

# Лабораторная работа №5

Основы информационной безопасности

---

Сабралиева М. Н.

Российский университет дружбы народов, Москва, Россия

## Информация

---

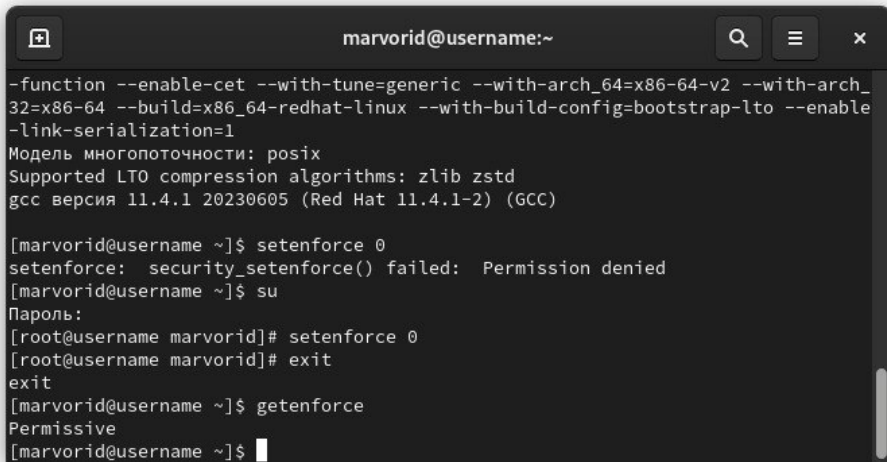
- Сабралиева Марворид Нуралиевна
- студентка НБИбд-01-22 кафедры прикладной информатики и теории вероятностей
- Российский университет дружбы народов

## Элементы презентации

---

- 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.

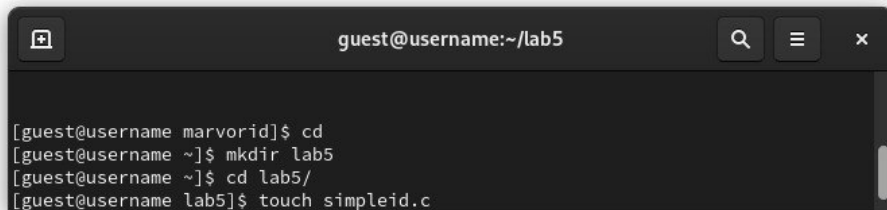
1. Для выполнения части заданий потребуются средства разработки приложений. В частности, при подготовке стенда следует убедиться, что в системе установлен компилятор gcc. У меня его не было, поэтому я установила компилятор
2. Система защиты SELinux не должна мешать выполнению заданий работы. Если вы не знаете, что это такое, просто отключите систему запретов до очередной перезагрузки системы командой `setenforce 0`
3. Команда `getenforce` выводит `Permissive`.



```
marvorid@username:~  
-function --enable-cet --with-tune=generic --with-arch_64=x86-64-v2 --with-arch_  
32=x86-64 --build=x86_64-redhat-linux --with-build-config=bootstrap-lto --enable  
-link-serialization=1  
Модель многопоточности: posix  
Supported LTO compression algorithms: zlib zstd  
gcc версия 11.4.1 20230605 (Red Hat 11.4.1-2) (GCC)  
  
[marvorid@username ~]$ setenforce 0  
setenforce: security_setenforce() failed: Permission denied  
[marvorid@username ~]$ su  
Пароль:  
[root@username marvorid]# setenforce 0  
[root@username marvorid]# exit  
exit  
[marvorid@username ~]$ getenforce  
Permissive  
[marvorid@username ~]$
```

Рис. 1: Подготовка

1. Вошли в систему от имени пользователя guest.



