 Oct  7 17:20 readfile.c
-rwx-----, 1 guest guest  1209 Sep 16 21:14 script_lab02.sh
-rwxr-xr-x. 1 guest guest 25960 Oct  7 16:52 simpleid
-rwsr-xr-x. 1 root  guest 26064 Oct  7 16:54 simpleid2
-rw-r--r--, 1 guest guest   312 Oct  7 16:54 simpleid2.c
-rw-r--r--, 1 guest guest   181 Oct  7 16:52 simpleid.c
drwxr-xr-x. 2 guest guest    6 Sep 16 18:29 Templates
drwxr-xr-x. 2 guest guest    6 Sep 16 18:29 Videos
[guest@makim ~]$ su -
Password:
[root@makim ~]# chown root:guest /home/guest/readfile
[root@makim ~]# chmod u+s /home/guest/readfile
[root@makim ~]# ll /home/guest/readfile
-rwsr-xr-x. 1 root guest 26008 Oct  7 17:27 /home/guest/readfile
[root@makim ~]# exit
logout

```

Рис. 4.10: Смена владельца и прав файла readfile

11. Проверим, может ли программа прочитать мой собственный файл, файл readfile.c, файл /etc/shadow. Как мы видим, программа может это сделать, так как владелец данной программы - суперпользователь (root) и установлен SetUD-бит. Соответственно, запуск программы осуществляется с правами пользователя - root. Поэтому мы имеем доступ к файлам, недоступным для пользователя guest (рис. 4.11, 4.12).

```

[guest@makim ~]$ ./readfile script_lab02.sh
#!/bin/bash

echo "Starting makim's script for checking file usability:"

read -p "Enter file1 chmod current numeric code: " file1num
echo "====="
echo "1. Trying to create file..."
(umask 777; touch /home/guest/dir1/file_temp1)
echo "====="
echo "2. Trying to remove file..."
rm /home/guest/dir1/file_temp1
echo "====="
echo "3. Trying to write to file..."
echo "test" > /home/guest/dir1/file1
echo "====="
echo "4. Trying to read file..."
cat /home/guest/dir1/file1
echo "====="
echo "5. Trying to change directory..."
cd /home/guest/dir1
echo "====="
echo "6. Trying to view files in direcotry..."
ls -l /home/guest/dir1
echo "====="
echo "7. Trying to rename file..."
(umask 777; mv /home/guest/dir1/file1 /home/guest/dir1/file_temp2)
(umask 777; mv /home/guest/dir1/file_temp2 /home/guest/dir1/file1)
echo "====="
echo "8. Trying to change attributes..."
chmod $file1num /home/guest/dir1/file1
echo "====="
[guest@makim ~]$ ./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@makim ~]$ cat readfile.c
cat: readfile.c: Permission denied

```

Рис. 4.11: Чтение файлов при помощи программы readfile. 1


```
[guest@makim ~]$ ll / | grep tmp
drwxrwxrwt. 14 root root 4096 Oct  7 17:28 tmp
[guest@makim ~]$ echo "test" > /tmp/file01.txt
[guest@makim ~]$ ll /tmp/file01.txt
-rw-r--r--. 1 guest guest 5 Oct  7 17:32 /tmp/file01.txt
[guest@makim ~]$ chmod o+rw /tmp/file01.txt
[guest@makim ~]$ ll /tmp/file01.txt
-rw-r--rw-. 1 guest guest 5 Oct  7 17:32 /tmp/file01.txt
```

Рис. 4.13: Просмотр атрибутов директории /tmp. Изменение атрибутов файла /tmp/file01.txt

2. От пользователя guest2 попробуем дозаписать в файл /tmp/file01.txt слово test2 командой `echo "test2" >> /tmp/file01.txt`. Выполнить данную операцию нельзя. От пользователя guest2 попробуем записать в файл /tmp/file01.txt слово test3, стерев при этом всю имеющуюся в файле информацию командой `echo "test3" > /tmp/file01.txt`. Выполнить данную операцию нельзя. (рис. 4.14).

