
Front matter

lang: ru-RU title: "Лабораторная работа №5" subtitle: "Дисциплина:
Основы информационной безопасности" author: "Георгес Геден"

Formatting

toc-title: "Содержание" toc: true # Table of contents toc_depth: 2 lof:
true # Список рисунков lot: true # Список таблиц fontsize: 12pt
linestretch: 1.5 papersize: a4paper documentclass: scrreprt polyglossia-
lang: russian polyglossia-otherlangs: english mainfont: PT Serif romanfont:
PT Serif sansfont: PT Sans monofont: PT Mono mainfontoptions:
Ligatures=TeX romanfontoptions: Ligatures=TeX sansfontoptions:
Ligatures=TeX,Scale=MatchLowercase monofontoptions:
Scale=MatchLowercase indent: true pdf-engine: lualatex header-includes:

- `\linepenalty=10` # the penalty added to the badness of each line within a paragraph (no associated penalty node) Increasing the value makes tex try to have fewer lines in the paragraph.
- `\interlinepenalty=0` # value of the penalty (node) added after each line of a paragraph.
- `\hyphenpenalty=50` # the penalty for line breaking at an automatically inserted hyphen
- `\exhyphenpenalty=50` # the penalty for line breaking at an explicit hyphen
- `\binoppenalty=700` # the penalty for breaking a line at a binary operator
- `\relpenalty=500` # the penalty for breaking a line at a relation
- `\clubpenalty=150` # extra penalty for breaking after first line of a paragraph
- `\widowpenalty=150` # extra penalty for breaking before last line of a paragraph
- `\displaywidowpenalty=50` # extra penalty for breaking before last line before a display math
- `\brokenpenalty=100` # extra penalty for page breaking after a hyphenated line
- `\predisplaypenalty=10000` # penalty for breaking before a display
- `\postdisplaypenalty=0` # penalty for breaking after a display

- `\floatingpenalty = 20000` # penalty for splitting an insertion (can only be split footnote in standard LaTeX)
- `\raggedbottom` # or `\flushbottom`
- `\usepackage{float}` # keep figures where there are in the text
- `\floatplacement{figure}{H}` # keep figures where there are in the text

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.

Теоретическое введение

SetUID, SetGID и Sticky - это специальные типы разрешений позволяют задавать расширенные права доступа на файлы или каталоги. • SetUID (set user ID upon execution — «установка ID пользователя во время выполнения») являются флагами прав доступа в Unix, которые разрешают пользователям запускать исполняемые файлы с правами владельца исполняемого файла. • SetGID (set group ID upon execution — «установка ID группы во время выполнения») являются флагами прав доступа в Unix, которые разрешают пользователям запускать исполняемые файлы с правами группы исполняемого файла. • Sticky bit в основном используется в общих каталогах, таких как /var или /tmp, поскольку пользователи могут создавать файлы, читать и выполнять их, принадлежащие другим пользователям, но не могут удалять файлы, принадлежащие другим пользователям. Более подробно см. в [1].

Выполнение лабораторной работы

1 часть: Создание программы

1) Для начала мы убеждаемся, что компилятор gcc установлен, используя команду "gcc -v". Затем отключаем систему запретов до очередной перезагрузки системы командой "sudo setenforce 0", после чего команда "getenforce" выводит "Permissive" (Рисунок 3.1).

```
ggeorges@localhost:~$ gcc -v
Utilisation des specs internes.
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
Cible : x86_64-redhat-linux
Configuré avec: ../configure --enable-bootstrap --enable-host-pie --enable-host-bind-now
--enable-languages=c,c++,fortran,lto --prefix=/usr --mandir=/usr/share/man --infodir=/u
sr/share/info --with-bugurl=https://bugs.rockylinux.org/ --enable-shared --enable-thread
s=posix --enable-checking=release --with-system-zlib --enable-_cxa_atexit --disable-lib
unwind-exceptions --enable-gnu-unique-object --enable-linker-build-id --with-gcc-major-v
ersion-only --enable-plugin --enable-initfini-array --without-isl --enable-multilib --wi
th-linker-hash-style=gnu --enable-offload-targets=nvptx-none --without-cuda-driver --ena
ble-gnu-indirect-function --enable-cet --with-tune=generic --with-arch_64=x86-64-v2 --wi
th-arch_32=x86-64 --build=x86_64-redhat-linux --with-build-config=bootstrap-lto --enable
-link-serialization=1
Modèle de thread: posix
Algorithmes de compression LTO supportés: zlib zstd
gcc version 11.4.1 20231218 (Red Hat 11.4.1-3) (GCC)
[ggeorges@localhost ~]$ sudo setenforce 0
[ggeorges@localhost ~]$ getenforce
Permissive
[ggeorges@localhost ~]$
```

