
Front matter

lang: ru-RU title: Отчёт по лабораторной работе №5 author: Георгес
Гедеон institute: РУДН, Москва, Россия

date: 05 Октября 2024

Formatting

toc: false slide_level: 2 theme: metropolis header-includes:

- `\metroset{progressbar=frametitle,sectionpage=progressbar,numbering=fraction}`
- `'\makeatletter'`
- `'\beamer@ignorenonframefalse'`
- `'\makeatother'` aspectratio: 43 section-titles: true

Отчет по лабораторной работе №5

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.

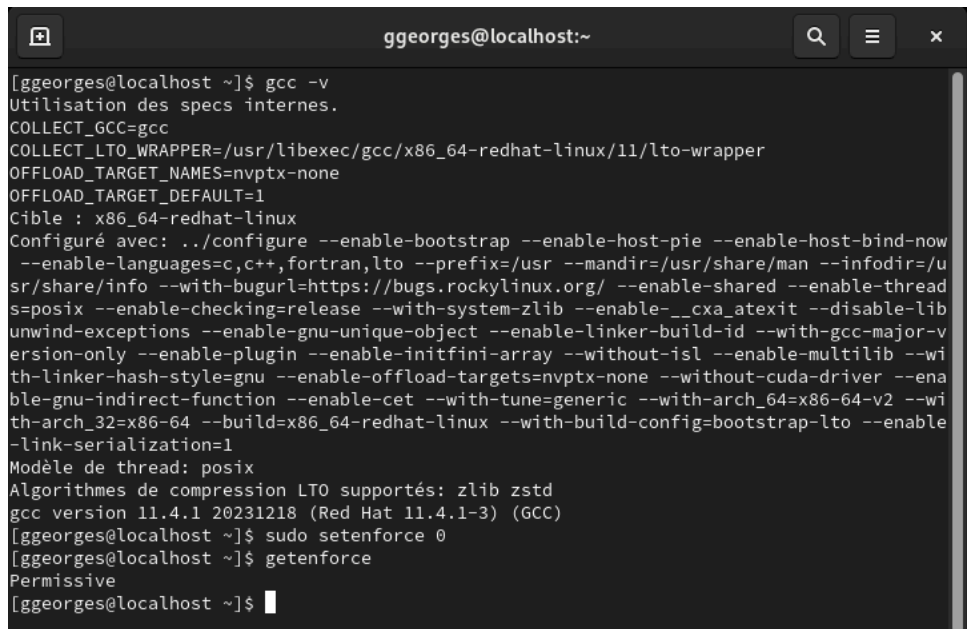
Теоретическое введение

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.

1 часть: Создание программы

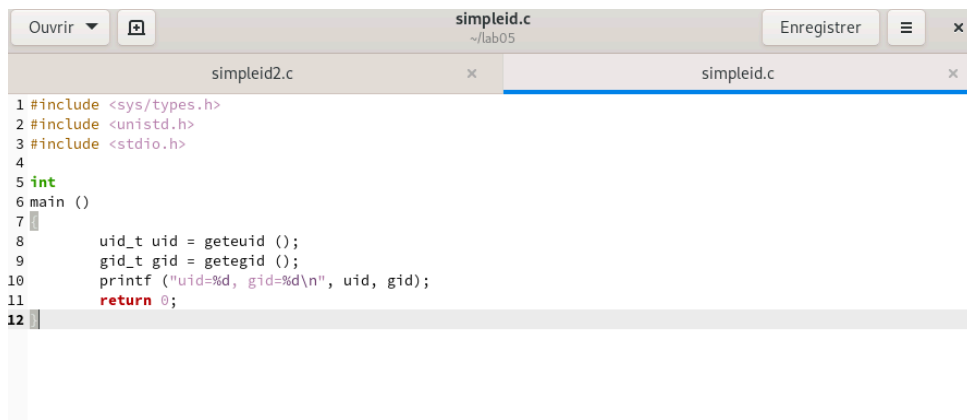
Для начала мы убеждаемся, что компилятор gcc установлен, используя команду "gcc -v". Затем отключаем систему запретов до очередной перезагрузки системы командой "sudo setenforce 0", после чего команда "getenforce" выводит "Permissive".

A terminal window titled 'ggeorges@localhost:~' with search, menu, and close icons. The terminal shows the output of the command 'gcc -v', which includes details about the GCC version (11.4.1), target architecture (x86_64-redhat-linux), and various compiler options. Below this, the command 'sudo setenforce 0' is executed, followed by 'getenforce', which outputs 'Permissive'.

```
[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% }

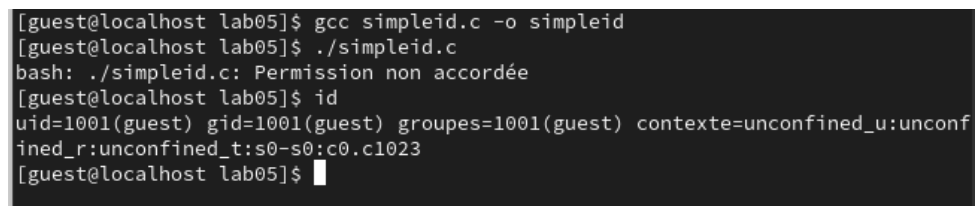
Код программы выглядит следующим образом.



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

{ width=70% }

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).



