

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ  
УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**

**ІКНІ**  
Кафедра ПЗ



**ЗВІТ**

До лабораторної роботи №6

**на тему:** “Багатопоточність в операційній системі Linux. Створення, керування та синхронізація потоків”

**з дисципліни:** “Операційні системи”

**Лектор:**  
старший викладач кафедри ПЗ  
Грицай О.Д.

**Виконав:**  
студент групи ПЗ-24  
Губик А. С.

**Прийняв:**  
доцент кафедри ПЗ  
Горечко О. М.

**Тема роботи:**Багатопоточність в операційній системі Linux. Створення, керування та синхронізація потоків

**Мета роботи:**Навчитися створювати потоки та керувати ними в операційній системі Linux. Ознайомитися з методами синхронізації потоків в операційній системі Linux. Навчитися реалізовувати багатопоточний алгоритм розв'язку задачі з використанням синхронізації в операційній системі Linux.

## Теоретичні відомості

Потоки в операційній системі Linux. Поняття потоку, як одиниці виконання процесу в операційній системі (ОС) Linux, аналогічне як і у ОС Windows. Кожен процес можна представити як контейнер ресурсів і послідовність інструкцій виконання. Таким чином, можна сказати, що кожен процес містить хоча б один потік виконання, якому надані лічильник інструкцій, регістри і стек. Крім того, у кожному процесі можна створити додаткові гілки виконання – потоки, тоді такий процес називають багатопоточним. Різниця між потоками в ОС Linux і в ОС Windows полягає у їх представленні в ядрі операційних систем. В ОС Windows потоки виконання у режимі користувача зіставляються з потоками у режим ядра. У перших версіях ядра Linux потоки користувача зіставлялись з процесами у ядрі. Створення потоку відбувалось з допомогою системного виклику `clone()`. Виклик `clone()`, як і `fork()` дозволяє створювати новий процес, але з певними відмінностями : - одразу повне копіювання батьківського процесу - створення власного стеку - необхідно вказати спеціальний набір прапорців успадкування для того, щоб визначити, як будуть розподілятися ресурси (адресний простір, відкриті файли, обробники сигналів) між батьківським і дочірнім процесом. Таким чином, створювався новий потік у режимі користувача, який відобрається у процес ядра. Відображення здійснюється за моделлю 1:1. Оскільки керуючий блок процесу в Linux представлений в ядрі структурою Фактично у ядрі потоки і процеси не розрізнялися. Але системний виклик `clone()` не підтримувався стандартом POSIX і тому розроблялись бібліотеки потоків, що дозволяли працювати з потоками, використовуючи `clone()` з різними атрибутами виконання. У сучасних версіях Linux підтримуються спеціальні об'єкти ядра - потоки ядра, побудованні на змінному і розширеному системному виклику `clone()`. Підтримка потоків здійснюється через POSIX-сумісну бібліотеку NPTL (Native POSIX Threads Library). Типи даних і функції, що застосовуються до потоків

## Індивідуальне завдання

3. Бінарний пошук заданого елементу масиву (кількість елементів >10000, елементи рандомні). Вивести значення та індекс. (Синхронізація: м'ютекс, умовні змінні)

I. Розпаралелення бінарного пошуку на 2, 4, 8, 16 потоків

```
● artem@laptop:~/Progs++/OSlabs/Lab6$ ./no_sync
Enter the thread count: 1

Searched index: 7109374
0.000261537
● artem@laptop:~/Progs++/OSlabs/Lab6$ ./no_sync
Enter the thread count: 2

Searched index: 7109374
0.000481798
● artem@laptop:~/Progs++/OSlabs/Lab6$ ./no_sync
Enter the thread count: 4

Searched index: 7109374
0.000438219
○ artem@laptop:~/Progs++/OSlabs/Lab6$
```