{ width=70% }

2)Проверяем успешное выполнение команд “whereis gcc” и “whereis g++”(их расположение)(Рисунок 3.2).

```
[ggeorges@localhost ~]$ 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
[ggeorges@localhost ~]$ whereis g++
g++: /usr/bin/g++ /usr/share/man/man1/g++.1.gz
[ggeorges@localhost ~]$
```

{ width=70% }

3)Входим в систему от имени пользователя guest командой “su - guest”. Создаём программу simpleid.c командой “touch simpleid.c” и открываем её в редакторе командой “gedit /home/guest/lab05/simpleid.c” (Рисунок 3.3).

```
guest@localhost:~/lab05

[guest@localhost lab05]$ touch simpleid.c
[guest@localhost lab05]$ ls
simpleid.c
[guest@localhost lab05]$ gedit /home/guest/lab05/simpleid.c

(gedit:46375): dbind-WARNING **: 13:17:53.859: Couldn't register with accessibility bus: Did not receive a reply. Possible causes include: the remote application did not send a reply, the message bus security policy blocked the reply, the reply timeout expired, or the network connection was broken.

(gedit:46375): dconf-WARNING **: 13:17:53.938: failed to commit changes to dconf: L'exécution du processus fils « dbus-launch » a échoué (Aucun fichier ou dossier de ce type)

(gedit:46375): dconf-WARNING **: 13:17:53.945: failed to commit changes to dconf: L'exécution du processus fils « dbus-launch » a échoué (Aucun fichier ou dossier de ce type)

(gedit:46375): dconf-WARNING **: 13:17:54.305: failed to commit changes to dconf: L'exécution du processus fils « dbus-launch » a échoué (Aucun fichier ou dossier de ce type)

(gedit:46375): dconf-WARNING **: 13:17:54.307: failed to commit changes to dconf: L'exécution du processus fils « dbus-launch » a échoué (Aucun fichier ou dossier de ce type)

(gedit:46375): dconf-WARNING **: 13:17:54.307: failed to commit changes to dconf: L'exécution du processus fils « dbus-launch » a échoué (Aucun fichier ou dossier de ce type)
```

{ width=70% }

4)Код программы выглядит следующим образом (Рисунок 3.4).

```
Ouvrir simpleid.c ~lab05 Enregistrer x
simpleid2.c x simpleid.c x

1 #include <sys/types.h>
2 #include <unistd.h>
3 #include <stdio.h>
4
5 int
6 main ()
7 {
8     uid_t uid = geteuid ();
9     gid_t gid = getegid ();
10    printf ("uid=%d, gid=%d\n", uid, gid);
11    return 0;
12 }
```

