

Лабораторная работа №5

Дисциплина: Основы информационной безопасности

Феоктистов Владислав Сергеевич

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	8
3.1	Изменение атрибутов	8
3.2	Использование SetUID, SetGID и Sticky-bit для расширенной на- стройки прав доступа в ОС Linux	9
3.3	Реальные и эффективные идентификаторы	9
3.4	Таблицы	10
4	Выполнение лабораторной работы	12
4.1	Подготовка лабораторного стенда	12
4.2	Создание программы	13
4.3	Исследование Sticky-бита	19
5	Выводы	22
	Список литературы	23

Список иллюстраций

4.1	Подготовка лабораторного стенда	13
4.2	Написание кода simpleid.c	14
4.3	Создание файла simpleid.c и запуск программы	14
4.4	Написание кода simpleid2.c	15
4.5	Создание файла simpleid2.c и запуск программы	15
4.6	Смена владельца файла и добавление атрибута s	16
4.7	Повторный запуск simpleid2 с установленным SetUID-битом . . .	16
4.8	Повторный запуск simpleid2 с установленным SetGID-битом . . .	17
4.9	Написание кода readfile.c	17
4.10	Создание файла readfile.c	18
4.11	Компиляция и изменение прав и владельца файла readfile.c . . .	18
4.12	Смена владельца и атрибутов с проверкой работы программы readfile	19
4.13	Проверка Sticky-бита и создание файла с определенными правами	20
4.14	Работа с Sticky-битом	20
4.15	Снятие Sticky-бита с каталога /tmp	20
4.16	Работа без Sticky-бита	21
4.17	Возвращение Sticky-бита	21

Список таблиц

3.1	Описание некоторых каталогов файловой системы GNU Linux . .	10
3.2	Описание некоторых используемых в работе команд	10

1 Цель работы

Целью данной работы является:

- изучение механизмов изменения идентификаторов, применения SetUID- и Sticky-битов;
- получение практических навыков работы в консоли с дополнительными атрибутами;
- рассмотрение работы механизма смены идентификаторов процессов пользователей;
- изучение влияния Sticky-бита на запись и удаление файлов.

2 Задание

Первая часть задания:

- Написать программу `simpleid.c`, скомпилировать ее и сравнить результат ее выполнения с командой `id`;
- Усложнить программу, назвав его `simpleid2.c` и добавив вывод действительных идентификаторов и повторить предыдущие действия;
- От имени суперпользователя изменить владельца исполняемого файла `simpleid2` и установить для нового владельца файла `SetUID`;
- Запустить исполняемый файл `simpleid2` и снова сравнить результат с командой `id`;
- Повторить два предыдущих действия, установив `SetGID`;
- Создать программу `readfile.c` для чтения содержимого файлов и скомпилировать ее;
- Попробовать прочитать файл `readfile.c`, заранее отобрав права на чтение для пользователя `guest` и сменив владельца файла;
- Попробовать прочитать файл `readfile.c` и `/etc/shadow` через исполняемый файл `readfile`, заранее изменив владельца исполняемого файла и установив для него `SetUID`.

Вторая часть задания:

- Просмотреть наличие `Stick`-бита у каталога `/tmp` и попробовать создать, прочить и изменить текстовый файл в нем, а затем попытаться его удалить;
- Попробовать сделать тоже самое, только сняв атрибут `t`;

- Вернуть обратно Stick-бит для каталога /tmp.

3 Теоретическое введение

3.1 Изменение атрибутов

В ОС Linux права доступа к файлам, атрибуты и владение управляют уровнем доступа, который система обрабатывает, а пользователи имеют к файлам. Это гарантирует, что только авторизованные пользователи и процессы могут получить доступ к определенным файлам и каталогам. Атрибуты состоят из девяти битов, которые и определяют права для разных групп пользователей. Первая тройка битов определяет права доступа для владельца, вторая тройка - для членов группы, последняя тройка - для всех остальных пользователей в системе. Каждая тройка битов (класс пользователей) определяет права на чтение, запись и исполнение. Эта концепция позволяет контролировать, какие пользователи могут читать, записывать (изменять) или выполнять файлы/каталоги.

Чтобы просмотреть права доступа к файлу, используется команда `ls` с опцией `-l`. Первый символ указывает тип файла. Это может быть обычный файл (`-`), каталог (`d`), символическая ссылка (`l`) или другие специфические типы файлов. Следующие девять символов предоставляют доступ к файлу, три тройки по три символа каждая (три пользователя, три типа прав: `r` - чтение, `w` - запись, `x` - исполнение).

Права доступа к файлу/каталогу можно изменить с помощью команды `chmod`. Только `root`, владелец файла или пользователь с привилегией `sudo` могут изменять права доступа к файлу или каталогу. Разрешения можно указывать с помощью символического, числового или справочного режимов [1].

3.2 Использование SetUID, SetGID и Sticky-bit для расширенной настройки прав доступа в ОС Linux

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. Классический пример использования этого бита в операционной системе это команда `sudo`.

Принцип работы **Setgid** очень похож на `setuid` с отличием, что файл будет запускаться пользователем от имени группы, которая владеет файлом.

Последний специальный бит разрешения – это **Sticky Bit**. В случае, если этот бит установлен для папки, то файлы в этой папке могут быть удалены только их владельцем.

Дополнительную информацию можно получить на сайте [2].

3.3 Реальные и эффективные идентификаторы

Реальный идентификатор - идентификатор пользователя, запустившего процесс. Эффективный идентификатор служит для определения прав доступа процесса к системным ресурсам (в первую очередь к ресурсам файловой системы).

Эффективный идентификатор (effective id, effective gid) - идентификатор пользователя или группы, получаемый процессом после вызова выполняемого файла; определяет права процесса. . Возможность изменения эффективных идентификаторов процесса удобна для организации абстрактных типов данных. Используя этот механизм, можно строить файлы, с которыми разрешено выполнять только определенный набор операций.

Обычно реальный и эффективный идентификаторы эквивалентны, т.е. процесс имеет в системе те же права, что и пользователь, запустивший его.

