Міністерство освіти та науки України
Львівський національний університет ім. Івана Франка
Факультет електроніки та комп'ютерних технологій
Кафедра радіоелектронних і комп'ютерних систем

Звіт

про виконання лабораторної роботи №6

«Програмна реалізація міжпотокової взаємодії в ОС Windows і Linux»

Виконав:
Студент групи ФЕІ-23
Ангелов М.С.
Перевірив:
ас. Сінькевич О.О.

Львів – 2019

1. Завдання

1) Написати функцію для відправки та для отримання повідомлень потоком.

2) Розробити динамічну структуру даних до якої можна буде доступитись з різних потоків.

3) Розробити програму реалізації блокувань читання-записування з перевагою записування.

2. Виконання

2.1. Повідомлення

1) Виконання з командного рядка

```
nick@nick:~/lab6$ ./p1
Creating thread 0
Creating thread 1
Creating thread 2
Creating thread 3
Thread 1 is waiting
Thread 2 is waiting
Creating thread 4
Thread 3 is waiting
Thread 4 is waiting
Creating thread 5
Created threads, sending message
Sent message, waiting..
Thread 1 finished
Thread 5 is waiting
Thread 5 finished
Thread 2 finished
Thread 6 is waiting
Thread 6 finished
Thread 3 finished
Thread 4 finished
```

2) Код програми

```c
#include <stdio.h>
#include <pthread.h>

pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t cond = PTHREAD_COND_INITIALIZER;
int tcount = 0;
int condition = 0;

void send_msg()
{
    condition = 1;
    pthread_cond_broadcast(&cond);
}

void receive_msg()
{
    pthread_mutex_lock(&mutex);
    while(!condition)
    pthread_cond_wait(&cond, &mutex);
    pthread_mutex_unlock(&mutex);
}

void* someThread(void* p)
{
    int threadID = ++tcount;
    printf("Thread %i is waiting\n", threadID);
    receive_msg();
    printf("Thread %i finished\n", threadID);
}

int main()
{
 int THREAD_CONUT = 6;
 pthread_t threads[THREAD_CONUT];

    for(int i = 0; i < THREAD_CONUT; ++i)
    {
        printf("Creating thread %i\n", i);
        pthread_create(&threads[i], NULL, someThread, &i);
    }

 printf("Created threads, sending message\n");
 send_msg();
 printf("Sent message, waiting..\n");
    for(int i = 0; i < THREAD_CONUT; ++i)
        pthread_join(threads[i], NULL);
 return 0;
}
```

2.2. Динамічна структура даних

1) Виконання з командного рядка

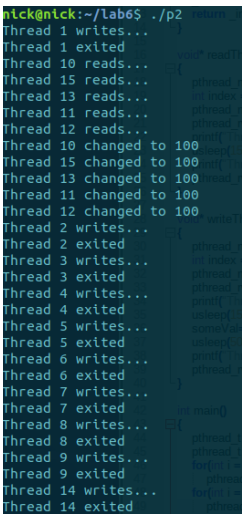
```
Push 1
Push 11
Push 111
Pop 111
Pop 11
Pop 1
Stack is empty
Push 2
Push 22
Push 222
Pop 222
Pop 22
Pop 2
Stack is empty
Push 3
Push 33
Push 333
Pop 333
Pop 33
Pop 3
Stack is empty
```

2) Код програми

```
#include <stdio.h>
#include <pthread.h>
#include <math.h>

pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
int* stack;
int size;
int currentPos = 0;

void createStack(int s) {
    size = s;
    stack = new int[size];
}

int push(int value) {
    pthread_mutex_lock(&mutex);
    if (currentPos >= size) {
        pthread_mutex_unlock(&mutex);
        return -1;
    }
    stack[currentPos++] = value;
    pthread_mutex_unlock(&mutex);
}

int pop() {
    pthread_mutex_lock(&mutex);
    if (currentPos <= 0) {
        pthread_mutex_unlock(&mutex);
        return -1;
    }
    currentPos--;
    pthread_mutex_unlock(&mutex);
    return stack[currentPos];
}

int index = 0;
pthread_mutex_t i = PTHREAD_MUTEX_INITIALIZER;

int getIndex() {
    index++;
    return index;
}

void* threadFunc(void* p) {
    pthread_mutex_lock(&i);
    int value = getIndex();
    pthread_mutex_unlock(&i);
    int currentVal = value;
    for (int i = 1; i < 4; ++i) {
        if (push(currentVal) != -1)
            printf("Pushed %i \n", currentVal);
        else
            printf("Stack is full!\n");
        currentVal = value * pow(10, i) + currentVal;
    }
    for (int i = 0; i < 4; ++i)
        if ((currentVal = pop()) != -1)
            printf("Pop %i \n", currentVal);
        else
            printf("Stack is empty!\n");
}
int main() {
    int THREAD_COUNT = 3;
    pthread_t threads[THREAD_COUNT];
    createStack(THREAD_COUNT * 5);
    for (int i = 1; i <= THREAD_COUNT; ++i)
        pthread_create(&threads[i - 1], NULL, threadFunc, NULL);
    for (int i = 0; i < THREAD_COUNT; ++i)
        pthread_join(threads[i], NULL);
    return 0;
}
```

2.3. IO блокування

1) Виконання з командного рядка



2) Код програми

```c
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
pthread_rwlock_t rwlock = PTHREAD_RWLOCK_INITIALIZER;
int someVal=0;
pthread_mutex_t indexMutex = PTHREAD_MUTEX_INITIALIZER;
int _index=0;

int getIndex()
{
    _index++;
    return _index;
}

void* readThread (void* p)
{
    pthread_mutex_lock(&indexMutex);
    int index = getIndex();
    pthread_mutex_unlock(&indexMutex);
    pthread_rwlock_rdlock(&rwlock);
    printf("Thread %i reads...\n", index);
    usleep(1500);
    printf("Thread %i changed to %i\n", index, someVal);
    pthread_rwlock_unlock(&rwlock);
}

void* writeThread(void* p)
{
    pthread_mutex_lock(&indexMutex);
    int index = getIndex();
    pthread_mutex_unlock(&indexMutex);
    pthread_rwlock_wrlock(&rwlock);
    printf("Thread %i writes...\n", index);
    usleep(1500);
    someVal=index*100;
    usleep(500);
    printf("Thread %i exited\n", index);
    pthread_rwlock_unlock(&rwlock);
}

int main()
{
    pthread_t rthreads[5];
    pthread_t wthreads[10];
    for(int i = 0; i < 10; ++i)
        pthread_create(&wthreads[i], NULL, writeThread, NULL);
    for(int i = 0; i < 5; ++i)
        pthread_create(&rthreads[i], NULL, readThread, NULL);
    for(int i = 0; i < 5; ++i)
        pthread_join(rthreads[i], NULL);
    for(int i = 0; i < 10; ++i)
        pthread_join(wthreads[i], NULL);
    return 0;
}
```

3. Висновок

Під час виконання лабораторноі роботи я дізнався про способи міжпотокової взаємодії у Linux та створив базову програму для демонстрації.