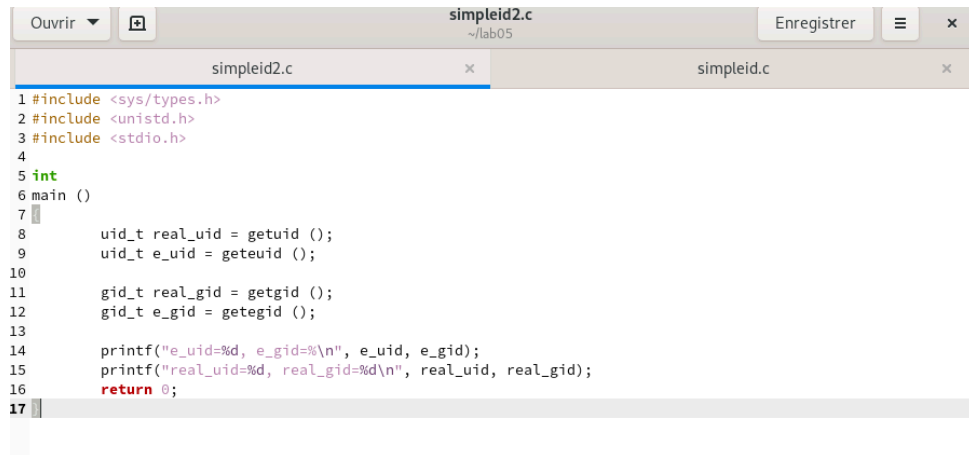
{ width=70% }

5)Скомпилируем программу и убедимся, что файл программы был создан командой "gcc simpleid.c -o simpleid". Выполняем программу simpleid командой "./simpleid", а затем системную программу id командой "id". Результаты, полученные в результате выполнения обеих команд, совпадают(uid=1001 и gid=1001) (Рисунок 3.5).

```
[guest@localhost lab05]$ gcc simpleid.c -o simpleid
[guest@localhost lab05]$ ./simpleid.c
bash: ./simpleid.c: Permission non accordée
[guest@localhost lab05]$ id
uid=1001(guest) gid=1001(guest) groupes=1001(guest) contexte=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@localhost lab05]$
```

{ width=70% }

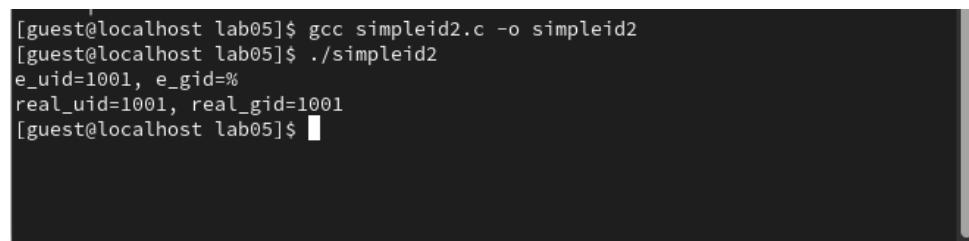
6) Усложняем программу, добавив вывод действительных идентификаторов, новый файл назовём simpleid.c (Рисунок 3.6).



```
1 #include <sys/types.h>
2 #include <unistd.h>
3 #include <stdio.h>
4
5 int
6 main ()
7 {
8     uid_t real_uid = getuid ();
9     uid_t e_uid = geteuid ();
10
11     gid_t real_gid = getgid ();
12     gid_t e_gid = getegid ();
13
14     printf("e_uid=%d, e_gid=%d\n", e_uid, e_gid);
15     printf("real_uid=%d, real_gid=%d\n", real_uid, real_gid);
16     return 0;
17 }
```

{ width=70% }

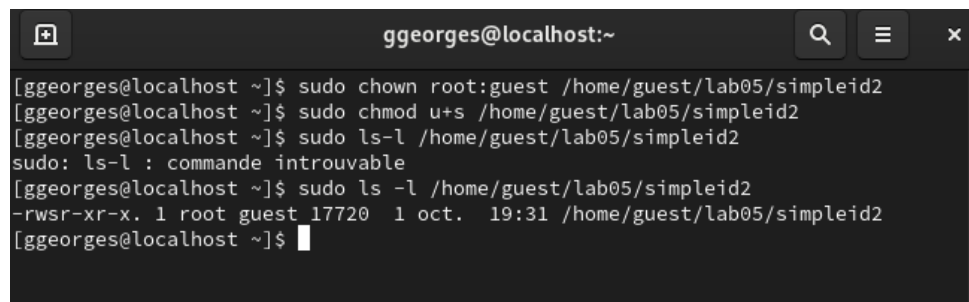
7) Скомпилируем и запустим simpleid2.c командами "gcc simpleid2.c -o simpleid2" и "./simpleid2" (Рисунок 3.7).



```
[guest@localhost lab05]$ gcc simpleid2.c -o simpleid2
[guest@localhost lab05]$ ./simpleid2
e_uid=1001, e_gid=
real_uid=1001, real_gid=1001
[guest@localhost lab05]$
```

