Міністерство освіти і науки України
НТУУ «Київський політехнічний інститут»
Фізико-технічний інститут

ОПЕРАЦІЙНІ СИСТЕМИ

Комп’ютерний практикум №7

Варіант №11

Виконав:

Студент 2 курсу ФТІ
Групи ФІ-92
Поночевний Назар Юрійович

Перевірив:

Ільїн Костянтин Іванович

Київ – 2021

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

Варіант №11

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

Завдання для самостійної підготовки

1. Ознайомитись з документацією і прикладами використання бібіліотеки pthread:

- man pages;
- книги з числа рекомендованих, зокрема [1, розд. 2.2], [5, п. 3.8.4];
- для першого знайомства — POSIX Threads, матеріал з Вікіпедії [8] (російською мовою, бо на поточний момент стаття українською мовою є вкрай неповною);
- дуже непогана стаття [9];
- простий приклад застосування pthread [10];
- велика книга [11];
- авторитетне першоджерело [12].

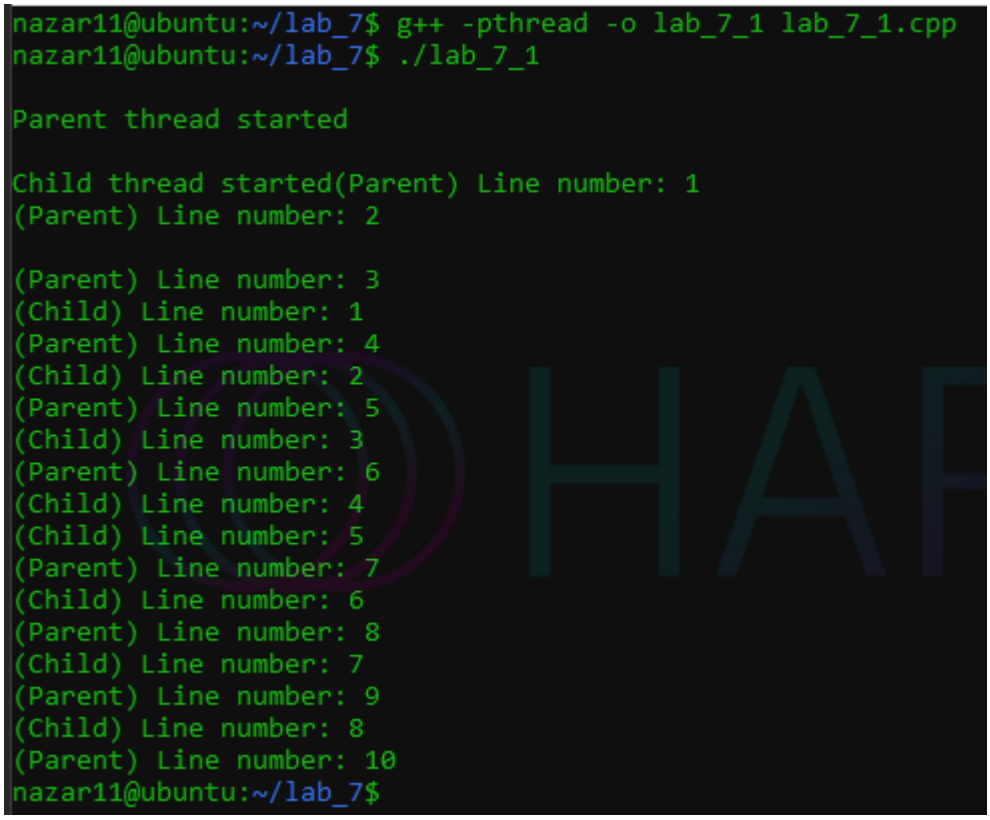
2. Якщо не робили попередню роботу, то перевірити, чи встановлений у вашій системі Linux компілятор C/C++ (g++). Якщо ні, встановіть за допомогою менеджера пакетів.

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.

