**Завдання 9.1** Напишіть програму, яка читає файл /etc/passwd за допомогою команди getent passwd, щоб дізнатись, які облікові записи визначені на вашому комп’ютері.  
 Програма повинна визначити, чи є серед них звичайні користувачі (ідентифікатори UID повинні бути більші за 500 або 1000, залежно від вашого дистрибутива), окрім вас.

#include <stdio.h>

#include <stdlib.h>

#include <pwd.h>

#include <unistd.h> // додано для getuid()

int main() {

    struct passwd \*pw;

    while ((pw = getpwent()) != NULL) {

        if (pw->pw\_uid >= 1000 && pw->pw\_uid != getuid()) {

            printf("Звичайний користувач: %s (UID: %d)\n", pw->pw\_name, pw->pw\_uid);

        }

    }

    endpwent();

    return 0;

}



**Завдання 9.2** Напишіть програму, яка виконує команду cat /etc/shadow від імені адміністратора, хоча запускається від звичайного користувача.  
 (Ваша програма повинна робити необхідне, виходячи з того, що конфігурація системи дозволяє отримувати адміністративний доступ за допомогою відповідної команди.)

#include <stdio.h>

#include <stdlib.h>

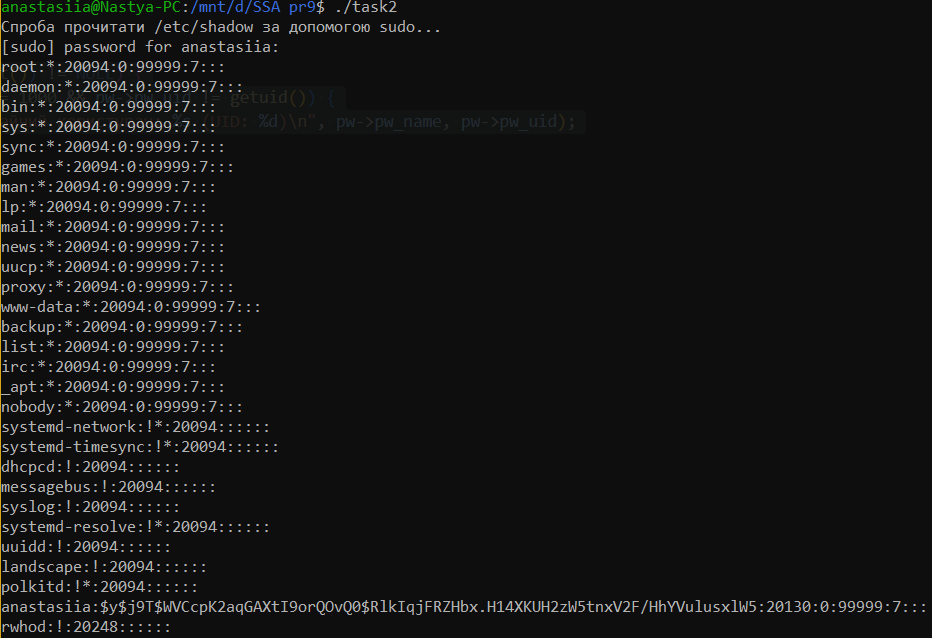
int main() {

    printf("Спроба прочитати /etc/shadow за допомогою sudo...\n");

    system("sudo cat /etc/shadow");

    return 0;

}



**Завдання 9.3** Напишіть програму, яка від імені root копіює файл, який вона перед цим створила від імені звичайного користувача. Потім вона повинна помістити копію у домашній каталог звичайного користувача.  
 Далі, використовуючи звичайний обліковий запис, програма намагається змінити файл і зберегти зміни. Що відбудеться?  
 Після цього програма намагається видалити цей файл за допомогою команди rm. Що відбудеться?

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main() {

    // Створення файлу як звичайний користувач

    system("echo 'Test file' > /tmp/original.txt");

    // Копіювання від root (припускаємо, що sudo працює)

    system("sudo cp /tmp/original.txt $HOME/copied\_by\_root.txt");

    // Спроба змінити файл

    printf("Спроба змінити файл...\n");

    system("echo 'New line' >> $HOME/copied\_by\_root.txt");

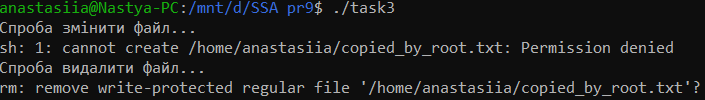
    // Спроба видалити

    printf("Спроба видалити файл...\n");

    system("rm $HOME/copied\_by\_root.txt");

    return 0;

}



**Завдання 9.4** Напишіть програму, яка по черзі виконує команди whoami та id, щоб перевірити стан облікового запису користувача, від імені якого вона запущена.  
 Є ймовірність, що команда id виведе список різних груп, до яких ви належите. Програма повинна це продемонструвати.