{ width=70% }

8) От имени суперпользователя выполняем команды "sudo chown root:guest /home/guest/lab05/simpleid2" и "sudo chmod u+s /home/guest/lab05/simpleid2", затем выполняем проверку правильности установки новых атрибутов и смены владельца файла simpleid2 командой "sudo ls -l /home/guest/lab05/simpleid2" (Рисунок 3.8). Этими командами была произведена смена пользователя файла на root и установлен SetUID-бит.



```
ggeorges@localhost:~
[ggeorges@localhost ~]$ sudo chown root:guest /home/guest/lab05/simpleid2
[ggeorges@localhost ~]$ sudo chmod u+s /home/guest/lab05/simpleid2
[ggeorges@localhost ~]$ sudo ls -l /home/guest/lab05/simpleid2
sudo: ls-l : commande introuvable
[ggeorges@localhost ~]$ sudo ls -l /home/guest/lab05/simpleid2
-rwsr-xr-x. 1 root guest 17720 1 oct. 19:31 /home/guest/lab05/simpleid2
[ggeorges@localhost ~]$
```

{ width=70% }

9)Запускаем программы simpleid2 и id. Теперь появились различия в uid (Рисунок 3.9).

```
[guest@localhost lab05]$ gcc simpleid2.c -o simpleid2
[guest@localhost lab05]$ ./simpleid2
e_uid=1001, e_gid=%
real_uid=1001, real_gid=1001
[guest@localhost lab05]$ id
uid=1001(guest) gid=1001(guest) groupes=1001(guest) contexte=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@localhost lab05]$
```

{ width=70% }

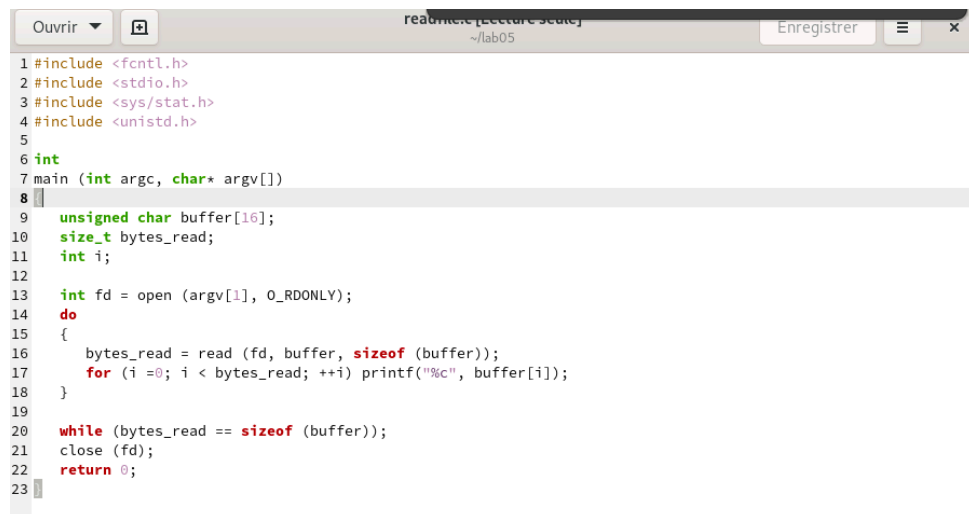
10)Прделаем тоже самое относительно SetGID-бита. Также можем заметить различия с предыдущим пунктом (Рисунок 3.10).

```
[ggeorges@localhost ~]$ sudo chown root:guest /home/guest/lab05/simpleid2
[ggeorges@localhost ~]$ sudo chmod g+s /home/guest/lab05/simpleid2
[ggeorges@localhost ~]$ sudo ls -l /home/guest/lab05/simpleid2
-rwxr-sr-x. 1 root guest 17720 1 oct. 19:31 /home/guest/lab05/simpleid2
[ggeorges@localhost ~]$
```

{ width=70% }

```
[guest@localhost lab05]$ gcc simpleid2.c -o simpleid2
[guest@localhost lab05]$ ./simpleid2
e_uid=1001, e_gid=%
real_uid=1001, real_gid=1001
[guest@localhost lab05]$ id
uid=1001(guest) gid=1001(guest) groupes=1001(guest) contexte=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@localhost lab05]$
```

11)Создаем программу readfile.c(Рисунок 3.11).

