МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра Інформаційної Безпеки

Операційні системи

Комп’ютерний практикум

Робота №7. Основи роботи з потоками у Linux з використанням бібліотеки pthread

Мета:

Оволодіння практичними навичками роботи з потоками POSIX у Linux з використанням бібліотеки pthread.

Перевірив:

_____

Виконав:
студент II курсу
групи ФБ-01
Сахній Н.Р.

Київ 2022

Завдання до виконання:

1. **Створення потоку.** Напишіть програму, що створює потік. Застосуйте атрибути за умовчанням. Батьківський і дочірній потоки мають роздрукувати по десять рядків тексту.

```
nazar@ubuntu:~$ cd OS; mkdir lab_7; cd lab_7
nazar@ubuntu:~/OS/lab_7$ nano my_pthread.cpp
```

nazar@ubuntu: ~/OS/lab_7

GNU nano 4.8 my_pthread.cpp

```cpp
#include <iostream>
#include <stdlib.h>
#include <pthread.h>

using namespace std;


void *func(void *param)
{
    for (int i = 1; i <= 10; i++)
    {
        printf("Child thread string №%d \n", i);
    }
    pthread_exit(NULL);
}


int main()
{
    pthread_t thread;
    pthread_create(&thread, NULL, func, NULL);
    for (int i = 1; i <= 10; i++)
    {
        printf("Parent thread string №%d \n", i);
    }
    return 0;
}

```

```
nazar@ubuntu:~/OS/lab_7$ g++ -pthread -o my_pthread my_pthread.cpp
nazar@ubuntu:~/OS/lab_7$ ls -l
total 24
-rwxrwxr-x 1 nazar nazar 17456 Mar 29 22:51 my_pthread
-rw-rw-r-- 1 nazar nazar   468 Mar 29 22:49 my_pthread.cpp
nazar@ubuntu:~/OS/lab_7$
```

```
nazar@ubuntu:~/OS/lab_7$ ./my_pthread
Parent thread string №1
Parent thread string №2
Parent thread string №3
Parent thread string №4
Parent thread string №5
Parent thread string №6
Parent thread string №7
Parent thread string №8
Parent thread string №9
Parent thread string №10
```

```
Child thread string №1
Child thread string №2
Child thread string №3
Child thread string №4
Child thread string №5
Child thread string №6
Child thread string №7
Child thread string №8
Child thread string №9
Child thread string №10
nazar@ubuntu:~/OS/lab_7$
```

2. **Очікування потоку.** Модифікуйте програму п. 1 так, щоби батьківський потік здійснював роздрукування після завершення дочірнього (функція `pthread_join()`).

```
nazar@ubuntu:~/OS/lab_7$ nano my_pthread.cpp
```

```
18 int main()
19 {
20     pthread_t thread;
21     pthread_create(&thread, NULL, func, NULL);
22     pthread_join(thread, NULL);
23     for (int i = 1; i <= 10; i++)
24     {
25         printf("Parent thread string №%d \n", i);
26     }
27     return 0;
28 }
```

```
nazar@ubuntu:~/OS/lab_7$ g++ -pthread -o my_pthread my_pthread.cpp
nazar@ubuntu:~/OS/lab_7$ ./my_pthread
Child thread string №1
Child thread string №2
Child thread string №3
Child thread string №4
Child thread string №5
Child thread string №6
Child thread string №7
Child thread string №8
Child thread string №9
Child thread string №10
Parent thread string №1
Parent thread string №2
Parent thread string №3
Parent thread string №4
Parent thread string №5
Parent thread string №6
Parent thread string №7
Parent thread string №8
Parent thread string №9
Parent thread string №10
nazar@ubuntu:~/OS/lab_7$
```

