

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут 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 : ст.викл каф.ПЗ

Грицай О.Д.

Виконала: ст.гр.ПЗ-23

Кохман О.В.

Прийняла: ст.викл каф.ПЗ

Грицай О.Д.

« ____ » _____ 2022 р.

Σ _____ .

Львів – 2022

Тема: Багатопоточність в операційній системі LINUX. Створення, керування та синхронізація потоків.

Мета: 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 `task_struct`, то і представлення потоку здійснювалось через `task_struct`. 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, побудовані 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 POSIX, мають префікс `pthread_` і доступні через підключення `pthread.h`.

Індивідуальне завдання

1. Реалізувати заданий (згідно варіанту) алгоритм в окремому потоці.
 2. Виконати розпаралелення заданого алгоритму на 2, 4, 8, 16 потоків.
 3. Реалізувати можливість зміни/встановлення пріоритету потоку (для планування потоків) або встановлення відповідності виконання на ядрі.
 4. Реалізувати можливість зробити потік від'єднаним.
 5. Реалізувати можливість відміни потоку.
 6. Реалізувати синхронізацію потоків за допомогою вказаних методів (згідно варіанту)
 7. Порівняти час виконання задачі відповідно до кількості потоків і методу синхронізації (чи без синхронізації).
 8. Результати виконання роботи оформити у звіт
- 9 варіант : Вивід слів з файлу, що розпочинаються на задану літеру (кількість рядків у файлі > 1000, текст довільна наукова стаття) (Синхронізація: спінлок, умовні змінні).

Код програми

Назва файлу: `main.cpp`

```
#include <pthread.h>
#include <iostream>
#include <fstream>
#include <chrono>
#include <unistd.h>
#include <cstring>
#include <sched.h>

using namespace std;
using namespace std::chrono;

int countThread;
pthread_spinlock_t *spinLock;
pthread_mutex_t mutexLock = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t mutexCond = PTHREAD_COND_INITIALIZER;

pthread_t* threads;
int readyCondition = 0;
void * value_ptr;

typedef struct {
    string* array;
```

```

    int size;
} th_param; //структура передачі параметрів

void asyncJob(th_param params);
void spinLockJob(th_param params);
void conditionalVariablesJob(th_param params);
void setPriorityJob(th_param params);
void otherThings(th_param params);

void search(string* array, int start, int end, char value) {
    for (int i = start; i < end; i++) {
        if (array[i][0] == value || array[i][0] == toupper(value) || array[i][0]
== tolower(value)) {
            cout << array[i] << " ";
        }
    }
    cout << "END SEARCH\n";
}

void* th_func(void* args) {
    search((string*)((th_param*)args)->array, 0, (int)((th_param*)args)->size,
'z');
    return NULL;
}

void* sleep_th_func(void* args) {
    sleep(1000000);
    search((string*)((th_param*)args)->array, 0, (int)((th_param*)args)->size,
'z');
    return NULL;
}

void* cond_th_func(void* args) {
    pthread_mutex_lock(&mutexLock);
    search((string*)((th_param*)args)->array, 0, (int)((th_param*)args)->size,
'z');
    readyCondition = 1;
    pthread_cond_signal(&mutexCond);
    pthread_mutex_unlock(&mutexLock);
    return NULL;
}

void* spin_th_func(void* args) {
    pthread_spin_lock(spinLock);
    search((string*)((th_param*)args)->array, 0, (int)((th_param*)args)->size,
'z');
    pthread_spin_unlock(spinLock);
    return NULL;
}

int main() {
    fstream file;
    string word, filename;
    filename = "myfile.txt";
    file.open(filename.c_str());
    int size = 0;
    while (file >> word) {
        size++;
    }
    file.close();
    string* array = new string[size];
    file.open(filename.c_str());
    int i = 0;