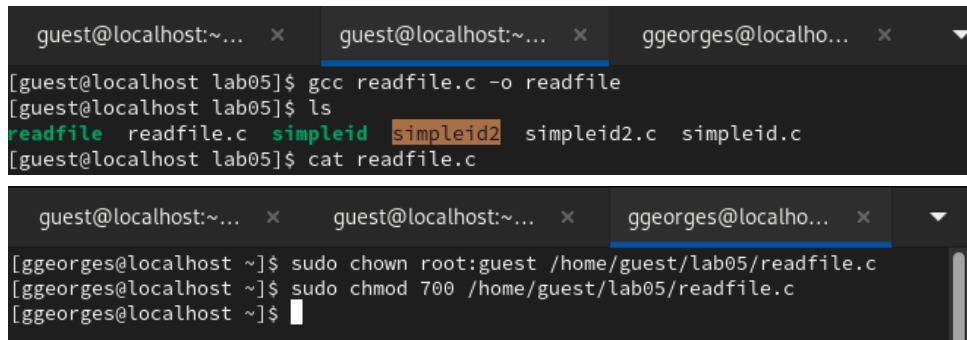


```
1 #include <fcntl.h>
2 #include <stdio.h>
3 #include <sys/stat.h>
4 #include <unistd.h>
5
6 int
7 main (int argc, char* argv[])
8 {
9     unsigned char buffer[16];
10    size_t bytes_read;
11    int i;
12
13    int fd = open (argv[1], O_RDONLY);
14    do
15    {
16        bytes_read = read (fd, buffer,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

{ width=70% }

12)Скомпилируем созданную программу командой "gcc readfile.c -o readfile". Сменим владельца у файла readfile.c командой "sudo chown root:guest /home/guest/readfile.c" и поменяем права так, чтобы только суперпользователь мог прочитать его, а guest не мог, с помощью

команды "sudo chmod 700 /home/guest/readfile.c". Убеждаемся, что пользователь guest не может прочитать файл readfile.c командой "cat readfile.c", получив отказ в доступе (Рисунок 3.12).



```
guest@localhost:~... x guest@localhost:~... x ggeorges@localho... x
[guest@localhost lab05]$ gcc readfile.c -o readfile
[guest@localhost lab05]$ ls
readfile readfile.c simpleid simpleid2 simpleid2.c simpleid.c
[guest@localhost lab05]$ cat readfile.c

[guest@localhost:~... x guest@localhost:~... x ggeorges@localho... x
[ggeorges@localhost ~]$ sudo chown root:guest /home/guest/lab05/readfile.c
[ggeorges@localhost ~]$ sudo chmod 700 /home/guest/lab05/readfile.c
[ggeorges@localhost ~]$
```

{ width=70% }

13) Поменяем владельца у программы readfile и установим SetUID. Проверим, может ли программа readfile прочитать файл readfile.c командой "./readfile readfile.c". Прочитать удалось. Аналогично проверяем, можно ли прочитать файл /etc/shadow. Прочитать удалось (Рисунок 3.13).



```
readfile readfile.c simpleid simpleid2 simpleid2.c simpleid.c
[guest@localhost lab05]$ cat readfile.c
#include <fcntl.h>
#include <stdio.h>
#include <sys/stat.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@localhost lab05]$ ./readfile readfile.c