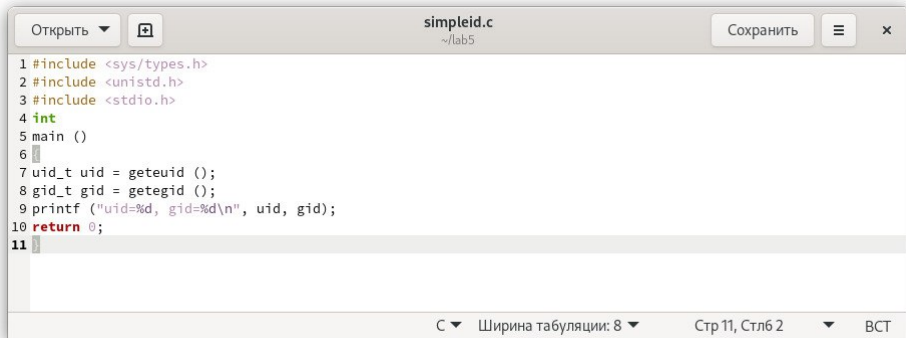
```
guest@username:~/lab5

[guest@username marvorid]$ cd
[guest@username ~]$ mkdir lab5
[guest@username ~]$ cd lab5/
[guest@username lab5]$ touch simpleid.c
```

Рис. 2: Создание программы



## 2. Создаем программу simpleid.c:

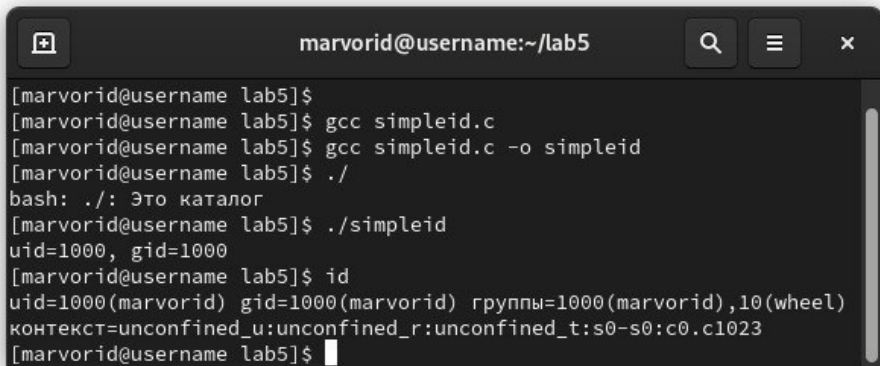


```
1 #include <sys/types.h>
2 #include <unistd.h>
3 #include <stdio.h>
4 int
5 main ()
6 {
7     uid_t uid = geteuid ();
8     gid_t gid = getegid ();
9     printf ("uid=%d, gid=%d\n", uid, gid);
10    return 0;
11 }
```

Окно редактора кода с заголовком **simpleid.c** и путем `~/lab5`. В менюшках: "Открыть", "+", "Сохранить", "≡", "×". В строке состояния: "C", "Ширина табуляции: 8", "Стр 11, Стлб 2", "ВСТ".

Рис. 3: Заполнение программы

3. Скомпилируем программу и убедимся, что файл программы создан: `gcc simpleid.c -o simpleid`
4. Выполним программу `simpleid`: `./simpleid`
5. Выполним системную программу `id`: `id` и сравните полученный вами результат с данными предыдущего пункта задания. `uid` и `gid` совпадают в обеих программах.



A terminal window titled "marvoriid@username:~/lab5" with standard macOS window controls (back, search, menu, close). The terminal shows the following sequence of commands and output:

```
[marvoriid@username lab5]$  
[marvoriid@username lab5]$ gcc simpleid.c  
[marvoriid@username lab5]$ gcc simpleid.c -o simpleid  
[marvoriid@username lab5]$ ./.  
bash: ./: Это каталог  
[marvoriid@username lab5]$ ./simpleid  
uid=1000, gid=1000  
[marvoriid@username lab5]$ id  
uid=1000(marvoriid) gid=1000(marvoriid) группы=1000(marvoriid),10(wheel)  
контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023  
[marvoriid@username lab5]$
```