```
[guest2@makim guest]$ cat /tmp/file01.txt
test
[guest2@makim guest]$ echo "test2" >> /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@makim guest]$ cat /tmp/file01.txt
test
[guest2@makim guest]$ echo "test3" > /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@makim guest]$ cat /tmp/file01.txt
test
[guest2@makim guest]$
```

Рис. 4.14: Попытка изменить файл /tmp/file01.txt от имени пользователя guest2

3. От пользователя guest2 попробуем удалить файл /tmp/file01.txt командой `rm /tmp/file01.txt`. Выполнить данную операцию нельзя. (рис. 4.15).


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

Рис. 4.15: Попытка удалить файл /tmp/file01.txt от имени пользователя guest2

4. Повысим свои права до суперпользователя и выполним команду, снимающую атрибут `t` с директории /tmp: `chmod -t /tmp`. Покинем режим суперпользователя. Проверим, снят ли атрибут `t` у директории /tmp. Повторим предыдущие шаги: теперь мы имеем возможность удалить файл /tmp/file01.txt (рис. 4.16).

```
[guest2@makim guest]$ su -
Password:
[root@makim ~]# chmod -t /tmp
[root@makim ~]# exit
logout
[guest2@makim guest]$ ll / | grep tmp
drwxrwxrwx. 16 root root 4096 Oct  7 17:37 tmp
[guest2@makim guest]$ cat /tmp/file01.txt
test
[guest2@makim guest]$ echo "test2" >> /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@makim guest]$ cat /tmp/file01.txt
test
[guest2@makim guest]$ echo "test3" > /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@makim guest]$ cat /tmp/file01.txt
test
[guest2@makim guest]$ rm /tmp/file01.txt
rm: remove write-protected regular file '/tmp/file01.txt'? y
[guest2@makim guest]$ ll /tmp
total 0
drwx-----, 3 root root 17 Oct  7 16:49 systemd-private-053bc5c2aab647f69583520f7e06de83-chrond.service-0EnLUW
drwx-----, 3 root root 17 Oct  7 16:49 systemd-private-053bc5c2aab647f69583520f7e06de83-colord.service-aZZZv8
drwx-----, 3 root root 17 Oct  7 16:49 systemd-private-053bc5c2aab647f69583520f7e06de83-dbus-broker.service-wZ6zay
drwx-----, 3 root root 17 Oct  7 16:49 systemd-private-053bc5c2aab647f69583520f7e06de83-fwupd.service-MuWzRT
drwx-----, 3 root root 17 Oct  7 16:49 systemd-private-053bc5c2aab647f69583520f7e06de83-ModemManager.service-guI9c
drwx-----, 3 root root 17 Oct  7 16:49 systemd-private-053bc5c2aab647f69583520f7e06de83-power-profiles-daemon.serv
drwx-----, 3 root root 17 Oct  7 16:49 systemd-private-053bc5c2aab647f69583520f7e06de83-rtkit-daemon.service-YmAkY
drwx-----, 3 root root 17 Oct  7 16:49 systemd-private-053bc5c2aab647f69583520f7e06de83-switcheroo-control.service
drwx-----, 3 root root 17 Oct  7 16:49 systemd-private-053bc5c2aab647f69583520f7e06de83-systemd-logind.service-wnv
drwx-----, 3 root root 17 Oct  7 16:49 systemd-private-053bc5c2aab647f69583520f7e06de83-upower.service-m767oI
[guest2@makim guest]$
```

Рис. 4.16: Попытка изменить и удалить файл /tmp/file01.txt от имени пользователя guest2 при отсутствии Sticky-бита на директории /tmp

5. Вернем обратно законный атрибут `t` для директории /tmp (рис. 4.17).

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

Рис. 4.17: Возвращение Sticky-бита директории /tmp

5 Анализ результатов

Работа выполнена без каких-либо проблем. Работа с терминалом ОС Rocky Linux в данном случае нареканий не вызвала. Также порадовало наличие вкладок в терминале «из коробки».

6 Выводы

Изучены механизмы изменения идентификаторов, применения SetUID- и Sticky-битов. Получены практические навыки работы в консоли с дополнительными атрибутами. Рассмотрена работа механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

Список литературы

1. Терминал Linux [Электронный ресурс]. URL: [%7Bhttps://www.reg.ru/blog/linux-shpargalka-komandy-terminala-dlya-novichkov/%7D](https://www.reg.ru/blog/linux-shpargalka-komandy-terminala-dlya-novichkov/).
2. Права доступа [Электронный ресурс]. URL: <https://codechick.io/tutorials/unix-linux/unix-linux-permissions>.
3. Дополнительные атрибуты файлов [Электронный ресурс]. Enchanted Technology, 2022. URL: https://wiki.enchtex.info/doc/linux_file_attributes.
4. Использование SETUID, SETGID и Sticky bit для расширенной настройки прав доступа в операционных системах Linux [Электронный ресурс]. RuVDS, 2021. URL: <https://ruvds.com/ru/helpcenter/suid-sgid-sticky-bit-linux/>.