Відокремлений структурний підрозділ «**Волинський фаховий коледж Національного університету харчових технологій**»

Освітньо-професійна програма: **Інженерія програмного забезпечення**

**ОК Системне програмування**

**Звіт лабораторної роботи №1**  
**Тема “Застосування системних викликів getuid() i getgid(), найпростіша системна програма в UNIX”**

Студентки 4-того курсу   
Групи ІПЗ-41  
**Павліхи І.В**

**Луцьк 2025**

Лабораторна робота №1

**Тема: Застосування системних викликів getuid() і getgid(), найпростіша системна програма в UNIX**

**Мета роботи**

Заволодіти поняттям найпростіша програма в операційній системі сімейства UNIX. Навчитися отримувати ідентифікатори співрідних процесів та користувачів у системі, використовуючи системні виклики.

**Завдання 1.**

*Створити найпростішу програму з використанням системного виклику getuid() для отримання ідентифікатора процесу користувача.*

**#include <stdio.h>**

**#include <sys/types.h>**

**#include <unistd.h>**

**int main() {**

**uid\_t i;**

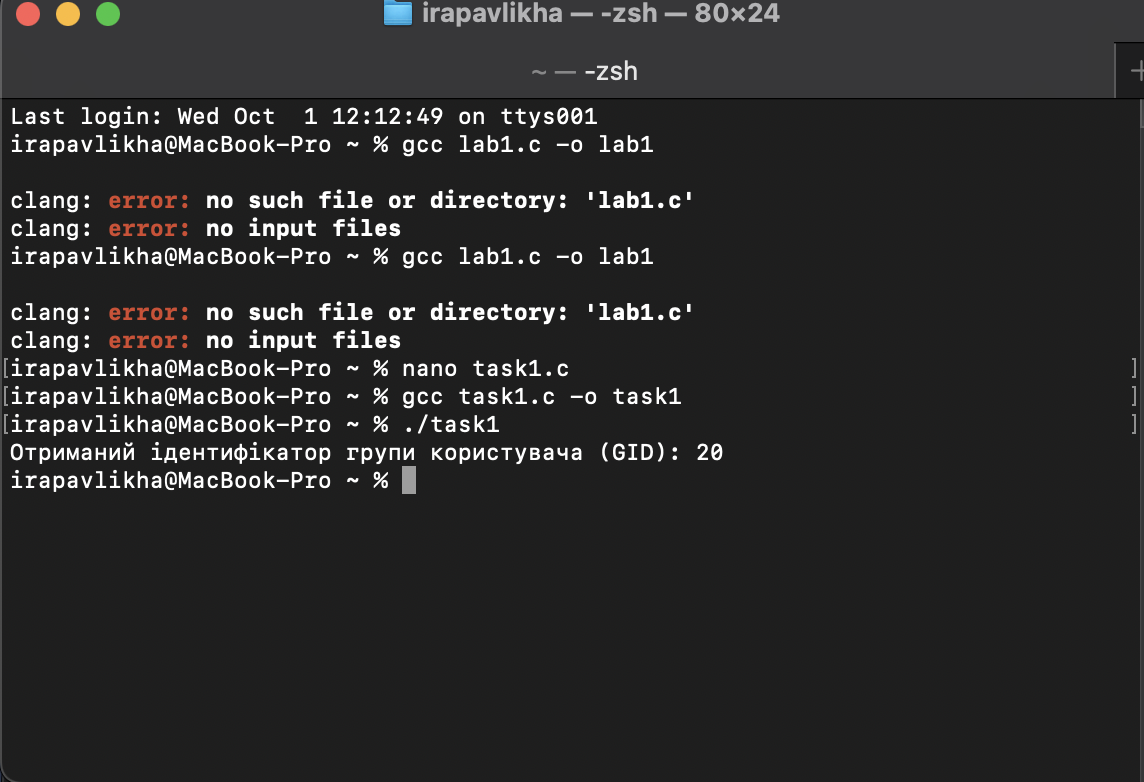
**i = getuid();**

**printf("Отриманий ідентифікатор користувача (UID): %d\n", (int) i);**

**return 0;**

**}**

У цій програмі використано системний виклик getuid(), який повертає унікальний числовий ідентифікатор користувача (UID), від імені якого виконується процес. Бібліотека <sys/types.h> визначає тип uid\_t, а <unistd.h> містить оголошення самої функції getuid(). Результат виводиться через функцію printf(). Приведення (int)i використовується, щоб явно відобразити значення UID у числовому форматі.



Мал 1.1 - Виконання програми 1

**Головні моменти:**

* Функція getuid() — системний виклик UNIX.
* UID визначає користувача, який запустив програму.
* Для виводу використано приведення типу uid\_t → int.