3. **Параметри потоку.** Напишіть програму, що створює чотири потоки, що виконують одну й ту саму функцію. Ця функція має роздруковувати послідовність текстових рядків, переданих як параметр. Кожний зі створених потоків має роздруковувати різні послідовності рядків.

```
nazar@ubuntu:~/OS/lab_7$ nano thread_param.cpp
```

nazar@ubuntu: ~/OS/lab_7

GNU nano 4.8 thread_param.cpp

```cpp
#include <iostream>
#include <stdlib.h>
#include <pthread.h>

using namespace std;


typedef struct message
{
    int id;
    const char *string;
}
mess;


void *text_string(void *str)
{
    mess *text = (mess*) str;
    cout << text -> string << endl;
    return 0;
}


int main()
{
    int i;
    pthread_t thread[4];
    mess str[4];
    const char *message[] = {"Thread №1: 'string'", "Thread №2: 'list'", "Thread №3: 'set'", "Thread №4: 'text'"};
    for (i = 0; i < 4; i++)
    {
        str[i].id = i;
        str[i].string = message[i];
    }
    for (i = 0; i < 4; i++)
    {
        pthread_create(&thread[i], NULL, text_string, (void*) &str[i]);
        pthread_join(thread[i], NULL);
    }
    return 0;
}

```

```
nazar@ubuntu:~/OS/lab_7$ ./thread_param
Thread №1: 'string'
Thread №2: 'list'
Thread №3: 'set'
Thread №4: 'text'
nazar@ubuntu:~/OS/lab_7$
```

4. **Примусове завершення потоку.** Дочірній потік має роздруковувати текст на екран. Через дві секунди після створення дочірнього потоку, батьківський потік має перервати його (функція `pthread_cancel()`).

```
nazar@ubuntu:~/OS/lab_7$ nano sleep_thread.cpp
```

nazar@ubuntu: ~/OS/lab_7

GNU nano 4.8 sleep_thread.cpp

```cpp
#include <iostream>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>  // To call the sleep() function

using namespace std;


void *just_text(void *str)
{
    for (int i = 1; ; i++)
    {
        printf("%i line \n", i);
        sleep(1);  // To reduce the number of lines
    }
}


int main()
{
    pthread_t thread;
    pthread_create(&thread, NULL, just_text, NULL);

    sleep(2);
    pthread_cancel(thread);

    pthread_exit(NULL);  // To exit from main thread
    return 0;
}

```

```
nazar@ubuntu:~/OS/lab_7$ g++ -pthread -o sleep_thread sleep_thread.cpp
nazar@ubuntu:~/OS/lab_7$ ./sleep_thread
1 line
2 line
nazar@ubuntu:~/OS/lab_7$
```

5. **Обробка завершення потоку.** Модифікуйте програму п. 4 так, щоби дочірній потік перед завершенням роздруковував повідомлення про це (`pthread_cleanup_push()`).

```
nazar@ubuntu:~/OS/lab_7$ nano sleep_thread.cpp
```

```
void thread_exit(void *)
{
    cout << "Thread was was canceled" << endl;
}


void *just_text(void *)
{
    // ↓ Pushes routine onto the top of the stack of clean-up handlers
    pthread_cleanup_push(thread_exit, NULL);
    for (int i = 1; ; i++)
    {
        printf("%i line \n", i);
        sleep(1);  // To reduce the number of lines
    }
    // ↓ Removes the routine at the top of the stack of clean-up handlers
    pthread_cleanup_pop(1);
}
```

```
nazar@ubuntu:~/OS/lab_7$ g++ -pthread -o sleep_thread sleep_thread.cpp
nazar@ubuntu:~/OS/lab_7$ ./sleep_thread
1 line
2 line
3 line
Thread was terminated
nazar@ubuntu:~/OS/lab_7$
```

Висновки:

У ході виконання комп'ютерного практикуму я усвідомив переваги багатопоточних операційних систем, а саме те, що один процес може поділятися на декілька потоків, які в свою чергу уже виконують певні сегменти програми, але при цьому вони пов'язані єдиними адресним простором, тобто можуть виконувати певні дії над одним і тим же блоком пам'яті. Також перевагами потоків є їхнє квазіпаралельне використання, яке функціонує за допомогою станів потоку, які вказують на готовність, виконання або ж очікування в даний момент.

Також під час роботи з потоками у Linux я ознайомився із бібліотекою pthread та використанням її функцій: `pthread_create()`, `pthread_exit()`, `pthread_join()`, `pthread_cancel()`, та "обробником очищення", який складається із двох ф-ій `pthread_cleanup_push()`, `pthread_cleanup_pop()` Отже, я оволодів теоретичними та практичними навичками роботи з потоками POSIX у Linux з використанням відповідного для цього модулю pthread.