Лабораторная работа №5

Дискреционное разграничение прав в Linux. Исследование влияния дополнительных атрибутов.

Ли Тимофей Александрович

Содержание

[Цель работы 1](#_Toc87572644)

[Выполнение лабораторной работы 1](#_Toc87572645)

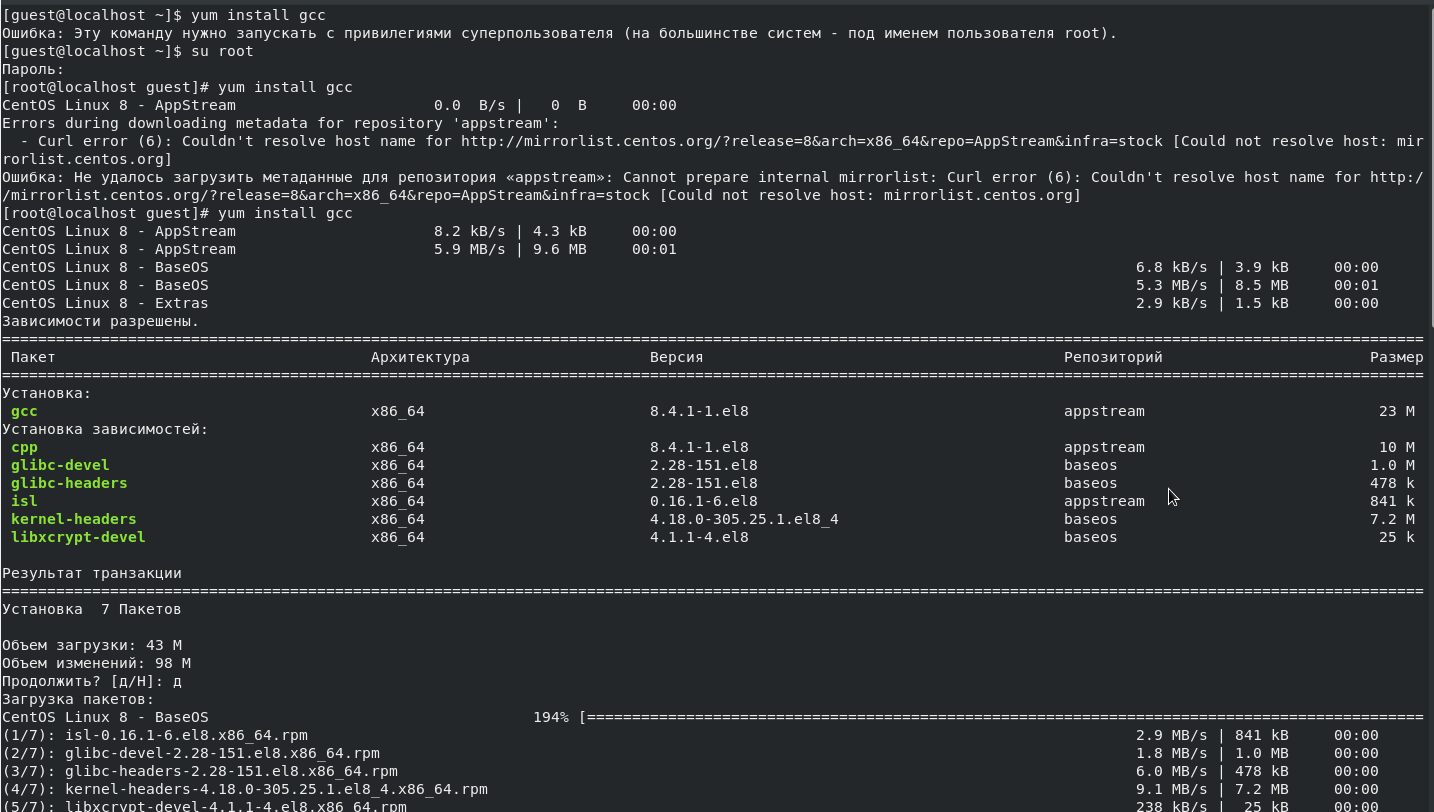
[Выводы 6](#_Toc87572646)