```
[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% }

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% }

Запускаем программы simpleid2 и id. Теперь появились различия в uid.

```
[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% }

Прделаем тоже самое относительно SetGID-бита. Также можем заметить различия с предыдущим пунктом.

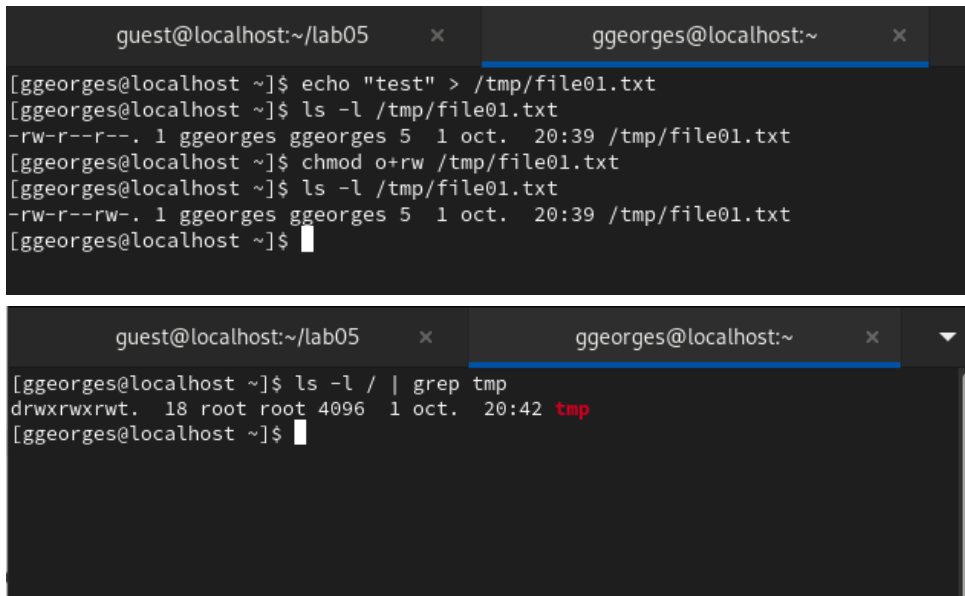
```
[ggeorges@localhost ~]$ sudo chown root:guest /home/guest/lab05/simpleid2  
[ggeorges@localhost ~]$ sudo chmod g+s /home/guest/lab05/simpleid2  
[ggeorges@localhost ~]$ sudo ls -l /home/guest/lab05/simpleid2  
-rwxr-sr-x. 1 root guest 17720  1 oct.  19:31 /home/guest/lab05/simpleid2  
[ggeorges@localhost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

2 часть: Исследование Sticky-бита

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".



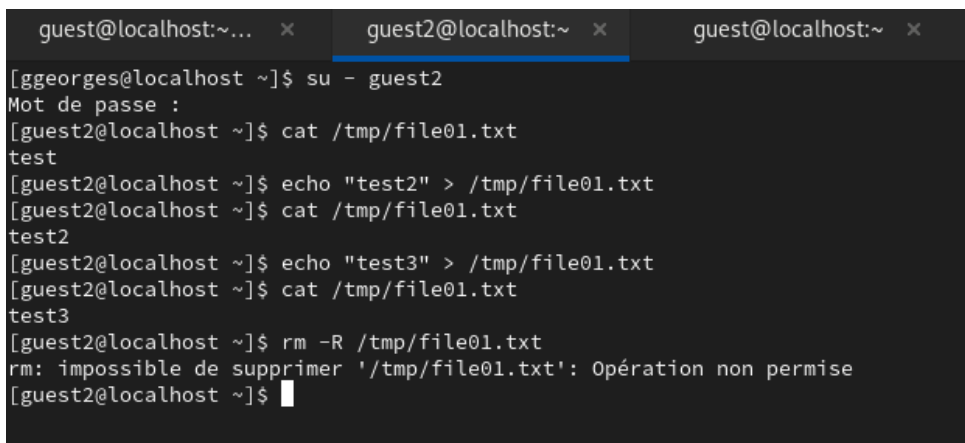
The first terminal window shows the user ggeorges@localhost creating a file /tmp/file01.txt with the command 'echo "test" > /tmp/file01.txt'. It then shows the file's permissions as '-rw-r--r--' and the user changing them to '-rw-r--rw-' with 'chmod o+rw /tmp/file01.txt'. The second terminal window shows the user ggeorges@localhost running 'ls -l / | grep tmp', which displays the file's details: 'drwxrwxrwt. 18 root root 4096 1 oct. 20:42 tmp'.

```
guest@localhost:~/lab05 x ggeorges@localhost:~ x
[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 ~]$

guest@localhost:~/lab05 x ggeorges@localhost:~ x
[ggeorges@localhost ~]$ ls -l / | grep tmp
drwxrwxrwt. 18 root root 4096 1 oct. 20:42 tmp
[ggeorges@localhost ~]$
```

{ width=70% }

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.



The terminal window shows the user guest2@localhost performing several operations on /tmp/file01.txt. It starts with 'su - guest2', then 'cat /tmp/file01.txt' (output: test), 'echo "test2" > /tmp/file01.txt', 'cat /tmp/file01.txt' (output: test2), 'echo "test3" > /tmp/file01.txt', and 'cat /tmp/file01.txt' (output: test3). Finally, it attempts to remove the file with 'rm -R /tmp/file01.txt', which fails with the error 'rm: impossible de supprimer '/tmp/file01.txt': Opération non permise'.

```
guest@localhost:~/... x guest2@localhost:~ x guest@localhost:~ x
[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 ~]$
```

{ width=70% }

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.

```
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-chrond.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% }

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.