Дополнительную информацию можно получить на сайтах [3], [4].

3.4 Таблицы

Таблица 3.1: Описание некоторых каталогов файловой системы GNU Linux

Имя каталога	Описание каталога
/	Корневая директория, содержащая всю файловую систему
/bin	Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям
/etc	Общесистемные конфигурационные файлы и файлы конфигурации установленных программ
/home	Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя
/media	Точки монтирования для сменных носителей
/root	Домашняя директория пользователя root
/tmp	Временные файлы
/usr	Вторичная иерархия для данных пользователя

Таблица 3.2: Описание некоторых используемых в работе команд

Команда	Описание команды
cat	Вывод содержимого указанного файла.
ls	Выводит содержимое каталога. Опция -l выводит дополнительную информацию, -a отображает скрытые файлы, в названии которых в самом начале стоит символ '.'

Ко-	
манда	Описание команды
<hr/>	
id	Выводи UID (идентификатор пользователя), GID (идентификатор группы пользователя), groups (основные группы пользователя)
chmod	Изменение прав доступа к файлам и каталогам, используемых в Unix-подобных операционных системах.
chown	Изменение владельца и группы пользователей файла или каталога, используемых в Unix-подобных операционных системах.
echo	Вывод переданных аргументов, строки, текста.
touch	Создает текстовый файл по указанному пути и с указанным именем внутри пути.
rm	Удаляет файл(ы) (каталог(и) при указании опции -r) по указанному(ым) пути(ям).
cd	Перемещение по файловой системе.
grep	Дает возможность вести поиск строк. Также можно передать вывод любой команды в grep, что сильно упрощает работу во время поиска
whereis	Выводит расположение бинарных или исходных файлов.
gcc	Копиляция исходного C файла в исполняемый (объектный) файл.

Более подробно об Unix см. в [5–10].

4 Выполнение лабораторной работы

4.1 Подготовка лабораторного стенда

Перед выполнением лабораторной работы стоит убедиться в том, что gcc уже предустановлен в системе, для этого введем команду `gcc -v`. Отсутствие сообщения об ошибке и наличие версии gcc, говорит о том, что gcc уже установлен в системе. Кроме того, так как программы с установленным битом SetUID могут представлять большую брешь в системе безопасности, в современных системах используются дополнительные механизмы защиты. Чтобы защита системы не мешала выполнению работы, необходимо будет ее отключить, для этого нужно будет ввести команду `setenforce 0` от имени суперпользователя. В случае успеха команда `getenforce` должна вывести сообщение *Permissive*.

Дополнительно можно посмотреть пути до компиляторов для языков C и C++ [**cmds:** `whereis gcc` и `whereis g++`] (рис. 4.1).

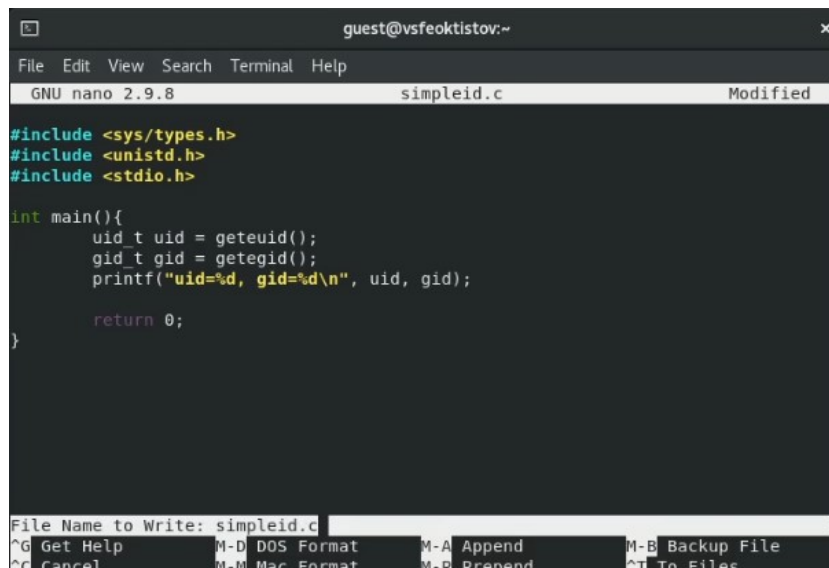
```
guest@vsfeoktistov:~  
File Edit View Search Terminal Help  
Configured with: ../configure --enable-bootstrap --enable-languages=c,c++,fortran,lto --prefix=/usr --mandir=/usr/share/man --infodir=/usr/share/info --with-bug-url=https://bugs.rockylinux.org/ --enable-shared --enable-threads=posix --enable-checking=release --enable-multilib --with-system-zlib --enable- cxa atexit --disable-libunwind-exceptions --enable-gnu-unique-object --enable-linker-build-id --with-gcc-major-version-only --with-linker-hash-style=gnu --enable-plugin --enable-initfini-array --with-isl --disable-libmpx --enable-offload-targets=nvptx-none --without-cuda-driver --enable-gnu-indirect-function --enable-cet --with-tune=generic --with-arch 32=x86-64 --build=x86_64-redhat-linux  
Thread model: posix  
gcc version 8.5.0 20210514 (Red Hat 8.5.0-10) (GCC)  
[guest@vsfeoktistov ~]$ su -  
Password:  
[root@vsfeoktistov ~]# setenforce 0  
[root@vsfeoktistov ~]# getenforce  
Permissive  
[root@vsfeoktistov ~]# exit  
logout  
[guest@vsfeoktistov ~]$ 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  
[guest@vsfeoktistov ~]$ whereis g++  
g++: /usr/bin/g++ /usr/share/man/man1/g++.1.gz  
[guest@vsfeoktistov ~]$
```