Рис. 1: Графік залежності швидкості роботи програми від кількості потоків

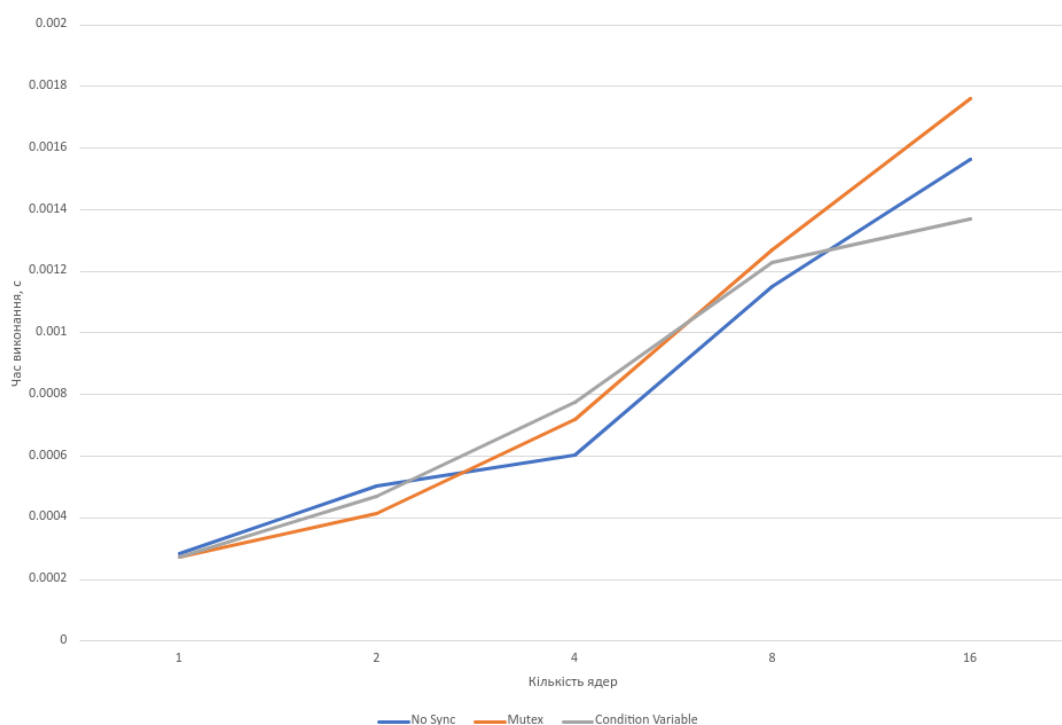


Рис. 2: Графік залежності швидкості роботи програми від кількості потоків

Як бачимо синхронізація сповільнює роботу і не впливає на правильність виконання бінарного пошуку.

## II. Задання пріоритету, відміна та affinity

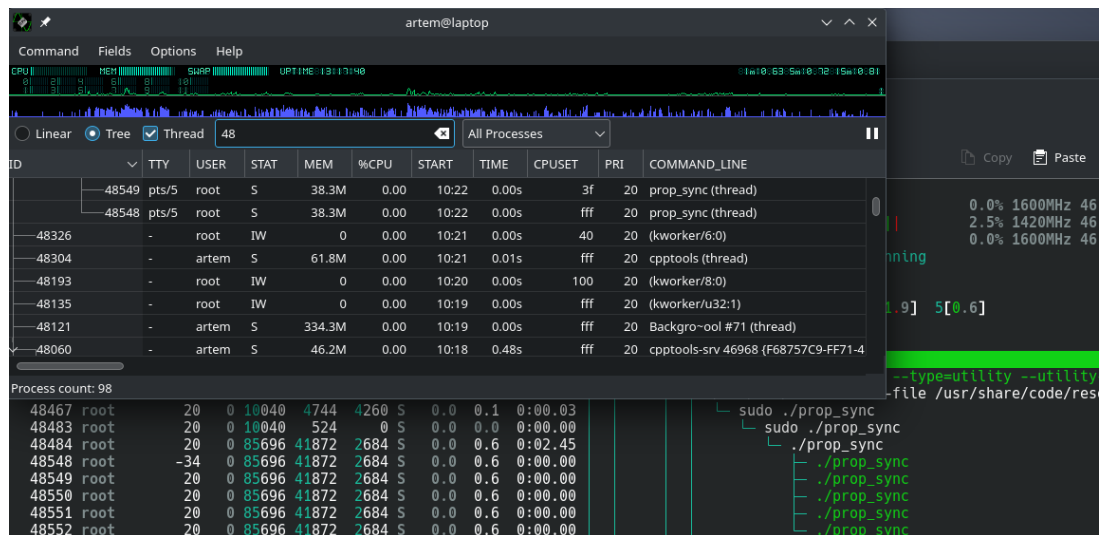


Рис. 3:

**Висновок:** Я ознайомився з бібліотекою POSIX Threads. Ефективність синхронізації залежить від поставленої перед нами задачі. У випадку з бінарним пошуком воно не потрібне і сповільнює роботу.