```

```

while (file >> word) {
    array[i] = word;
    i++;
}

th_param params;
params.size = size;
params.array = array;
threads = new pthread_t[countThread];
int choice = 0;
while (true) {
    cout << "Enter number:\n[1] - ASYNCHRONIZATION\n[2] - SPINLOCK\n[3] - 
CONDITIONAL VARIABLES\n[4] - MEASURE TIME\n[5] - OTHER\n[6] - EXIT\n";
    cin >> choice;
    if (choice == 1) {
        cout << "\nEnter number of threads to be created:" << endl;
        cin >> countThread;
        asyncJob(params);
    }
    else if (choice == 2) {
        cout << "\nEnter number of threads to be created:" << endl;
        cin >> countThread;
        spinLockJob(params);
    }
    else if (choice == 3) {
        cout << "\nEnter number of threads to be created:" << endl;
        cin >> countThread;
        conditionalVariablesJob(params);
    }
    else if (choice == 4) {
        int countThreadArray[] = { 1,2,4 };
        for (int i = 0; i < 3; i++) {
            countThread = countThreadArray[i];
            cout <<
"-----" << endl;
            cout << "\t\t\t\t\tNUMBER OF THREADS - " << countThread <<
endl;
            auto start = high_resolution_clock::now();
            asyncJob(params);
            auto stop = high_resolution_clock::now();
            auto duration = duration_cast<microseconds>(stop - start);
            cout << "Time taken by ASYNCHRONIZATION: " << duration.count() <<
" microseconds" << endl;
            auto start2 = high_resolution_clock::now();
            spinLockJob(params);
            auto stop2 = high_resolution_clock::now();
            auto duration2 = duration_cast<microseconds>(stop2 - start2);
            cout << "Time taken by SPIN LOCK: " << duration2.count() << "
microseconds" << endl;
            auto start3 = high_resolution_clock::now();
            conditionalVariablesJob(params);
            auto stop3 = high_resolution_clock::now();
            auto duration3 = duration_cast<microseconds>(stop3 - start3);
            cout << "Time taken by CONDITIONAL VARIABLES: " <<
duration3.count() << " microseconds" << endl;
            cout <<
"-----" << endl;
        }
    }
    else if (choice == 5) {
        otherThings(params);

```

```

    }
    else if(choice == 6) {
        break;
    }
    else {
        continue;
    }
}

return 0;
}

void setPriorityJob(th_param params) {
    cout << "Enter thread priority: " << endl;
    int priority = 10;
    int dPolicy = 0;
    cin >> priority;
    cout << "Enter the number of sched policy: \n1 - FIFO\n2 - RR\n3 - BATCH\n0 -
OTHER\n";
    std::cin >> dPolicy;
    pthread_t thread;
    pthread_attr_t thAttr;
    pthread_attr_init(&thAttr);

    int inh = PTHREAD_EXPLICIT_SCHED;
    pthread_attr_setinheritsched(&thAttr, inh);

    // int policy = SCHED_FIFO;
    struct sched_param p;
    p.sched_priority = priority;
    pthread_attr_setschedpolicy(&thAttr, dPolicy);

    pthread_attr_getschedpolicy(&thAttr, &dPolicy);
    pthread_attr_setschedparam(&thAttr, &p);

    pthread_attr_getschedpolicy(&thAttr, &dPolicy);
    pthread_attr_getschedparam(&thAttr, &p);

    pthread_create(&thread, &thAttr, &sleep_th_func, &params);

    pthread_join(thread, NULL);

    pthread_attr_destroy(&thAttr);
    cout << "policy: " << dPolicy << endl;
    cout << "priority " << priority << endl;
}

void conditionalVariablesJob(th_param params) {
    for (int i = 0; i < countThread; ++i) {
        pthread_create(&threads[i], NULL, &cond_th_func, &params);
        while (readyCondition == 0) {
            pthread_cond_wait(&mutexCond, &mutexLock);
        }
        pthread_mutex_unlock(&mutexLock);
        readyCondition = 0;
    }
}

void spinLockJob(th_param params) {
    spinLock = (pthread_spinlock_t*)malloc(sizeof(pthread_spinlock_t));

    pthread_spin_init(spinLock, 0);
}