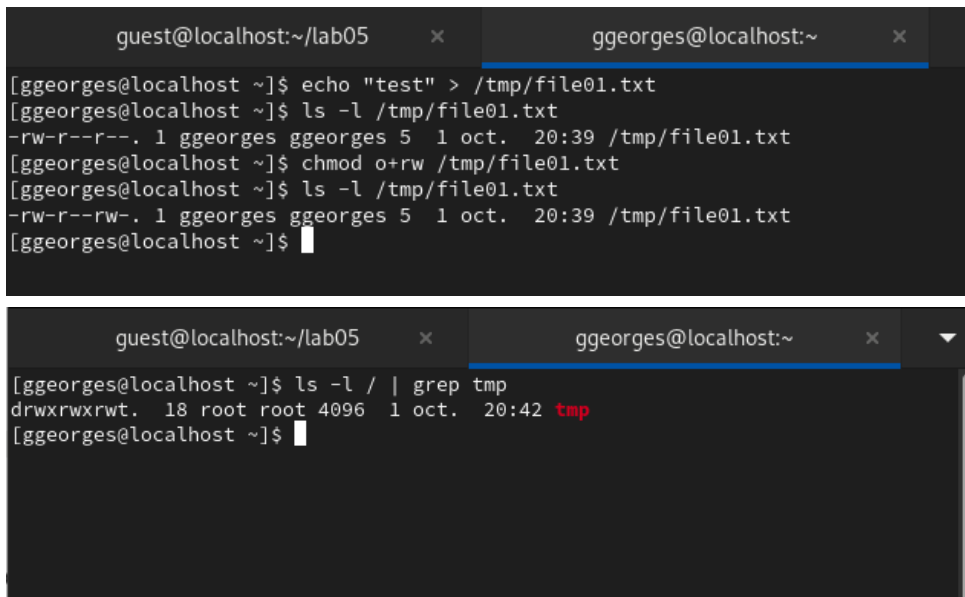
[ggeorges@localhost ~]$ sudo chown root:guest /home/guest/lab05/readfile
[ggeorges@localhost ~]$ sudo chmod u+s /home/guest/lab05/readfile
chmod: mode incorrect for u+s
```

{ width=70% }

2 часть: Исследование Sticky-бита

1) Командой "ls -l | grep tmp" убеждаемся, что атрибут Sticky на директории /tmp установлен. От имени пользователя guest создаём

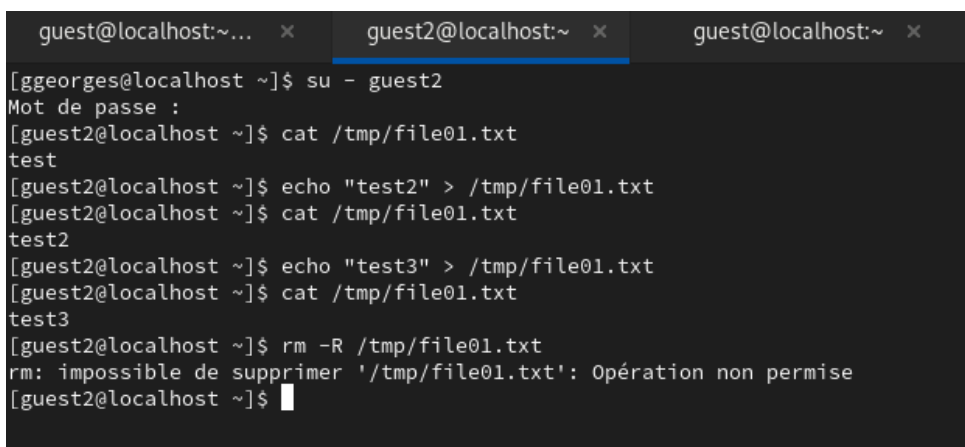
файл file01.txt в директории /tmp со словом test командой "echo"test" > /tmp/file01.txt". Просматриваем атрибуты у только что созданного файла и разрешаем чтение и запись для категории пользователей "все остальные" командами "ls -l /tmp/file01.txt" и "chmod o+rw /tmp/file01.txt" (Рисунок 3.14).



The image shows two terminal windows. The top window, titled 'ggeorges@localhost:~', shows the following commands and output:
[ggeorges@localhost ~]\$ echo "test" > /tmp/file01.txt
[ggeorges@localhost ~]\$ ls -l /tmp/file01.txt
-rw-r--r--. 1 ggeorges ggeorges 5 1 oct. 20:39 /tmp/file01.txt
[ggeorges@localhost ~]\$ chmod o+rw /tmp/file01.txt
[ggeorges@localhost ~]\$ ls -l /tmp/file01.txt
-rw-r--rw-. 1 ggeorges ggeorges 5 1 oct. 20:39 /tmp/file01.txt
[ggeorges@localhost ~]\$
The bottom window, titled 'ggeorges@localhost:~', shows:
[ggeorges@localhost ~]\$ ls -l / | grep tmp
drwxrwxrwt. 18 root root 4096 1 oct. 20:42 tmp
[ggeorges@localhost ~]\$

{ width=70% }

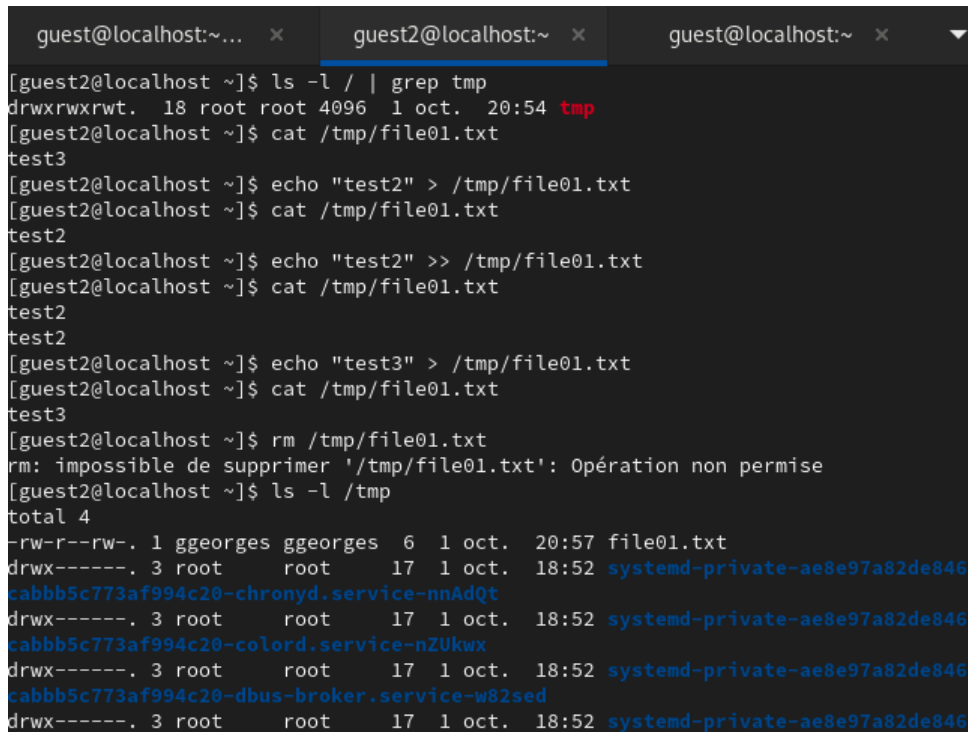