Рис. 4.1: Подготовка лабораторного стенда

4.2 Создание программы

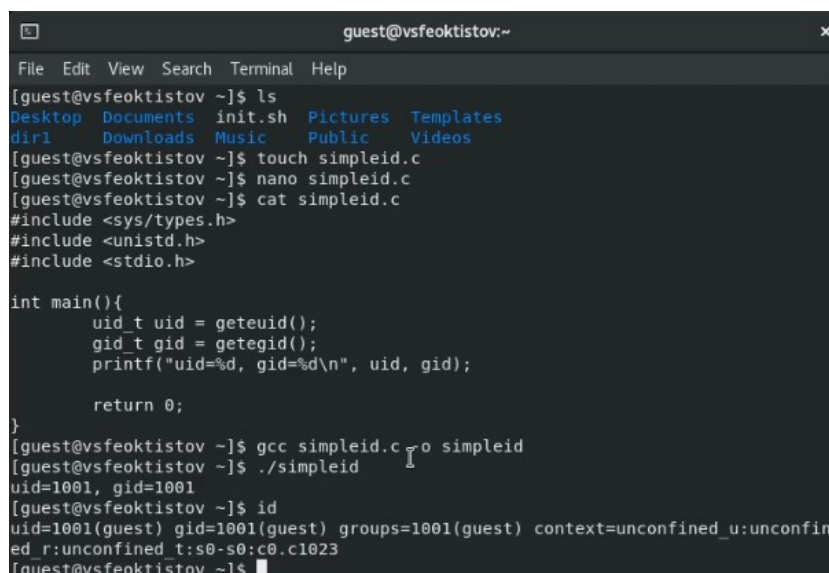
Для начала войдем в систему под пользователем `guest` после запуска виртуальной машины и операционной системы, а после создадим программу *simpleid.c* (рис. 4.2), как это указано в документе с описанием выполнения лабораторной работы №5 [**cmds:** *touch simpleid.c* и *nano simpleid.c*]. Для сохранения изменений нужно нажать клавиши `Ctrl + X`, ввести `y` и нажать `Enter`. Проверить, что исходный код успешно сохранился, можно командой `cat` [**cmd:** *cat simpleid.c*].

После компиляции [**cmd:** *gcc simpleid.c -o simpleid*] и запуска исполняемого файла [**cmd:** *./simpleid*] сравним результат работы программы с выводом команды *id*. Как можно заметить идентификаторы `uid` и `gid` совпадают в обоих случаях (рис. 4.3).



```
guest@vsfeoktistov:~  
File Edit View Search Terminal Help  
GNU nano 2.9.8 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 Get Help      M-D DOS Format  M-A Append      M-B Backup File  
^C Cancel        M-M Mac Format  M-P Prepend     ^T To Files
```

Рис. 4.2: Написание кода simpleid.c



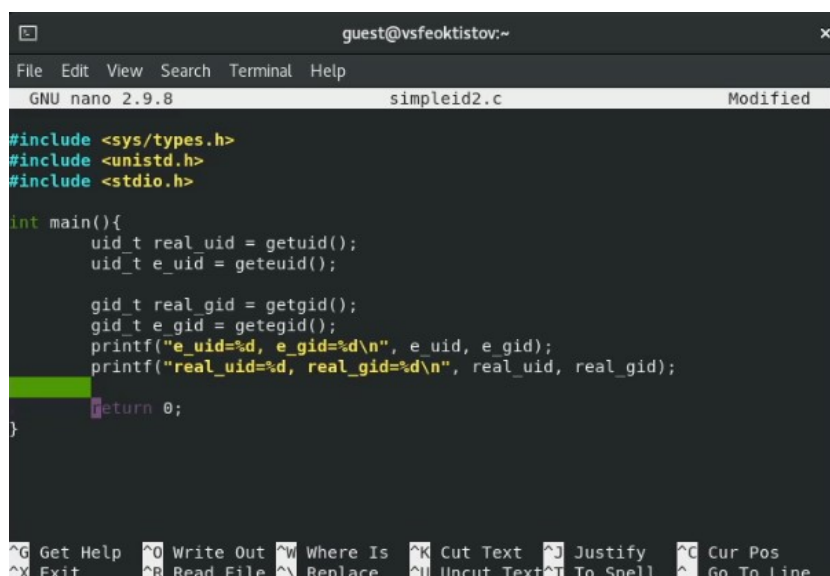
```
guest@vsfeoktistov:~  
File Edit View Search Terminal Help  
[guest@vsfeoktistov ~]$ ls  
Desktop Documents init.sh Pictures Templates  
dirl Downloads Music Public Videos  
[guest@vsfeoktistov ~]$ touch simpleid.c  
[guest@vsfeoktistov ~]$ nano simpleid.c  
[guest@vsfeoktistov ~]$ cat simpleid.c  
#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;  
}  
[guest@vsfeoktistov ~]$ gcc simpleid.c -o simpleid  
[guest@vsfeoktistov ~]$ ./simpleid  
uid=1001, gid=1001  
[guest@vsfeoktistov ~]$ id  
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023  
[guest@vsfeoktistov ~]$
```

Рис. 4.3: Создание файла simpleid.c и запуск программы

Усложним программу, добавив вывод действительных идентификаторов (рис. 4.4). Программу сохраним под названием *simpleid2.c*. Для удобства проще будет скопировать файл *simpleid.c* в этот же каталог, но уже под названием *simpleid2.c* [**cmd:** *cp simpleid.c simpleid2.c*], а затем уже отредактировать его.