```

```

    for (int i = 0; i < countThread; ++i) {
        pthread_create(&threads[i], NULL, &spin_th_func, &params);
    }

    for (int i = 0; i < countThread; ++i) {
        pthread_join(threads[i], NULL);
    }

    pthread_spin_destroy(spinLock);
}
void* threadFn(void* args) {
    pthread_detach(pthread_self());
    search((string*)((th_param*)args)->array, 0, (int)((th_param*)args)->size,
'z');
    pthread_exit(NULL);
}
void* calls(void* args) {
    search((string*)((th_param*)args)->array, 0, (int)((th_param*)args)->size,
'z');
    pthread_cancel(pthread_self());

    pthread_join(pthread_self(), NULL);
    pthread_exit(NULL);
}

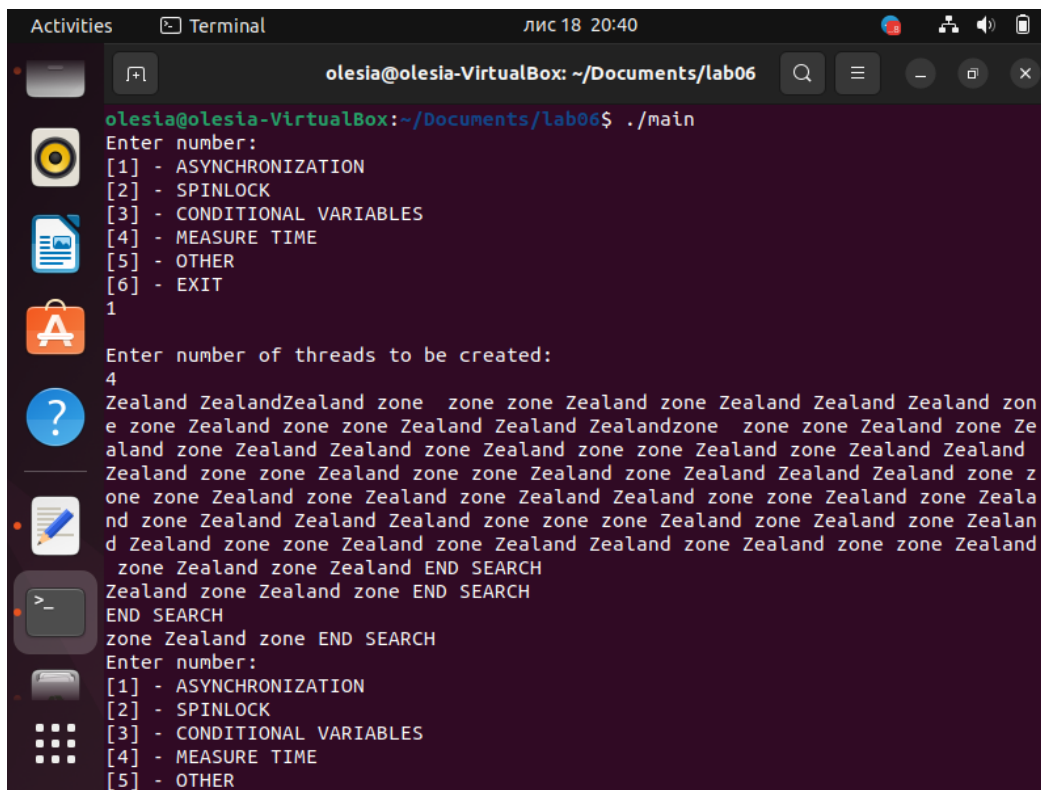
void asyncJob(th_param params) {
    for (int i = 0; i < countThread; ++i) {
        pthread_create(&threads[i], NULL, &th_func, &params);
    }