#include <stdio.h>

#include <stdlib.h>

int main() {

    printf("Користувач (whoami):\n");

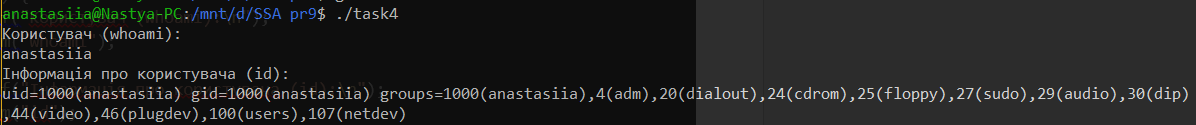
    system("whoami");

    printf("Інформація про користувача (id):\n");

    system("id");

    return 0;

}



**Завдання 9.5** Напишіть програму, яка створює тимчасовий файл від імені звичайного користувача. Потім від імені суперкористувача використовує команди chown і chmod, щоб змінити тип володіння та права доступу.  
 Програма повинна визначити, в яких випадках вона може виконувати читання та запис файлу, використовуючи свій обліковий запис.

#include <stdio.h>

#include <stdlib.h>

int main() {

    system("echo 'temp data' > /tmp/temp\_file.txt");

    // Зміна власника і прав (як root)

    system("sudo chown root:root /tmp/temp\_file.txt");

    system("sudo chmod 600 /tmp/temp\_file.txt");

    printf("Спроба прочитати як звичайний користувач:\n");

    system("cat /tmp/temp\_file.txt");

    return 0;

}



**Завдання 9.6** Напишіть програму, яка виконує команду ls -l, щоб переглянути власника і права доступу до файлів у своєму домашньому каталозі, в /usr/bin та в /etc.  
 Продемонструйте, як ваша програма намагається обійти різні власники та права доступу користувачів, а також здійснює спроби читання, запису та виконання цих файлів.

#include <stdio.h>

#include <stdlib.h>

int main() {

    printf("Домашній каталог:\n");

    system("ls -l $HOME");

    printf("\n/usr/bin:\n");

    system("ls -l /usr/bin | head -n 10");

    printf("\n/etc:\n");

    system("ls -l /etc | head -n 10");

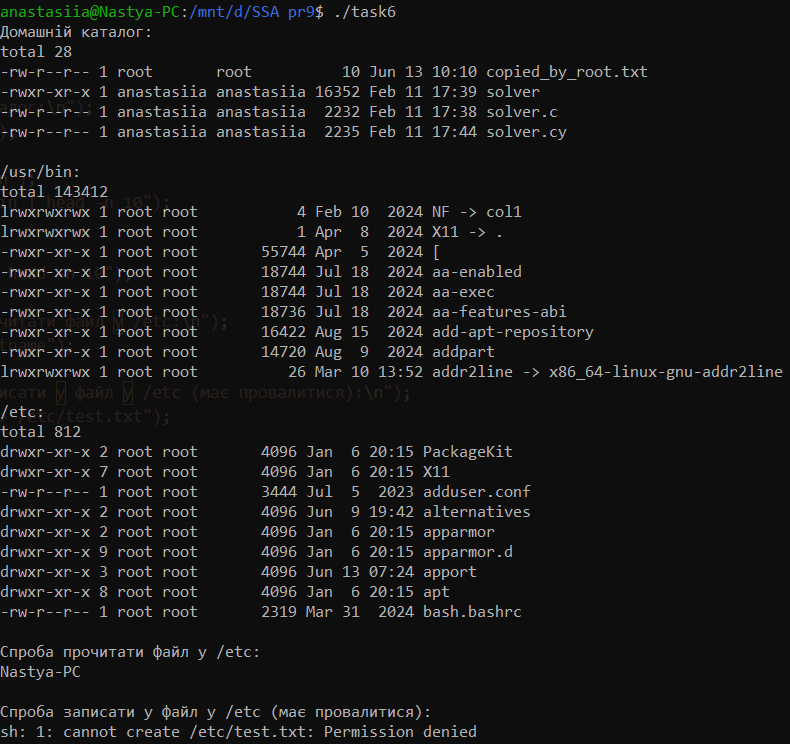
    printf("\nСпроба прочитати файл у /etc:\n");

    system("cat /etc/hostname");

    printf("\nСпроба записати у файл у /etc (має провалитися):\n");

    system("echo 'test' > /etc/test.txt");

    return 0;

}

### Як змінюється доступ до файлу, якщо його жорстке посилання видалено, але процес ще використовує файл?

### Що відбувається з доступом до файлу при видаленні його жорсткого посилання, якщо процес його ще відкрив?

* **Жорстке посилання (hard link)** — це ім'я файлу в файловій системі, яке вказує на певний індексний дескриптор (inode).
* Коли ви **видаляєте жорстке посилання** (наприклад, rm filename), ви фактично видаляєте *ім’я* файлу з каталогу, але **фізичні дані на диску не видаляються, поки існує хоча б одне посилання або відкритий дескриптор.**
* Якщо **процес вже відкрив файл** (має дескриптор), то:
  + Файл залишається в системі *до тих пір*, поки процес його не закриє.
  + Хоча файл **не має імені в каталозі**, він продовжує існувати фізично і бути доступним для процесу через відкритий дескриптор.
  + Інші процеси **не зможуть відкрити файл за іменем**, бо посилання видалено.

**Висновок:**

* **Доступ для процесу, який відкрив файл раніше, не змінюється.** Він може і надалі читати, писати, або виконувати файл.
* Файл **фактично "прихований" для інших** — його ім’я зникло з файлової системи.
* Після закриття дескриптора файлу ОС звільнить простір на диску.

**Аналогія:**

Видалення жорсткого посилання — це як забрати книгу з полиці (ім’я), але якщо хтось тримає цю книгу в руках (дескриптор), вона ще не зникла, доки він її не відкладе.