**Завдання 2.**

*Створити найпростішу програму з використанням системного виклику getgid() для отримання ідентифікатора групи користувачів.*

**#include <stdio.h>**

**#include <sys/types.h>**

**#include <unistd.h>**

**int main() {**

**gid\_t g;**

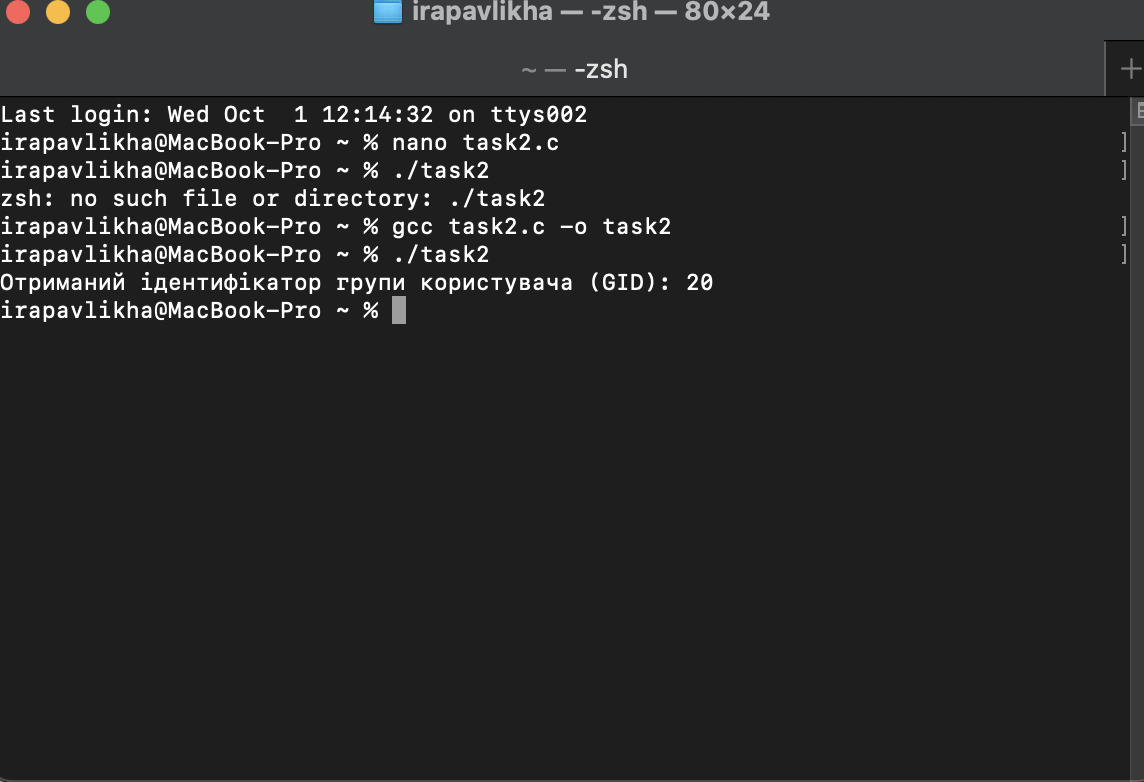
**g = getgid();**

**printf("Отриманий ідентифікатор групи користувача (GID): %d\n", (int) g);**

**return 0;**

**}**

Програма виконує системний виклик getgid(), який повертає ідентифікатор групи (GID) користувача, до якої він належить. Бібліотеки <sys/types.h> і <unistd.h> використовуються так само, як у попередньому завданні. Вивід здійснюється через printf().



Мал 1.2 - Виконання програми 2

**Головні моменти:**

* GID — це груповий ідентифікатор користувача.
* getgid() дозволяє дізнатися, до якої групи належить поточний користувач.
* Приведення типу використовується для коректного відображення значення.

**Завдання 3.**

*Створити найпростішу програму для отримання всіх ідентифікаторів користувача в системі, враховуючи діючі процеси.*