    for (int i = 0; i < countThread; ++i) {
        pthread_join(threads[i], NULL);
    }
}

void otherThings(th_param params) {
    while (true) {
        int number = 0;
        cout << "Enter number:\n[1] - SET PRIORITY\n[2] - DETACH\n[3] -
CANCEL\n[4] - EXIT\n";
        cin >> number;
        if (number == 1) {
            setPriorityJob(params);
        }
        else if (number == 2) {
            pthread_t threadId;
            pthread_create(&threadId, NULL, threadFn, &params);
            cout << "thread was detached" << endl;
        }
        else if (number == 3) {
            pthread_t threadId;

            pthread_create(&threadId, NULL, calls, &params);
            cout << "thread was cancelled" << endl;
        }
        else if (number == 4) {
            break;
        }
    }
}

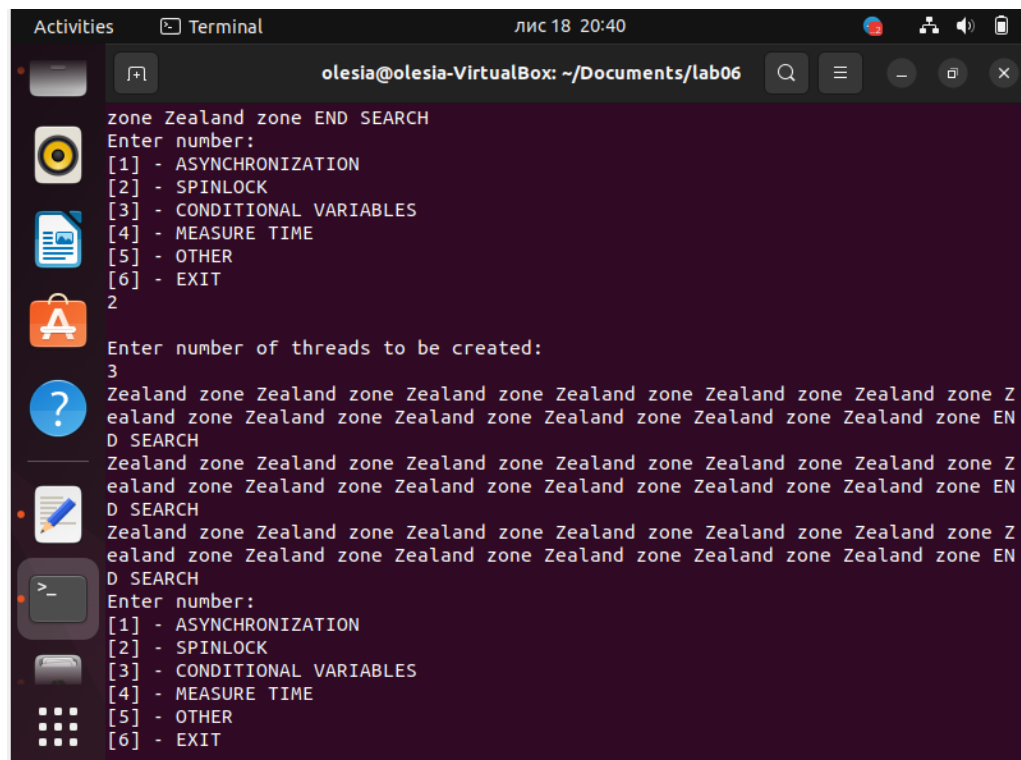
```