Рис. 4: Результат программы simpleid

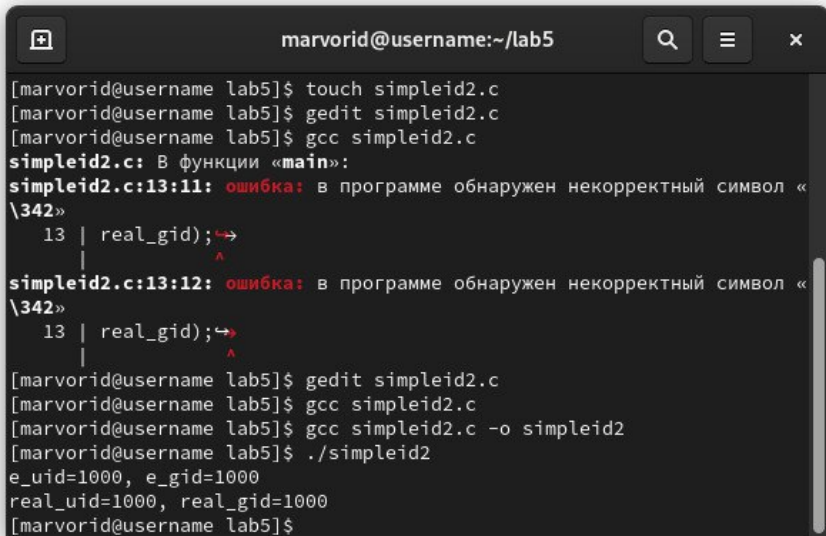
6. Усложним программу, добавив вывод действительных идентификаторов Получившуюся программу назовем simpleid2.c.



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

Рис. 5: Программа simpleid2

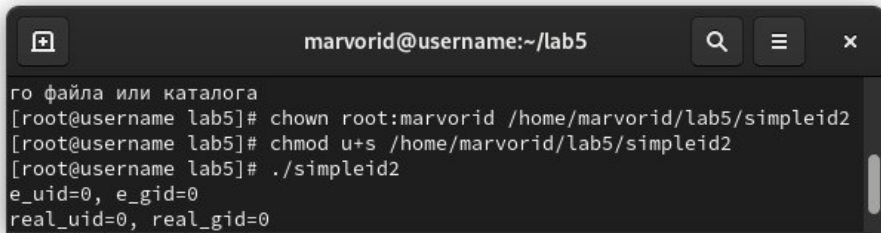
7. Скомпилируем и запустим simpleid2.c: gcc simpleid2.c -o simpleid2 ./simpleid2.



```
marvorid@username:~/lab5

[marvorid@username lab5]$ touch simpleid2.c
[marvorid@username lab5]$ gedit simpleid2.c
[marvorid@username lab5]$ gcc simpleid2.c
simpleid2.c: В функции «main»:
simpleid2.c:13:11: ошибка: в программе обнаружен некорректный символ «\342»
    13 | real_gid);↵
        |           ^
simpleid2.c:13:12: ошибка: в программе обнаружен некорректный символ «\342»
    13 | real_gid);↵
        |           ^
[marvorid@username lab5]$ gedit simpleid2.c
[marvorid@username lab5]$ gcc simpleid2.c
[marvorid@username lab5]$ gcc simpleid2.c -o simpleid2
[marvorid@username lab5]$ ./simpleid2
e_uid=1000, e_gid=1000
real_uid=1000, real_gid=1000
[marvorid@username lab5]$
```

8. От имени суперпользователя выполним команды: `chown root:guest /home/guest/simpleid2`  
`chmod u+s /home/guest/simpleid2`

