

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**  
**НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТУ**  
**“ЛЬВІВСЬКА ПОЛІТЕХНІКА”**

**Кафедра систем штучного інтелекту**

**Лабораторна робота № 10**

**з дисципліни**

**«Операційні системи»**

**по темі :**

**“Синхронізація потоків в ОС Linux”**

**Виконав:**

**студент групи КН-209**

**Ярчак А.В**

**Викладач:**

**Кривенчук Ю. П.**

**Львів – 2019р.**

**Мета.** Ознайомитися з особливостями синхронізації потоків в ОС Linux. Навчитися організовувати багатопоточність з використанням синхронізації в ОС Linux.

### Завдання.

1. Реалізувати алгоритм із лабораторної роботи №3.
2. Здійснити розпаралелювання даного алгоритму на 2, 4, 8 потоків із використанням синхронізації.
3. Реалізувати прогрес (хід) виконання задачі.
4. Для синхронізації потоків використати такі методи: м'ютекси, семафори, умовна змінна, очікування, сигнали, монітори.
5. Порівняти результати виконання програми під ОС Windows та Linux.
6. Результати виконання роботи відобразити у звіті.

### Код програми:

```
#include <iostream>
#include <chrono>
#include <cmath>
#include <unistd.h>

#define SIZE 10

using namespace std;
using namespace std::chrono;

struct sched_param param;

struct DATA {
    int *arr, priority;
    DATA(int a[], int p) : arr(a), priority(p) {}
};

void* sort_thread(void *data) {
    auto start = high_resolution_clock::now();
    int policy;
    param.sched_priority = ((DATA *) data)->priority;
    pthread_setschedparam(pthread_self(), SCHED_RR, &param);
    pthread_getschedparam(pthread_self(), &policy, &param);
    //cout << policy << endl;
    int *arr = ((DATA *) data)->arr;
    int i, key, j;
    for (i = 0; i < SIZE; i++) {
        key = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] < key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
    auto stop = high_resolution_clock::now();
    auto duration = duration_cast<microseconds>(stop - start);
    return nullptr;
}

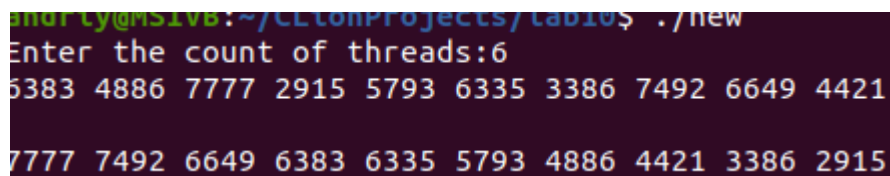
int main() {
```

```

int count;
cout << "Enter the count of threads:";
cin >> count;
int arr[SIZE];
for (int &a : arr) { a = rand() % 9000 + 1000; cout << a << ' ';}
cout << endl;
pthread_t threads[count];
pthread_attr_t attributes[count];
int priorities[] = {};
/*while(true)
{
    int priority, number;
    cout << "Enter number of thread and priority: ";
    cin >> number >> priority;
    if(number == 666) break;
    else priorities[number] = priority;
}*/
for (int i = 0; i < count; i++) {
    pthread_attr_init(&attributes[i]);
    pthread_create(&threads[i], &attributes[i], sort_thread, new DATA(arr,
priorities[i]));
}
for (int i = 0; i < count; i++) pthread_join(threads[i], nullptr);
cout << endl;
for (int &a : arr) cout << a << ' ';
return 0;
}

```

## Результат:



```

andriy@MSIV: ~/CLionProjects/lab10$ ./new
Enter the count of threads:6
5383 4886 7777 2915 5793 6335 3386 7492 6649 4421
7777 7492 6649 6383 6335 5793 4886 4421 3386 2915

```

**Висновок:** на цій лабораторній роботі я ознайомився з особливостями синхронізації потоків в ОС Linux. Навчився організовувати багатопоточність з використанням синхронізації в ОС Linux. Покращив написаний мною алгоритм в попередній лабораторній, а саме синхронізував сортування потоками.