Протокол роботи



```
Activities Terminal лис 18 20:40
olesia@olesia-VirtualBox: ~/Documents/lab06
olesia@olesia-VirtualBox:~/Documents/lab06$ ./main
Enter number:
[1] - ASYNCHRONIZATION
[2] - SPINLOCK
[3] - CONDITIONAL VARIABLES
[4] - MEASURE TIME
[5] - OTHER
[6] - EXIT
1
Enter number of threads to be created:
4
Zealand ZealandZealand zone zone zone Zealand zone Zealand Zealand Zealand zon
e zone Zealand zone zone Zealand Zealand Zealandzone zone zone Zealand zone Ze
aland zone Zealand Zealand zone Zealand zone zone Zealand zone Zealand Zealand
Zealand zone zone Zealand zone zone Zealand zone Zealand Zealand Zealand zone z
one zone Zealand zone Zealand zone Zealand Zealand zone zone Zealand zone Zeala
nd zone Zealand Zealand Zealand zone zone zone Zealand zone Zealand zone Zealan
d Zealand zone zone Zealand zone Zealand Zealand zone Zealand zone zone Zealand
zone Zealand zone Zealand END SEARCH
Zealand zone Zealand zone END SEARCH
zone Zealand zone END SEARCH
Enter number:
[1] - ASYNCHRONIZATION
[2] - SPINLOCK
[3] - CONDITIONAL VARIABLES
[4] - MEASURE TIME
[5] - OTHER
```

Рис. 1 Результат виконання програми при асинхронізації.



```
Activities Terminal лис 18 20:40
olesia@olesia-VirtualBox: ~/Documents/lab06
zone Zealand zone END SEARCH
Enter number:
[1] - ASYNCHRONIZATION
[2] - SPINLOCK
[3] - CONDITIONAL VARIABLES
[4] - MEASURE TIME
[5] - OTHER
[6] - EXIT
2
Enter number of threads to be created:
3
Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Z
ealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone EN
D SEARCH
Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Z
ealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone EN
D SEARCH
Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Z
ealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone EN
D SEARCH
Enter number:
[1] - ASYNCHRONIZATION
[2] - SPINLOCK
[3] - CONDITIONAL VARIABLES
[4] - MEASURE TIME
[5] - OTHER
[6] - EXIT
```

Рис. 2 Результат виконання програми при синхронізації за допомогою спіллок.