**#include <stdio.h>**

**#include <sys/types.h>**

**#include <unistd.h>**

**int main() {**

**uid\_t uid = getuid();**

**gid\_t gid = getgid();**

**pid\_t pid = getpid();**

**pid\_t ppid = getppid();**

**printf("UID користувача: %d\n", (int) uid);**

**printf("GID групи: %d\n", (int) gid);**

**printf("PID процесу: %d\n", (int) pid);**

**printf("PPID (батьківський PID): %d\n", (int) ppid);**

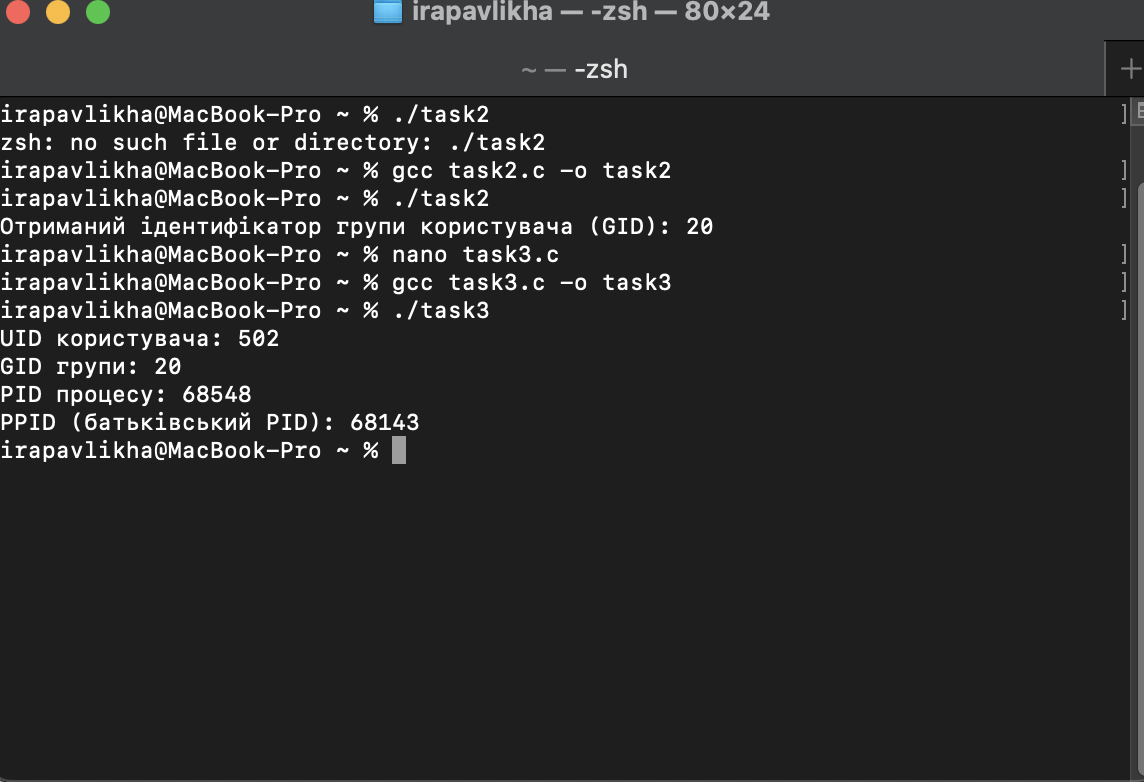
**return 0;**

**}**

У цій програмі реалізовано одразу кілька системних викликів, які дозволяють отримати основні ідентифікатори користувача та процесів у системі UNIX.

Виклик getuid() повертає унікальний ідентифікатор користувача (UID), під яким виконується програма.  
Виклик getgid() отримує ідентифікатор групи (GID), до якої належить користувач.  
Далі функція getpid() повертає ідентифікатор поточного процесу (PID), що дозволяє визначити кожну окрему програму, запущену в операційній системі.  
Водночас функція getppid() надає ідентифікатор батьківського процесу (PPID), тобто того, який створив поточний процес.

Завдяки цим чотирьом системним викликам можна отримати повну інформацію про зв’язок між користувачем, групою та процесами. Усі отримані значення виводяться на екран за допомогою функції printf(). Оскільки системні типи uid\_t, gid\_t і pid\_t можуть відрізнятися від стандартного цілого типу int, у коді використано операцію приведення типів (int), щоб значення коректно відображалися при виведенні.



Мал 1.3 - Виконання програми 3

**Головні моменти:**

* getpid() і getppid() — дозволяють відстежувати зв’язки між процесами.
* Кожен процес у UNIX має унікальний PID і посилання на PPID.
* Код демонструє зв’язок між користувачем, групою та процесами.

**Пояснення до питань самоконтролю**

1. **sys/types.h** — бібліотека, що визначає типи даних, які використовуються у системних викликах (uid\_t, gid\_t, pid\_t).
2. **unistd.h** — містить оголошення системних викликів UNIX (наприклад, getuid(), getpid() тощо).
3. **getuid()** — повертає UID користувача, який запустив процес.
4. **getgid(), getpid(), getppid()** — відповідно повертають GID користувача, PID поточного процесу та PID батьківського процесу.
5. **Типи uid\_t і gid\_t** — це синоніми одного з цілих типів, зазвичай unsigned int.
6. **Типи pid\_t і ppid\_t** — синоніми типу int.
7. **Операція приведення типів** — це спосіб вказати компілятору, як інтерпретувати дані іншого типу. Наприклад:

printf("%d", (int) uid);

1. **ESC-символи у printf:**
   1. \n — перехід на новий рядок
   2. \t — табуляція
   3. [\\](file:///\\) — вивід зворотного слеша
   4. \" — вивід лапок

**Висновок**

У ході виконання лабораторної роботи було розглянуто найпростіші системні програми в UNIX. Я навчилася використовувати бібліотеки <sys/types.h> та <unistd.h>, отримувати ідентифікатори користувача, групи та процесів. Також досліджено типи даних uid\_t, gid\_t, pid\_t, операції приведення типів і форматований вивід із застосуванням ESC-символів. Робота заклала основу розуміння системних викликів та їхнього використання у розробці програм під UNIX-системи.