2)От имени пользователя guest2 пробуем прочитать файл командой "cat /tmp/file01.txt" - это удалось. Далее пытаемся дозаписать в файл слово test2, проверить содержимое файла и записать в файл слово test3, стерев при этом всю имеющуюся в файле информацию - эти операции удалось выполнить только в случае, если еще дополнительно разрешить чтение и запись для группы пользователей командой "chmod g+rw /tmp/file01.txt". От имени пользователя guest2 пробуем удалить файл - это не удастся ни в каком из случаев, возникает ошибка (Рисунок 3.15).



The image shows three terminal windows. The middle window, titled 'guest2@localhost:~', shows the following commands and output:
[ggeorges@localhost ~]\$ su - guest2
Mot de passe :
[guest2@localhost ~]\$ cat /tmp/file01.txt
test
[guest2@localhost ~]\$ echo "test2" > /tmp/file01.txt
[guest2@localhost ~]\$ cat /tmp/file01.txt
test2
[guest2@localhost ~]\$ echo "test3" > /tmp/file01.txt
[guest2@localhost ~]\$ cat /tmp/file01.txt
test3
[guest2@localhost ~]\$ rm -R /tmp/file01.txt
rm: impossible de supprimer '/tmp/file01.txt': Opération non permise
[guest2@localhost ~]\$
The other two windows are titled 'guest@localhost:~...' and 'guest@localhost:~' and are currently empty.

{ width=70% }

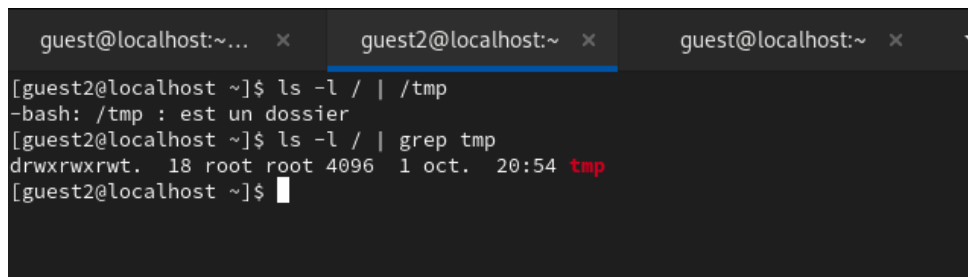
3)Повышаем права до суперпользователя командой "su -" и выполняем команду, снимающую атрибут t с директории /tmp "chmod -t /tmp". После чего покидаем режим суперпользователя командой "exit". Повторяем предыдущие шаги. Теперь нам удаётся удалить файл file01.txt от имени пользователя, не являющегося его владельцем (Рисунок 3.16).