The screenshot shows a terminal window titled "Terminal" with the system clock at "лис 18 20:41". The terminal is running a program in a virtual machine named "olesia@olesia-VirtualBox" at the directory "~/Documents/lab06". The program is executed with the command `./main`. It prompts the user to "Enter number:" and lists six options: [1] - ASYNCHRONIZATION, [2] - SPINLOCK, [3] - CONDITIONAL VARIABLES, [4] - MEASURE TIME, [5] - OTHER, and [6] - EXIT. The user selects option 3. Next, it prompts "Enter number of threads to be created:" and the user enters 2. The program then outputs a series of "Zealand zone" messages, indicating successful synchronization. The output is as follows:

```
olesia@olesia-VirtualBox: ~/Documents/lab06$ ./main
Enter number:
[1] - ASYNCHRONIZATION
[2] - SPINLOCK
[3] - CONDITIONAL VARIABLES
[4] - MEASURE TIME
[5] - OTHER
[6] - EXIT
3
Enter number of threads to be created:
2
Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Z
ealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone EN
D SEARCH
Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Z
ealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone EN
D SEARCH
Enter number:
[1] - ASYNCHRONIZATION
[2] - SPINLOCK
[3] - CONDITIONAL VARIABLES
[4] - MEASURE TIME
[5] - OTHER
[6] - EXIT
```

Рис. 3 Результат виконання програми при синхронізації за допомогою умовних змінних.

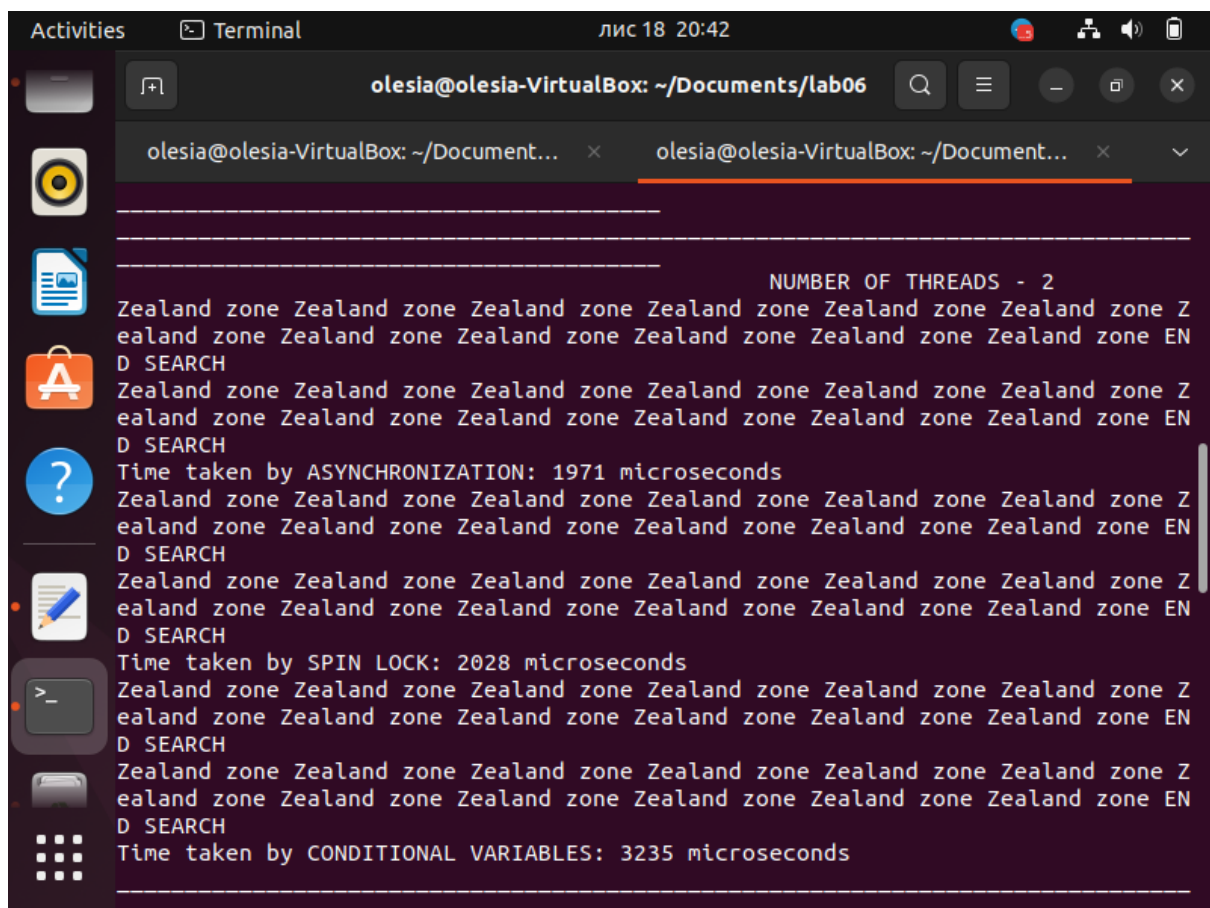
The screenshot shows a terminal window titled 'olesia@olesia-VirtualBox: ~/Documents/lab06'. The terminal output displays two sets of tests for thread synchronization. The first set, labeled 'NUMBER OF THREADS - 1', shows results for ASYNCHRONIZATION (1029 microseconds), SPIN LOCK (999 microseconds), and CONDITIONAL VARIABLES (829 microseconds). The second set, labeled 'NUMBER OF THREADS - 2', shows a result for ASYNCHRONIZATION (1971 microseconds). Each test output is preceded by a line of 'Zealand zone' text and followed by 'END SEARCH'.

```
olesia@olesia-VirtualBox: ~/Documents/lab06
olesia@olesia-VirtualBox: ~/Document... x olesia@olesia-VirtualBox: ~/Document... x v

-----
NUMBER OF THREADS - 1
Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Z
ealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone EN
D SEARCH
Time taken by ASYNCHRONIZATION: 1029 microseconds
Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Z
ealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone EN
D SEARCH
Time taken by SPIN LOCK: 999 microseconds
Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Z
ealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone EN
D SEARCH
Time taken by CONDITIONAL VARIABLES: 829 microseconds
-----

-----
NUMBER OF THREADS - 2
Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Z
ealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone EN
D SEARCH
Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Z
ealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone EN
D SEARCH
Time taken by ASYNCHRONIZATION: 1971 microseconds
Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Z
```