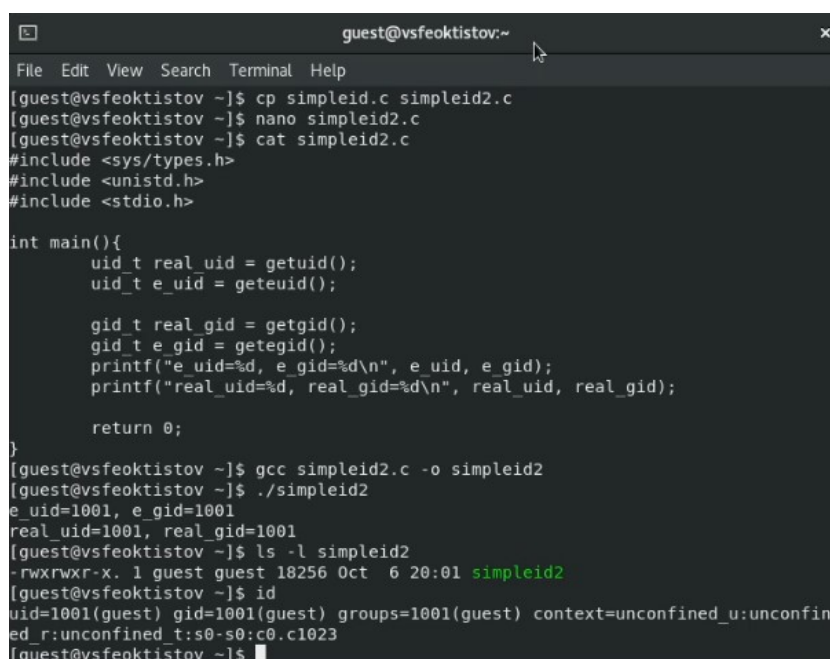
После компиляции [**cmd:** *gcc simpleid2.c -o simpleid2*] и запуска исполняемо-

го файла [cmd: `./simpleid2`] сравним результат работы программы с выводом команды `id`. Как можно заметить реальные, эффективные и пользовательские идентификаторы совпадают (рис. 4.5).



```
guest@vsfeoktistov:~  
File Edit View Search Terminal Help  
GNU nano 2.9.8 simpleid2.c Modified  
  
#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. 4.4: Написание кода simpleid2.c

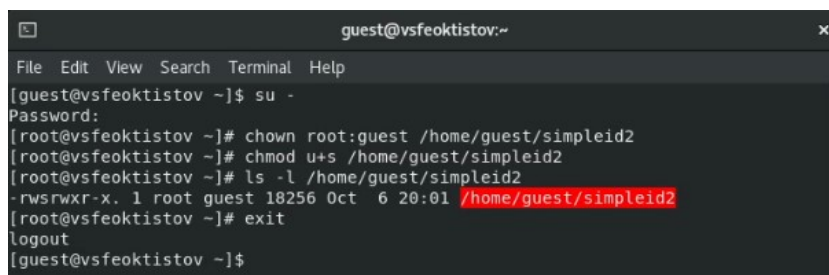


```
guest@vsfeoktistov:~  
File Edit View Search Terminal Help  
[guest@vsfeoktistov ~]$ cp simpleid.c simpleid2.c  
[guest@vsfeoktistov ~]$ nano simpleid2.c  
[guest@vsfeoktistov ~]$ cat simpleid2.c  
#include <sys/types.h>  
#include <unistd.h>  
#include <stdio.h>  
  
int main(){  
    uid_t real_uid = 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;  
}  
[guest@vsfeoktistov ~]$ gcc simpleid2.c -o simpleid2  
[guest@vsfeoktistov ~]$ ./simpleid2  
e_uid=1001, e_gid=1001  
real_uid=1001, real_gid=1001  
[guest@vsfeoktistov ~]$ ls -l simpleid2  
-rwxrwxr-x. 1 guest guest 18256 Oct  6 20:01 simpleid2  
[guest@vsfeoktistov ~]$ id  
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023  
[guest@vsfeoktistov ~]$
```

Рис. 4.5: Создание файла simpleid2.c и запуск программы

От имени суперпользователя [cmd: `su -`] изменим владельца исполняемо-

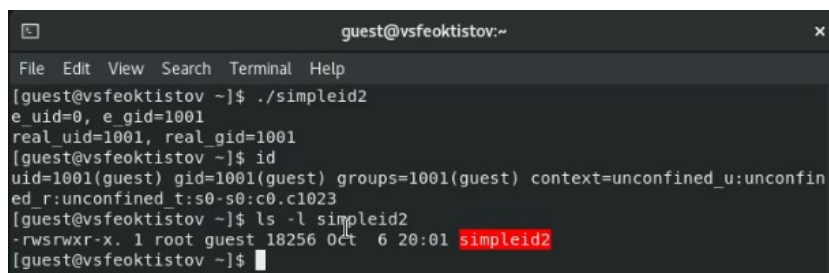
го файла *simpleid2* [**cmd:** *chown root:guest /home/guest/simpleid2*], установим для нового владельца дополнительный атрибут *s* (SetUID) [**cmd:** *chmod u+s /home/guest/simpleid2*] и проверим как изменились атрибуты прав и владельцы файла [**cmd:** *ls -l /home/guest/simpleid2*]. Как видно в первой тройке битов вместо привычного символа *x* стоит символ *s*, что и говорит об успешной установке SetUID. После выполнения этих команд нужно вернуться к пользователю *guest* [**cmd:** *exit*] (рис. 4.6).



```
guest@vsfeoktistov:~  
File Edit View Search Terminal Help  
[guest@vsfeoktistov ~]$ su -  
Password:  
[root@vsfeoktistov ~]# chown root:guest /home/guest/simpleid2  
[root@vsfeoktistov ~]# chmod u+s /home/guest/simpleid2  
[root@vsfeoktistov ~]# ls -l /home/guest/simpleid2  
-rwsrwxr-x. 1 root guest 18256 Oct  6 20:01 /home/guest/simpleid2  
[root@vsfeoktistov ~]# exit  
logout  
[guest@vsfeoktistov ~]$
```

Рис. 4.6: Смена владельца файла и добавление атрибута *s*

Повторно запустим исполняемый файл после вышеописанных действий и сравним результат выполнения программы с выводом команды *id*. Как можно заметить, теперь *e_uid* отличается от *real_uid* и *uid* в *id*. Это произошло, поскольку файл запускался от имени пользователя *guest*, а исполнялся от пользователя *root* (у *root*-пользователя *uid*=0) (рис. 4.7).