# 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.

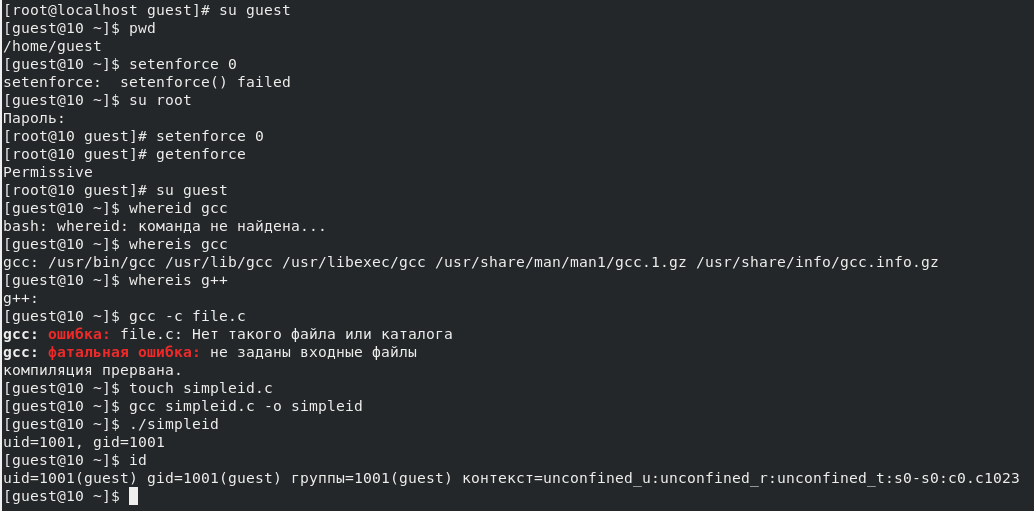
# Выполнение лабораторной работы

Для начала я, зайдя в root, установил компилятор gcc со всеми сопутствующими пакетами. (рис. @fig:001):



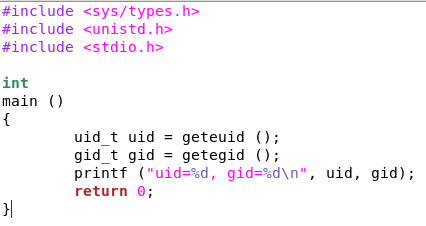
установка gcc

Затем, установил setenforce 0, проверил запуск gcc и g++. Создал файл simpleid.c, скомпилировал его и запустил. Сравнил вывод с выводом команды id: (рис. @fig:002)



создание simpleid.c

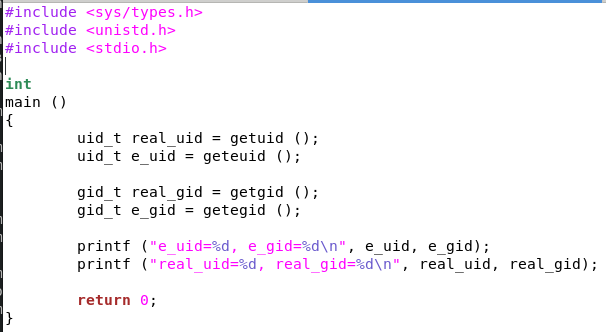
Код файла simpleid.c: (рис. @fig:003)