Рис. 4 Вимірювання часу виконання одного потоку при асинхронізації, синхронізації за допомогою спінлоку та синхронізації за допомогою умовних змінних.



The screenshot shows a terminal window titled "olesia@olesia-VirtualBox: ~/Documents/lab06". The terminal output displays benchmark results for two threads, comparing asynchronous synchronization (ASYNCHRONIZATION) and spin lock synchronization (SPIN LOCK). The output is as follows:

```
=====
=====
NUMBER OF THREADS - 2
Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Z
ealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone EN
D SEARCH
Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Z
ealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone EN
D SEARCH
Time taken by ASYNCHRONIZATION: 1971 microseconds
Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Z
ealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone EN
D SEARCH
Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Z
ealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone EN
D SEARCH
Time taken by SPIN LOCK: 2028 microseconds
Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Z
ealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone EN
D SEARCH
Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Z
ealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone EN
D SEARCH
Time taken by CONDITIONAL VARIABLES: 3235 microseconds
=====
```

Рис. 5 Вимірювання часу виконання двох потоків при асинхронізації, синхронізації за допомогою спінлоку та синхронізації за допомогою умовних змінних.

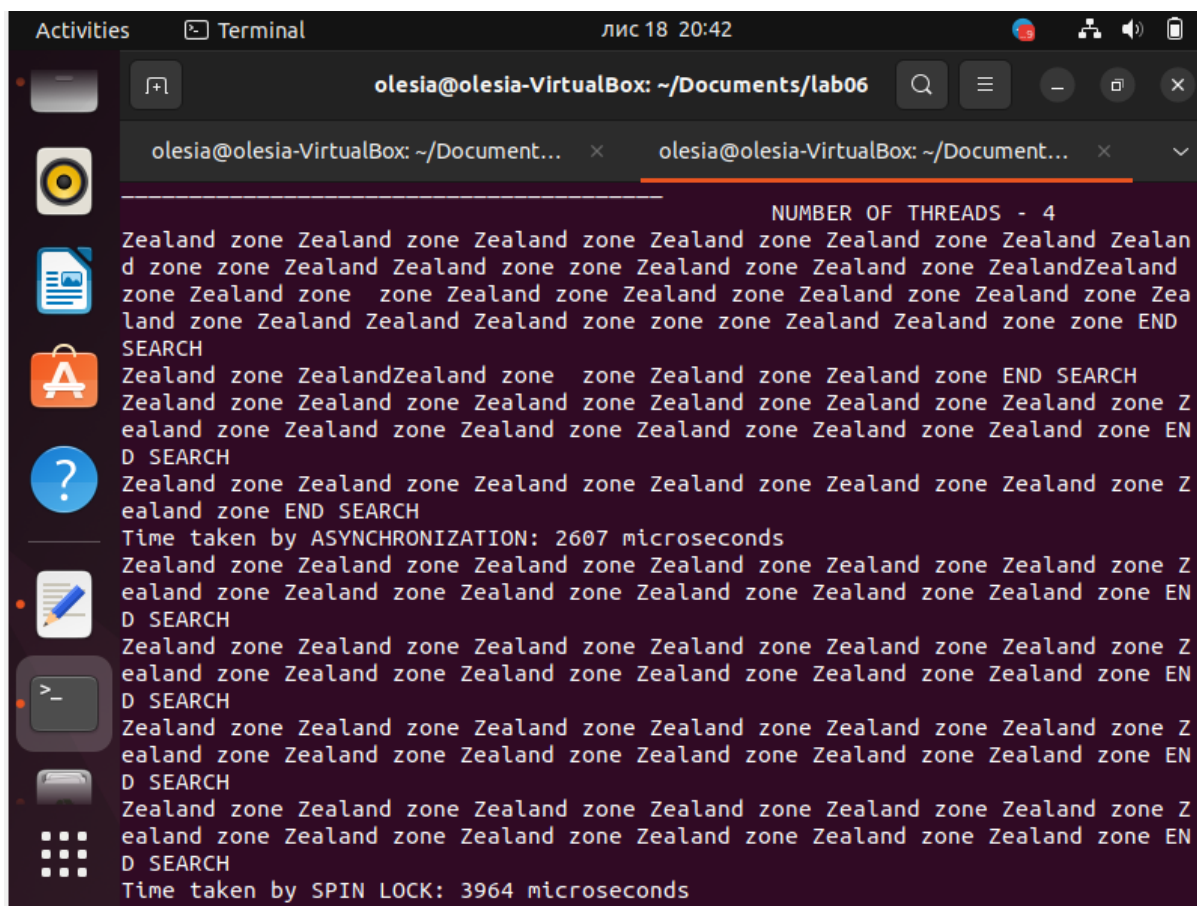


Рис. 6 Вимірювання часу виконання чотирьох потоків при асинхронізації, синхронізації за допомогою спіллоку та синхронізації за допомогою умовних змінних.



Рис. 7 Вимірювання часу виконання чотирьох потоків при асинхронізації, синхронізації за допомогою спіллоку та синхронізації за допомогою умовних змінних.


```
3
thread was cancelled
Enter number:
[1] - SET PRIORITY
[2] - DETACH
[3] - CANCEL
[4] - EXIT
Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone Z
ealand zone Zealand zone Zealand zone Zealand zone Zealand zone Zealand zone EN
D SEARCH
█
```

Рис. 10 Скасування потоку.

amount/type	asynchronous	spinlock	Conditional vars
1	1029	1971	2607
2	999	2028	3964
4	829	3235	4532

Таблиця 1 Порівняння часу виконання при 1,2 та 4 потоках та при асинхронізації, синхронізації за допомогою спінлок та умовних змінних.

Висновок

На цій лабораторній роботі я дізналась про багатопоточність в лінуку та реалізувала програму за допомогою POSIX, де створила декілька потоків , зробила синхронізацію за допомогою умовних змінних та за допомогою спінлоку, виміряла час виконання при різній кількості потоків, , зробила таблицю, де порівняла результати .від'єднала потік, відмінила потік , зробила можливість встановлення пріоритету та policy.