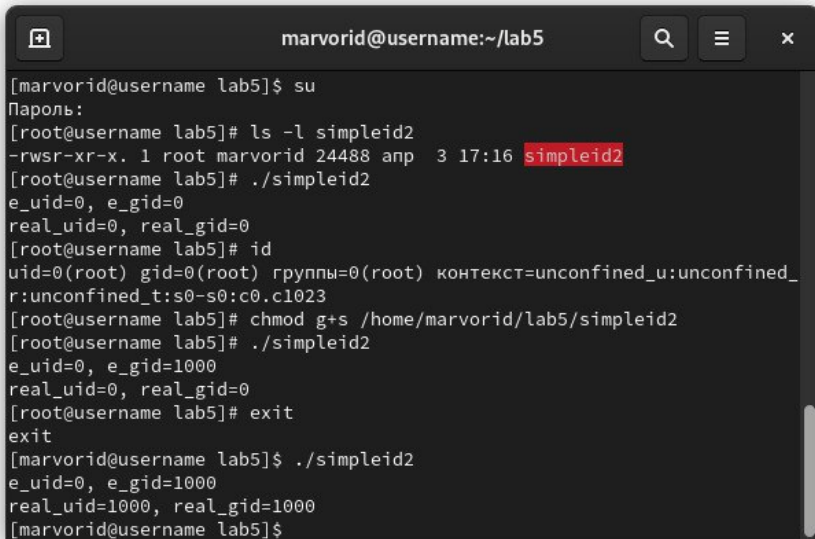


A terminal window with a dark background and light text. The title bar shows 'marvoriid@username:~/lab5'. The terminal content shows a user switching to root and running two commands to change ownership and permissions of a file named 'simpleid2'. The output shows the user is now root (e\_uid=0, e\_gid=0) and the file's real user and group IDs are also 0.

```
marvoriid@username:~/lab5
go файла или каталога
[root@username lab5]# chown root:marvoriid /home/marvoriid/lab5/simpleid2
[root@username lab5]# chmod u+s /home/marvoriid/lab5/simpleid2
[root@username lab5]# ./simpleid2
e_uid=0, e_gid=0
real_uid=0, real_gid=0
```

Рис. 7: Выполнение команд

9. Используем `su` для повышения своих прав.
10. Выполним проверку правильности установки новых атрибутов и смены владельца файла `simpleid2`: `ls -l simpleid2`
11. Запустим `simpleid2` и `id`: `./simpleid2 id` Результаты выполнения теперь немного отличаются
12. Прделаем тоже самое относительно `SetGID`-бита. .

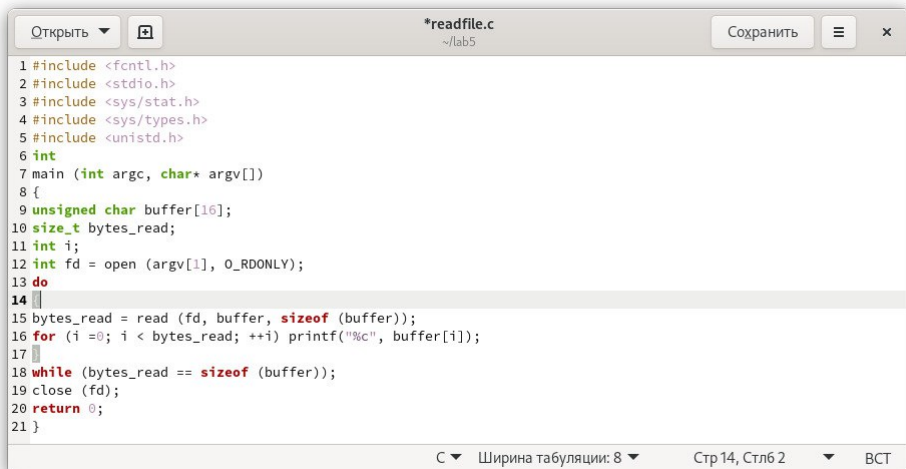


```
marvolid@username:~/lab5
[marvolid@username lab5]$ su
Пароль:
[root@username lab5]# ls -l simpleid2
-rwsr-xr-x. 1 root marvolid 24488 anp  3 17:16 simpleid2
[root@username lab5]# ./simpleid2
e_uid=0, e_gid=0
real_uid=0, real_gid=0
[root@username lab5]# id
uid=0(root) gid=0(root) группы=0(root) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@username lab5]# chmod g+s /home/marvolid/lab5/simpleid2
[root@username lab5]# ./simpleid2
e_uid=0, e_gid=1000
real_uid=0, real_gid=0
[root@username lab5]# exit
exit
[marvolid@username lab5]$ ./simpleid2
e_uid=0, e_gid=1000
real_uid=1000, real_gid=1000
[marvolid@username lab5]$
```