simpleid.c

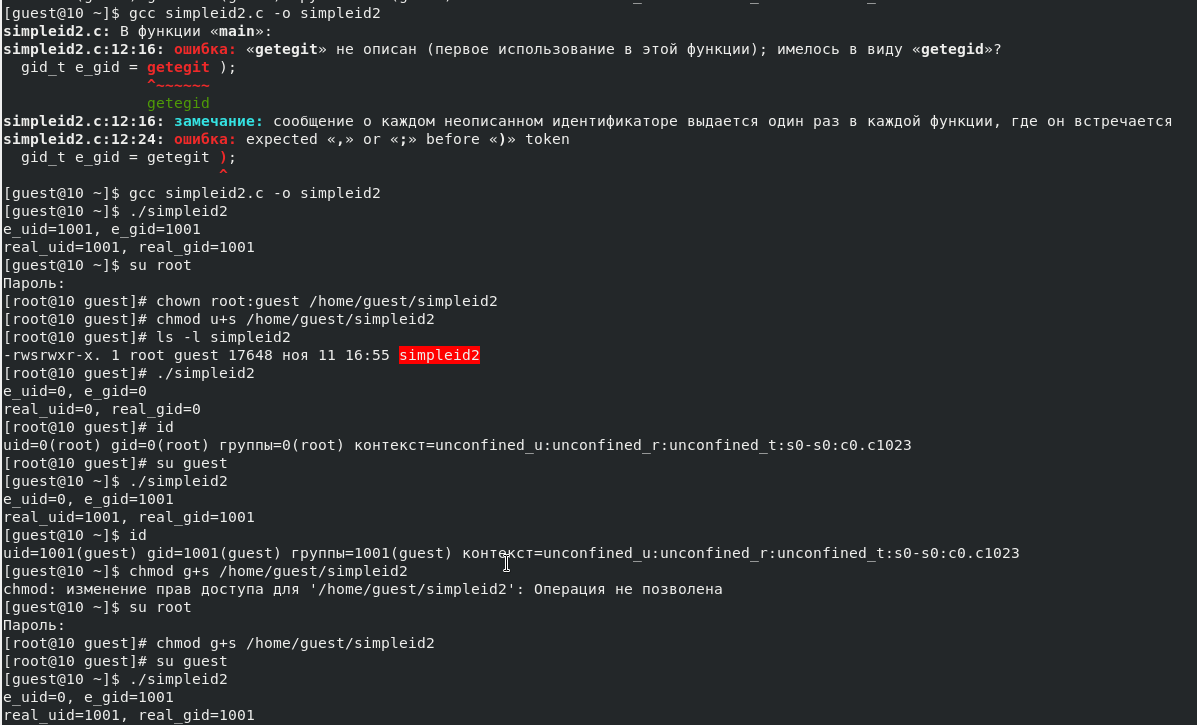
Как видим, вывод программы и команды id совпадают.

Добавил в код вывод действительных идентификаторов и сохранил файл как simpleid2.c: (рис. @fig:004)



simpleid2.c

Скомпилировал код и от имени root изменил владельца полученной программы на root и группу guest, а также добавил владельцу программы право на выполнение от имени суперпользователя. Затем, убедился в правильности установки атрибутов с помощью ls -l, запустил программу и вызвал команду id. Далее, с помощью root установил на программу для группы право выполнения от имени суперпользователя (SetGID бит) и повторил запуск программы (конец действий на скриншоте 6): (рис. @fig:005)



работа с SetUID и SetGID битами

Затем, написал программу readfile и скомпилировал ее: (рис. @fig:006)



readfile.c

Далее Сменил владельца программы на root и сменил права так, что только суперпользователь может читать этот файл. Убедился, что guest не может прочитать файл. Сменил владельца у скомпилированной программы и установил SetUID бит. Попробовал от имени guest прочитать файл readfile.c с помощью программы: (рис. @fig:007)