```
guest@vsfeoktistov:~  
File Edit View Search Terminal Help  
[guest@vsfeoktistov ~]$ ./simpleid2  
e_uid=0, e_gid=1001  
real_uid=1001, real_gid=1001  
[guest@vsfeoktistov ~]$ id  
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023  
[guest@vsfeoktistov ~]$ ls -l simpleid2  
-rwsrwxr-x. 1 root guest 18256 Oct  6 20:01 simpleid2  
[guest@vsfeoktistov ~]$
```

Рис. 4.7: Повторный запуск *simpleid2* с установленным SetUID-битом

Прделаем те же самые действия только с еще установленным SetGID-битом (рис. 4.8).


```
guest@vsfeoktistov:~  
File Edit View Search Terminal Help  
[guest@vsfeoktistov ~]$ su -  
Password:  
[root@vsfeoktistov ~]# chown root:vsfeoktistov /home/guest/simpleid2  
[root@vsfeoktistov ~]# chmod g+s /home/guest/simpleid2  
[root@vsfeoktistov ~]# ls -l /home/guest/simpleid2  
-rwxrwsr-x. 1 root vsfeoktistov 18256 Oct  6 20:01 /home/guest/simpleid2  
[root@vsfeoktistov ~]# chmod u+s /home/guest/simpleid2  
[root@vsfeoktistov ~]# ls -l /home/guest/simpleid2  
-rwsrwsr-x. 1 root vsfeoktistov 18256 Oct  6 20:01 /home/guest/simpleid2  
[root@vsfeoktistov ~]# exit  
logout  
[guest@vsfeoktistov ~]$ ./simpleid2  
e_uid=0, e_gid=1000  
real_uid=1001, real_gid=1001  
[guest@vsfeoktistov ~]$ id  
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023  
[guest@vsfeoktistov ~]$
```

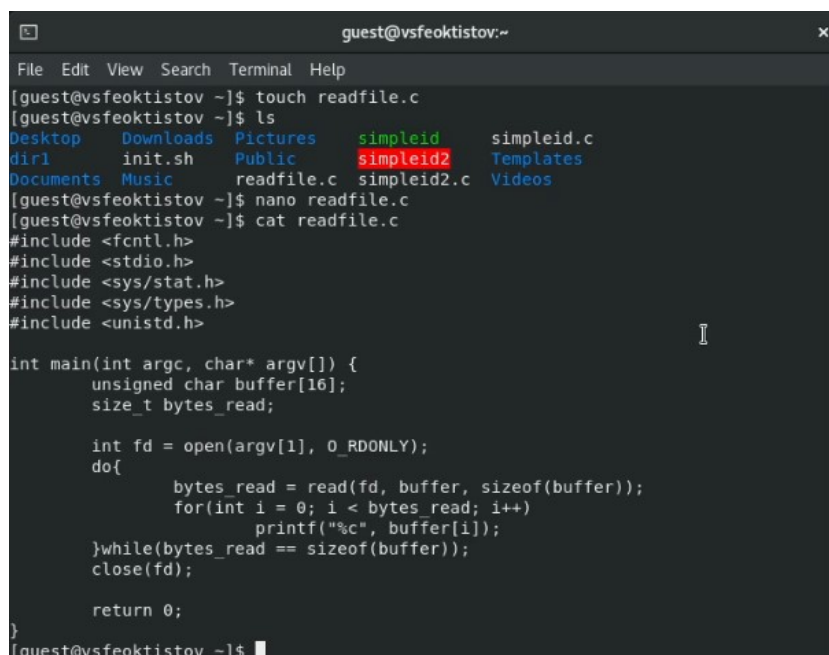
Рис. 4.8: Повторный запуск simpleid2 с установленным SetGID-битом

В результате, теперь еще и `e_gid` отличается от `real_gid` и `gid` пользователя. Теперь программа выполняется от новой установленной группы (`vsfeoktistov` `gid=1000`) (рис. 4.8).

Далее создадим программу `readfile.c`, которая будет читать содержимое указанного файла (рис. 4.9 и 4.10).

```
guest@vsfeoktistov:~  
File Edit View Search Terminal Help  
GNU nano 2.9.8 readfile.c Modified  
#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 fd = open(argv[1], O_RDONLY);  
    do {  
        bytes_read = read(fd, buffer, sizeof(buffer));  
        for(int i = 0; i < bytes_read; i++)  
            printf("%c", buffer[i]);  
    } while(bytes_read == sizeof(buffer));  
    close(fd);  
  
    return 0;  
}
```

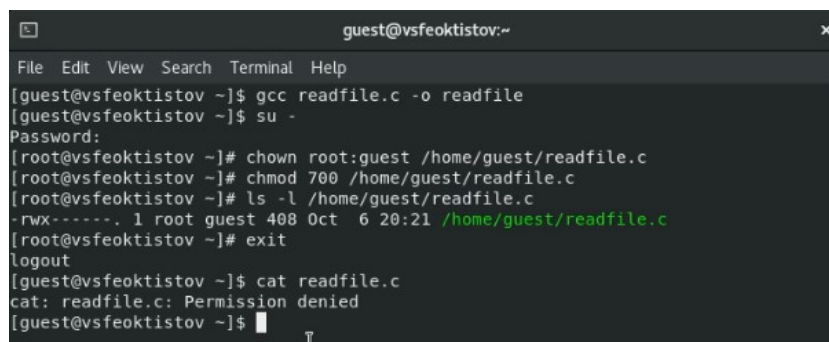
Рис. 4.9: Написание кода `readfile.c`