Рис. 8: Выполнение команд



### 13. Создадим программу readfile.c.



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

Рис. 9: программа readfile.c

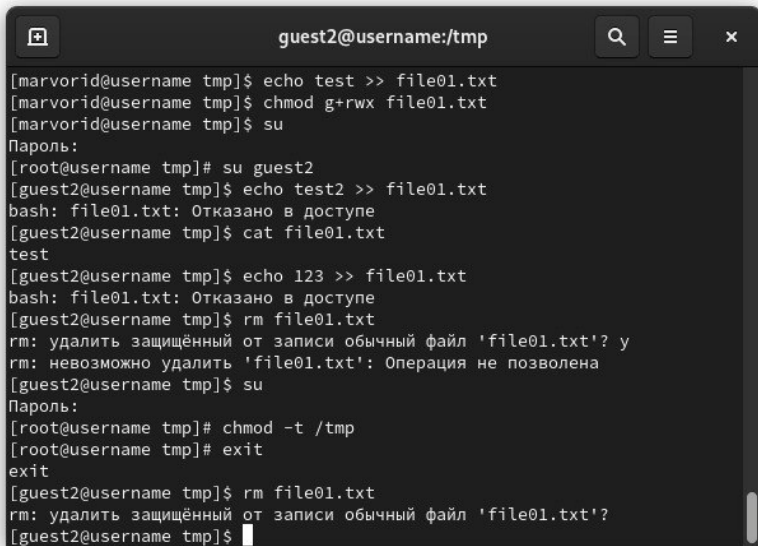
14. Откомпилируем её. `gcc readfile.c -o readfile`.
15. Сменим владельца у файла `readfile.c` и изменим права так, чтобы только суперпользователь (`root`) мог прочитать его, а пользователь не мог.
16. Проверили, что пользователь не может прочитать файл `readfile.c`.
17. Сменим у программы `readfile` владельца и установим SetU'D-бит.
18. Проверили, может ли программа `readfile` прочитать файл `readfile.c`?
19. Проверили, может ли программа `readfile` прочитать файл `/etc/shadow`

```
marvolid@username:~/lab5
[marvolid@username lab5]$ touch readfile.c
[marvolid@username lab5]$ gedit readfile.c
[marvolid@username lab5]$ gcc readfile.c
[marvolid@username lab5]$ gcc readfile.c -o readfile
[marvolid@username lab5]$ su
Пароль:
[root@username lab5]# chown root:root readfile
[root@username lab5]# chmod u+s readfile
[root@username lab5]# chmod -rwx readfile.c
[root@username lab5]# exit
exit
[marvolid@username lab5]$ cat readfile.c
cat: readfile.c: Отказано в доступе
[marvolid@username lab5]$ ./readfile
XXXXXXXXXXXXXXXXXXXX@b2 PPP PP
                000b2 00>@p 0000P` 0n 000 0PHM
                                     t00 0b2 00000X 000
000.20 @ 0000p@0b2 00000b2 00b2 00b2 00b2 00b2 00mD2 000b2 000b2 000b2 000b2 000b2 000b2
00
E2 00 E2 00 E2 00XE2 00E2 00 E2 000E2 000E2 000E2 000E2 00F2 00F2 00
```