действия с readfile

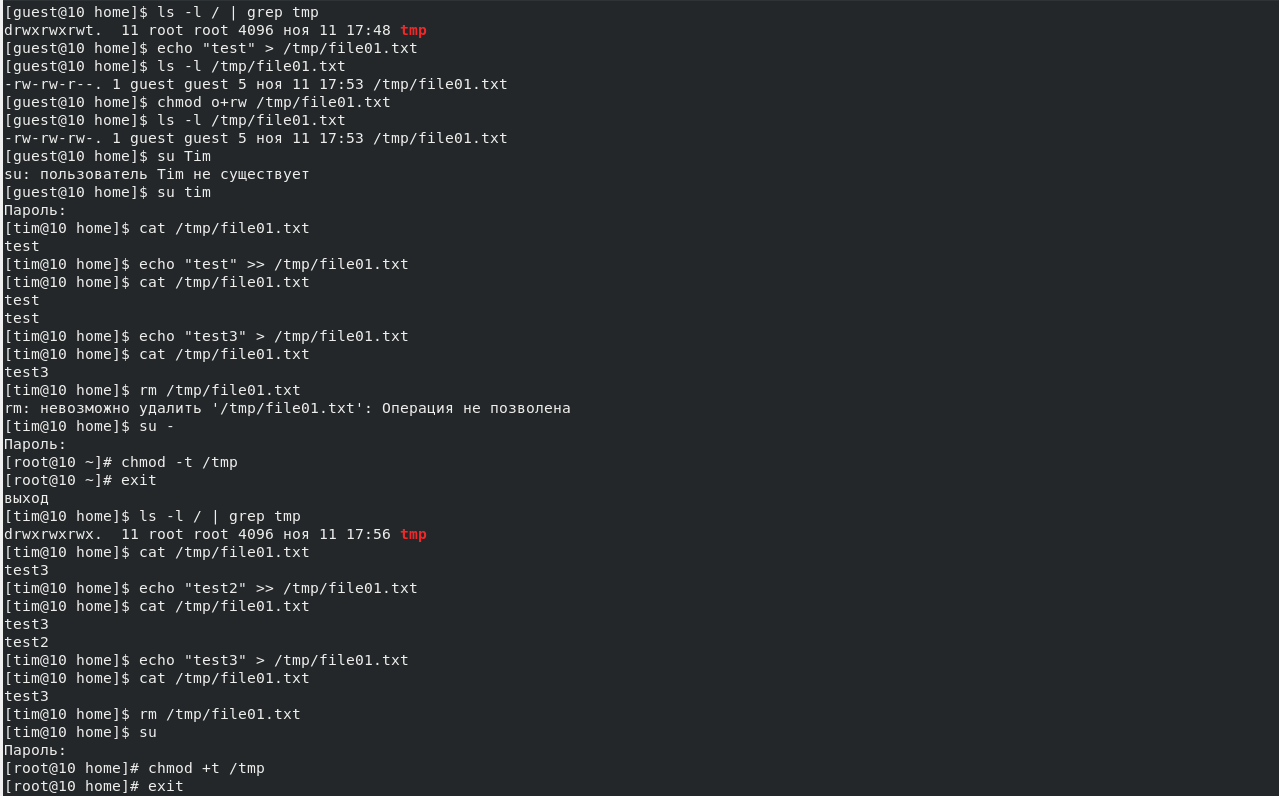
Также попробовал прочитать файл etc/shadow: (рис. @fig:008)



чтение etc/shadow

Оба файла прочитались несмотря на отсутствие прав у пользователя guest, поскольку программа запускается пользователем guest от имени суперпользователя.

Далее, начал работать с битом Sticky: (рис. @fig:009)



действия с Sticky-битом

Выяснил, что на директорию tmp установлен этот бит, тк среди атрибутов в конец есть t. Создал файл file01.txt в директории tmp и записал в него слово test. Посмотрел расширенные атрибуты и добавил чтение и запись для других пользователей. От пользователя tim (не явл. владельцем) прочитал файл, дозаписал в него и перезаписал его. Затем попробовал удалить файл, но безуспешно. Далее от имени root удалил атрибут t (Sticky бит) с директории tmp и повторил все действия от имени пользователя tim. На этот раз вышло то же самое, но еще удалось удалить файл. Это потому, что Sticke-бит защищает файлы, запрещая удаление файлов не владельцами. В конце вернул Sticky бит директории tmp.

# Выводы

Изучил механизмы изменения идентификаторов, применения 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.