```
guest@vsfeoktistov:~  
File Edit View Search Terminal Help  
[guest@vsfeoktistov ~]$ touch readfile.c  
[guest@vsfeoktistov ~]$ ls  
Desktop Downloads Pictures simpleid simpleid.c  
dir1 init.sh Public simpleid2 Templates  
Documents Music readfile.c simpleid2.c Videos  
[guest@vsfeoktistov ~]$ nano readfile.c  
[guest@vsfeoktistov ~]$ cat 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 fd = open(argv[1], O_RDONLY);  
    do{  
        bytes_read = read(fd, buffer, sizeof(buffer));  
        for(int i = 0; i < bytes_read; i++)  
            printf("%c", buffer[i]);  
    }while(bytes_read == sizeof(buffer));  
    close(fd);  
  
    return 0;  
}
```

Рис. 4.10: Создание файла readfile.c

Скомпилируем файл [**cmd:** `gcc readfile.c -o readfile`], сменим владельца файла `readfile.c` и изменим права так, чтобы только суперпользователь (`root`) мог прочитывать его, а `guest` не мог [**cmds:** `chown root:guest /home/guest/readfile.c` и `chmod 700 /home/guest/readfile.c`]. Как можно видеть, теперь пользователь `guest` не может его прочитать (рис. 4.11).

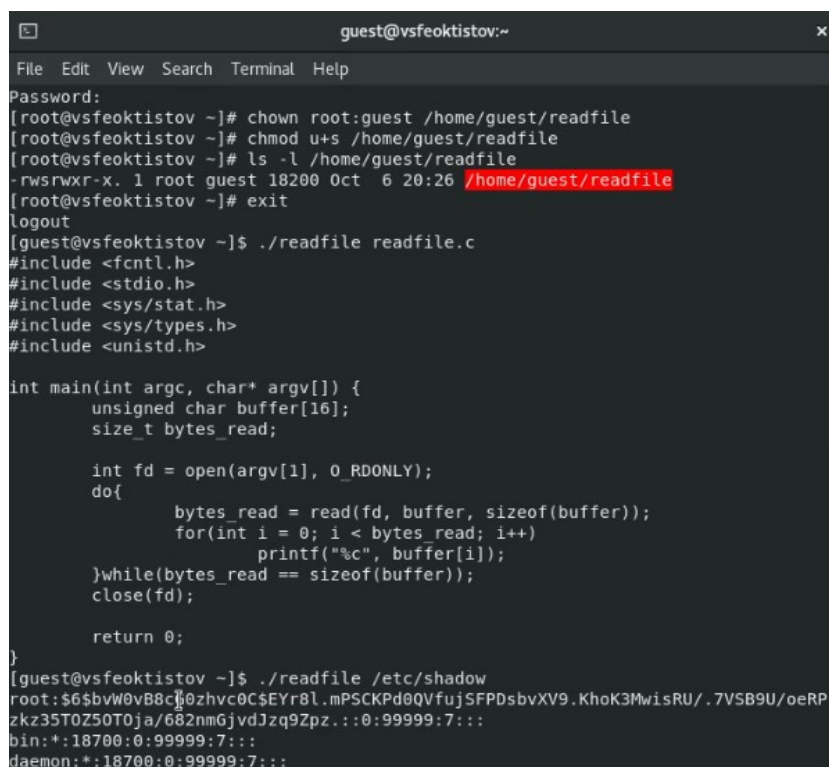


```
guest@vsfeoktistov:~  
File Edit View Search Terminal Help  
[guest@vsfeoktistov ~]$ gcc readfile.c -o readfile  
[guest@vsfeoktistov ~]$ su -  
Password:  
[root@vsfeoktistov ~]# chown root:guest /home/guest/readfile.c  
[root@vsfeoktistov ~]# chmod 700 /home/guest/readfile.c  
[root@vsfeoktistov ~]# ls -l /home/guest/readfile.c  
-rwx-----. 1 root guest 408 Oct 6 20:21 /home/guest/readfile.c  
[root@vsfeoktistov ~]# exit  
logout  
[guest@vsfeoktistov ~]$ cat readfile.c  
cat: readfile.c: Permission denied  
[guest@vsfeoktistov ~]$
```

Рис. 4.11: Компиляция и изменение прав и владельца файла readfile.c

Далее сменим владельца программы `readfile` и установим SetUID-бит [**cmds:** `chown root:guest /home/guest/readfile` и `chmod u+s /home/guest/readfile`]. Затем проверим: может ли программа `readfile` прочитать файл `readfile.c` [**cmd:** `./readfile`

readfile.c]. Как видно из рисунка 4.12, программа успешно прочитала содержимое, поскольку импонила от имени владельца, который имеет право чтения файла *readfile.c*. Кроме того, программа также может вывести содержимое файла */etc/shadow* [**cmd:** *./readfile /etc/shadow*] (рис. 4.12).



```

guest@vsfeoktistov:~
File Edit View Search Terminal Help
Password:
[root@vsfeoktistov ~]# chown root:guest /home/guest/readfile
[root@vsfeoktistov ~]# chmod u+s /home/guest/readfile
[root@vsfeoktistov ~]# ls -l /home/guest/readfile
-rwsrwxr-x. 1 root guest 18200 Oct  6 20:26 /home/guest/readfile
[root@vsfeoktistov ~]# exit
logout
[guest@vsfeoktistov ~]$ ./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 fd = open(argv[1], O_RDONLY);
    do{
        bytes_read = read(fd, buffer, sizeof(buffer));
        for(int i = 0; i < bytes_read; i++)
            printf("%c", buffer[i]);
    }while(bytes_read == sizeof(buffer));
    close(fd);