Рис. 10: Скомпилируем программу readfile.c

1. Выяснили, что установлен атрибут Sticky на директории /tmp, для чего выполните команду `ls -l / | grep tmp`
2. От имени пользователя создали файл file01.txt в директории tmp со словом test: `echo "test" > /tmp/file01.txt`
3. Просмотрели атрибуты у только что созданного файла и разрешили чтение и запись для категории пользователей «все остальные»: `ls -l /tmp/file01.txt chmod o+rw /tmp/file01.txt ls -l /tmp/file01.txt`
4. От пользователя guest2 (не являющегося владельцем) попробовали прочитать файл /tmp/file01.txt: `cat /tmp/file01.txt`
5. От пользователя guest2 попробовали дозаписать в файл /tmp/file01.txt слово test2 командой `echo "test2" > /tmp/file01.txt` операцию не удалась
6. Проверили содержимое файла командой `cat /tmp/file01.txt`

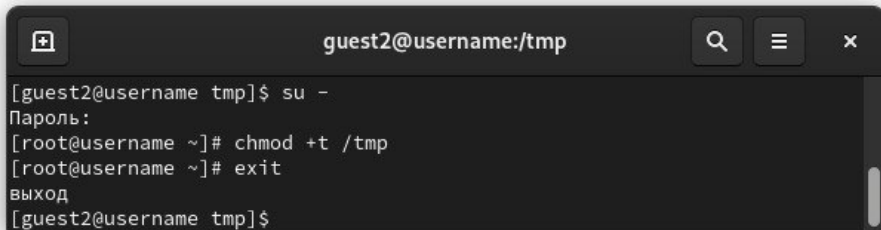
7. От пользователя guest2 попробовали записать в файл /tmp/file01.txt слово test3, степев при этом всю имеющуюся в файле информацию ко- мандой echo "test3" > /tmp/file01.txt
8. Проверили содержимое файла командой cat /tmp/file01.txt
9. От пользователя guest2 попробовали удалить файл /tmp/file01.txt командой rm /tmp/file0l.txt, но получили отказ
10. Повысили свои права до суперпользователя следующей командой su - и выполнили после этого команду, снимающую атрибут t (Sticky-бит) с директории /tmp: chmod -t /tmp
11. Покинули режим суперпользователя командой exit
12. От пользователя guest2 проверили, что атрибута t у директории /tmp нет: ls -l / | grep tmp
13. Повторили предыдущие шаги.
14. Удалось удалить файл от имени пользователя не являющегося владельцем.



```
guest2@username:/tmp
[marvorid@username tmp]$ echo test >> file01.txt
[marvorid@username tmp]$ chmod g+rwX file01.txt
[marvorid@username tmp]$ su
Пароль:
[root@username tmp]# su guest2
[guest2@username tmp]$ echo test2 >> file01.txt
bash: file01.txt: Отказано в доступе
[guest2@username tmp]$ cat file01.txt
test
[guest2@username tmp]$ echo 123 >> file01.txt
bash: file01.txt: Отказано в доступе
[guest2@username tmp]$ rm file01.txt
rm: удалить защищённый от записи обычный файл 'file01.txt'? y
rm: невозможно удалить 'file01.txt': Операция не позволена
[guest2@username tmp]$ su
Пароль:
[root@username tmp]# chmod -t /tmp
[root@username tmp]# exit
exit
[guest2@username tmp]$ rm file01.txt
rm: удалить защищённый от записи обычный файл 'file01.txt'?
[guest2@username tmp]$
```

Рис. 11: Выполнение команд

15. Повысили свои права до суперпользователя и вернули атрибут t на директорию /tmp:  
su - chmod +t /tmp exit

A terminal window with a dark background and light text. The title bar at the top reads 'guest2@username:/tmp' and includes standard window controls (minimize, maximize, close) on the right. The terminal content shows a user running 'su -' to become root, prompted for a password, then running 'chmod +t /tmp' and 'exit' to return to the original user. The prompt changes from '\$' to '#' when root is achieved. The word 'выход' (exit) is written in Russian. The final prompt is '[guest2@username tmp]\$'.

```
guest2@username:/tmp  
[guest2@username tmp]$ su -  
Пароль:  
[root@username ~]# chmod +t /tmp  
[root@username ~]# exit  
выход  
[guest2@username tmp]$
```

Рис. 12: Повышение прав

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



- Запоминается последняя фраза. © Штирлиц

...