```
guest@localhost:~... x guest2@localhost:~ x guest@localhost:~ x
[guest2@localhost ~]$ ls -l / | grep tmp
drwxrwxrwt. 18 root root 4096 1 oct. 20:54 tmp
[guest2@localhost ~]$ cat /tmp/file01.txt
test3
[guest2@localhost ~]$ echo "test2" > /tmp/file01.txt
[guest2@localhost ~]$ cat /tmp/file01.txt
test2
[guest2@localhost ~]$ echo "test2" >> /tmp/file01.txt
[guest2@localhost ~]$ cat /tmp/file01.txt
test2
test2
[guest2@localhost ~]$ echo "test3" > /tmp/file01.txt
[guest2@localhost ~]$ cat /tmp/file01.txt
test3
[guest2@localhost ~]$ rm /tmp/file01.txt
rm: impossible de supprimer '/tmp/file01.txt': Opération non permise
[guest2@localhost ~]$ ls -l /tmp
total 4
-rw-r--rw-. 1 ggeorges ggeorges 6 1 oct. 20:57 file01.txt
drwx-----, 3 root root 17 1 oct. 18:52 systemd-private-ae8e97a82de846-
cabb5c773af994c20-chronyd.service-nnAdQt
drwx-----, 3 root root 17 1 oct. 18:52 systemd-private-ae8e97a82de846-
cabb5c773af994c20-colord.service-nZUkwX
drwx-----, 3 root root 17 1 oct. 18:52 systemd-private-ae8e97a82de846-
cabb5c773af994c20-dbus-broker.service-w82sed
drwx-----, 3 root root 17 1 oct. 18:52 systemd-private-ae8e97a82de846-
```

{ width=70% }

4)Повышаем свои права до суперпользователя и возвращаем атрибут t на директорию /tmp (Рисунок 3.17).



```
guest@localhost:~... x guest2@localhost:~ x guest@localhost:~ x
[guest2@localhost ~]$ ls -l / | /tmp
-bash: /tmp : est un dossier
[guest2@localhost ~]$ ls -l / | grep tmp
drwxrwxrwt. 18 root root 4096 1 oct. 20:54 tmp
[guest2@localhost ~]$
```

{ width=70% }

Выводы

- В ходе выполнения данной лабораторной работы я изучил механизмы изменения идентификаторов, применение SetUID- и Sticky-битов. Получил практические навыки работы в консоли с

дополнительными атрибутами. Рассмотрел работу механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.