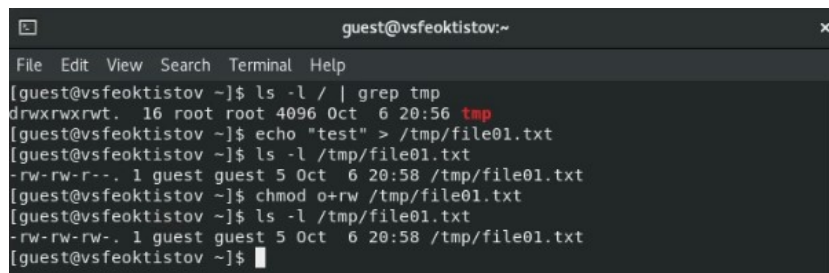
    return 0;
}
[guest@vsfeoktistov ~]$ ./readfile /etc/shadow
root:$6$bvW9vB8c0zhvc0C$EYr8L.mPSCkPd0QVfuJSFPDsbvXV9.KhoK3MwisRU/.7VSB9U/oeRP
zkz35TOZ50T0ja/682nmGjvdJzq9Zpz.:0:99999:7:::
bin:*.18700:0:99999:7:::
daemon:*.18700:0:99999:7:::

```

Рис. 4.12: Смена владельца и атрибутов с проверкой работы программы *readfile*

4.3 Исследование Sticky-бита

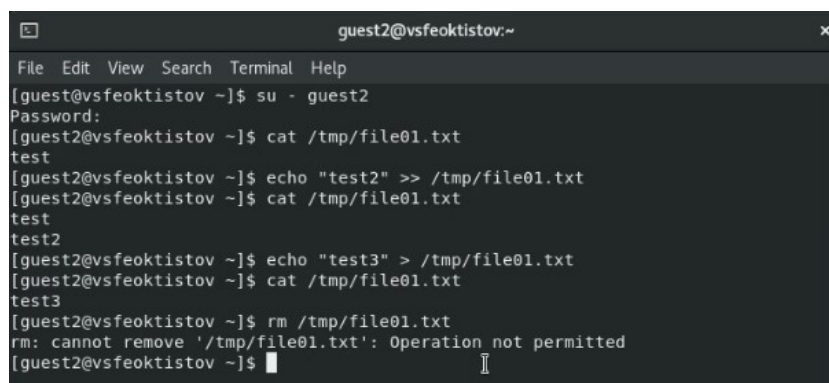
В первую очередь, выясним, установлен ли атрибут Sticky на директорию */tmp* [**cmd:** *ls -l | grep tmp*]. Символ *t* вместо *x* в конце, в третьей тройке битов атрибутов, указывает на его наличие. От имени пользователя *guest* создадим файл *file01.txt* в директории */tmp* со словом *test* [**cmd:** *echo "test" > /tmp/file01.txt*], а после посмотрим атрибуты у только что созданного файла [**cmd:** *ls -l /tmp/file01.txt*] и разрешим чтение и запись для категории пользователей “все остальные” [**cmd:** *chmod o+rw /tmp/file01.txt*] (рис. 4.13).



```
guest@vsfeoktistov:~  
File Edit View Search Terminal Help  
[guest@vsfeoktistov ~]$ ls -l / | grep tmp  
drwxrwxrwt. 16 root root 4096 Oct  6 20:56 tmp  
[guest@vsfeoktistov ~]$ echo "test" > /tmp/file01.txt  
[guest@vsfeoktistov ~]$ ls -l /tmp/file01.txt  
-rw-rw-r--. 1 guest guest 5 Oct  6 20:58 /tmp/file01.txt  
[guest@vsfeoktistov ~]$ chmod o+rw /tmp/file01.txt  
[guest@vsfeoktistov ~]$ ls -l /tmp/file01.txt  
-rw-rw-rw-. 1 guest guest 5 Oct  6 20:58 /tmp/file01.txt  
[guest@vsfeoktistov ~]$
```

Рис. 4.13: Проверка Stick-бита и создание файла с определенными правами

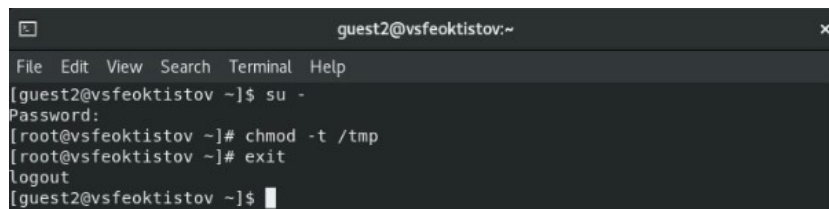
Далее от имени пользователя *guest2* [**cmd:** *su - guest2*] попробуем прочитать, дозаписать, перезаписать и удалить файл *file01.txt*. Как видно из рисунка 4.14, все работает, кроме удаления, поскольку Sticky-бит разрешает удаление файлов в каталоге только владельцу этого каталога.



```
guest2@vsfeoktistov:~  
File Edit View Search Terminal Help  
[guest@vsfeoktistov ~]$ su - guest2  
Password:  
[guest2@vsfeoktistov ~]$ cat /tmp/file01.txt  
test  
[guest2@vsfeoktistov ~]$ echo "test2" >> /tmp/file01.txt  
[guest2@vsfeoktistov ~]$ cat /tmp/file01.txt  
test  
test2  
[guest2@vsfeoktistov ~]$ echo "test3" > /tmp/file01.txt  
[guest2@vsfeoktistov ~]$ cat /tmp/file01.txt  
test3  
[guest2@vsfeoktistov ~]$ rm /tmp/file01.txt  
rm: cannot remove '/tmp/file01.txt': Operation not permitted  
[guest2@vsfeoktistov ~]$
```

Рис. 4.14: Работа с Sticky-битом

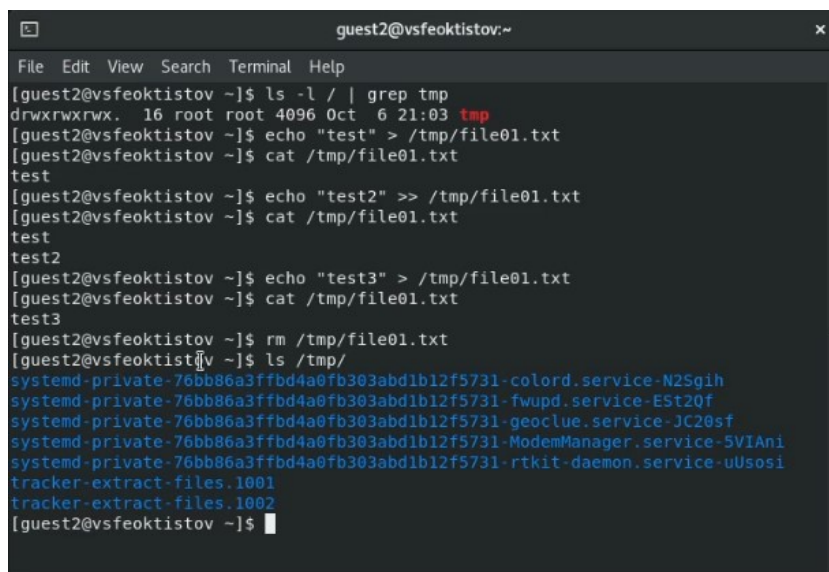
Повысим права до суперпользователя [**cmd:** *su -*], выполним команду, снимающую атрибут *t* (Sticky-бит) с директории */tmp* [**cmd:** *chmod -t /tmp*] и покинем режим суперпользователя [**cmd:** *exit*] (рис. 4.15).



```
guest2@vsfeoktistov:~  
File Edit View Search Terminal Help  
[guest2@vsfeoktistov ~]$ su -  
Password:  
[root@vsfeoktistov ~]# chmod -t /tmp  
[root@vsfeoktistov ~]# exit  
logout  
[guest2@vsfeoktistov ~]$
```

Рис. 4.15: Снятие Sticky-бита с каталога */tmp*

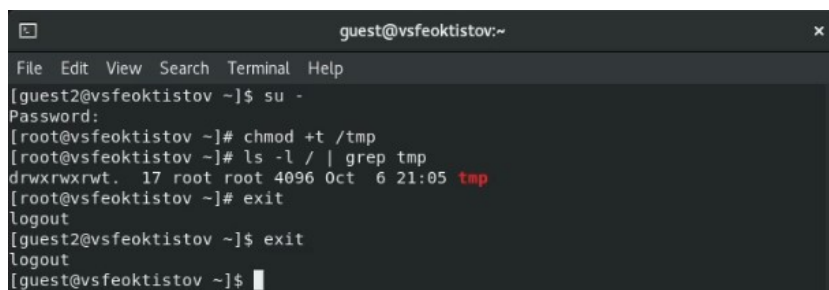
Повторим предыдущие шаги и наблюдаем изменения. В результате, все команды сработали, в том числе и удаление файла из директории (рис. 4.16).



```
guest2@vsfeoktistov:~  
File Edit View Search Terminal Help  
[guest2@vsfeoktistov ~]$ ls -l / | grep tmp  
drwxrwxrwx. 16 root root 4096 Oct 6 21:03 tmp  
[guest2@vsfeoktistov ~]$ echo "test" > /tmp/file01.txt  
[guest2@vsfeoktistov ~]$ cat /tmp/file01.txt  
test  
[guest2@vsfeoktistov ~]$ echo "test2" >> /tmp/file01.txt  
[guest2@vsfeoktistov ~]$ cat /tmp/file01.txt  
test  
test2  
[guest2@vsfeoktistov ~]$ echo "test3" > /tmp/file01.txt  
[guest2@vsfeoktistov ~]$ cat /tmp/file01.txt  
test3  
[guest2@vsfeoktistov ~]$ rm /tmp/file01.txt  
[guest2@vsfeoktistov ~]$ ls /tmp/  
systemd-private-76bb86a3ffb4a0fb303abd1b12f5731-color.service-N2Sgih  
systemd-private-76bb86a3ffb4a0fb303abd1b12f5731-fwupd.service-EST2Qf  
systemd-private-76bb86a3ffb4a0fb303abd1b12f5731-geoclue.service-JC20sf  
systemd-private-76bb86a3ffb4a0fb303abd1b12f5731-ModemManager.service-5VIAAni  
systemd-private-76bb86a3ffb4a0fb303abd1b12f5731-rtkit-daemon.service-uUsosi  
tracker-extract-files.1001  
tracker-extract-files.1002  
[guest2@vsfeoktistov ~]$
```

Рис. 4.16: Работа без Sticky-бита

После окончания работы необходимо вернуть атрибут `t` на директорию `/tmp`, чтобы сохранить безопасность и функциональность системы (рис. 4.17).



```
guest@vsfeoktistov:~  
File Edit View Search Terminal Help  
[guest2@vsfeoktistov ~]$ su -  
Password:  
[root@vsfeoktistov ~]# chmod +t /tmp  
[root@vsfeoktistov ~]# ls -l / | grep tmp  
drwxrwxrwt. 17 root root 4096 Oct 6 21:05 tmp  
[root@vsfeoktistov ~]# exit  
logout  
[guest2@vsfeoktistov ~]$ exit  
logout  
[guest@vsfeoktistov ~]$
```

Рис. 4.17: Возвращение Sticky-бита

5 Выводы

В процессе выполнения лабораторной работы:

- изучил механизмы изменения идентификаторов, применения SetUID- и Sticky-битов;
- получил практические навыки работы в консоли с дополнительными атрибутами;
- рассмотрел работу механизма смены идентификаторов процессов пользователей;
- изучил влияние Sticky-бита на запись и удаление файлов.

Список литературы

1. Понимание прав доступа к файлам в Linux [Электронный ресурс]. Baks, 2021. URL: <https://baks.dev/article/terminal/understanding-linux-file-permissions?ysclid=l8czjs1hnp553393513>.
2. 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/>.
3. Эффективный идентификатор [Электронный ресурс]. Большая энциклопедия нефти и газа, -. URL: <https://www.ngpedia.ru/id43832p1.html>.
4. Реальный (RID) и эффективный (EUID) идентификаторы пользователя [Электронный ресурс]. В помощь веб-мастеру, -. URL: <https://wm-help.net/lib/b/book/173895509/64>.
5. GNU Bash Manual [Электронный ресурс]. Free Software Foundation, 2016. URL: <https://www.gnu.org/software/bash/manual/>.
6. Newham C. Learning the bash Shell: Unix Shell Programming. O'Reilly Media, 2005. 354 с.
7. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.
8. Robbins A. Bash Pocket Reference. O'Reilly Media, 2016. 156 с.
9. Таненбаум Э. Архитектура компьютера. 6-е изд. СПб.: Питер, 2013. 874 с.

10. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.