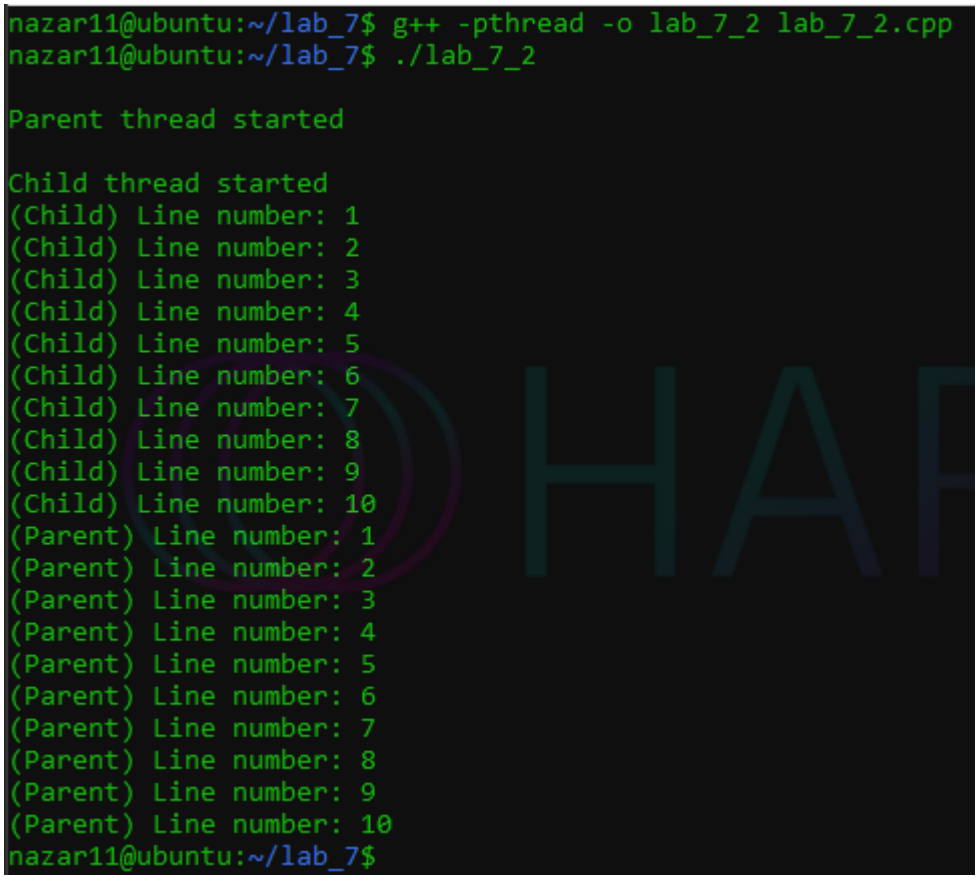
Lab7_1.cpp

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

using namespace std;

pthread_t tid[2];

void* func(void *arg) {
    cout << "\nChild thread started" << endl;
    for (int i = 1; i <= 10; i++) {
        usleep(500);
        cout << "(Child) Line number: " << i << endl;
    }
}

int main(void) {
    cout << "\nParent thread started" << endl;
    tid[0] = pthread_self();

    pthread_create(&(tid[1]), NULL, &func, NULL);

    for (int i = 1; i <= 10; i++) {
        usleep(500);
        cout << "(Parent) Line number: " << i << endl;
    }

    return 0;
}
```



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

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

using namespace std;

pthread_t tid[2];

void* func(void *arg) {
    cout << "\nChild thread started" << endl;
    for (int i = 1; i <= 10; i++) {
        usleep(500);
        cout << "(Child) Line number: " << i << endl;
    }
}

int main(void) {
    cout << "\nParent thread started" << endl;
    tid[0] = pthread_self();

    pthread_create(&(tid[1]), NULL, &func, NULL);
    pthread_join(tid[1], NULL);

    for (int i = 1; i <= 10; i++) {
        usleep(500);
        cout << "(Parent) Line number: " << i << endl;
    }

    return 0;
}
```



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.

```cpp
#include <iostream>
#include <pthread.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <vector>
#include <cstdlib>

using namespace std;

void* func(void *arg) {
    string str = *reinterpret_cast<string*>(arg);
    for (int i = 1; i <= 2; i++) {
        usleep(500);
        cout << str << " and " << i << endl;
    }
    pthread_exit(NULL);
}

int main(void) {
    cout << "Parent thread started" << endl;

    pthread_t tid[4];
    vector<string> text {
        "Amazing first text",
        "Second line here",
        "Why we need third text?",
        "Ohh! That is the last 4 string"
    };

    for (int i = 0; i < 4; i++) {
        pthread_create(&(tid[i]), NULL, &func, &text[i]);
    }
    for (int i = 0; i < 4; i++) {
        pthread_join(tid[i], NULL);
    }

    return 0;
}
```

```
nazar11@ubuntu:~/lab_7$ g++ -pthread -std=c++11 -o lab_7_3 lab_7_3.cpp
nazar11@ubuntu:~/lab_7$ ./lab_7_3
Parent thread started
Ohh! That is the last 4 string and 1
Why we need third text? and 1
Ohh! That is the last 4 string and 2
Why we need third text? and 2
Second line here and 1
Second line here and 2
Amazing first text and 1
Amazing first text and 2
nazar11@ubuntu:~/lab_7$
```

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

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

using namespace std;

void* func(void *arg) {
    cout << "Child thread started" << endl;
    for (int i = 1; i <= 25; i++) {
        usleep(500);
        cout << "(Child) Line number: " << i << endl;
    }
    pthread_exit(NULL);
}

int main(void) {
    cout << "Parent thread started" << endl;

    pthread_t thread;
    pthread_create(&thread, NULL, &func, NULL);

    usleep(5000);
    pthread_cancel(thread);
    cout << "Child thread canceled by parent" << endl;

    return 0;
}
```

```
nazar11@ubuntu:~/lab_7$ g++ -pthread -o lab_7_4 lab_7_4.cpp
nazar11@ubuntu:~/lab_7$ ./lab_7_4
Parent thread started
Child thread started
(Child) Line number: 1
Child thread canceled by parent
nazar11@ubuntu:~/lab_7$
```

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

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

using namespace std;

void cleanup_handler(void *arg) {
   cout << "Someone canceling me!" << endl;
}

void* func(void *arg) {
    pthread_cleanup_push(&cleanup_handler, NULL);
    cout << "Child thread started" << endl;
    for (int i = 1; i <= 25; i++) {
        usleep(500);
        cout << "(Child) Line number: " << i << endl;
    }
    pthread_cleanup_pop(1);
}

int main(void) {
    cout << "Parent thread started" << endl;

    pthread_t thread;
    pthread_create(&thread, NULL, &func, NULL);

    usleep(5000);
    pthread_cancel(thread);
    cout << "Child thread canceled by parent" << endl;

    return 0;
```

```
nazar11@ubuntu:~/lab_7$ g++ -pthread -o lab_7_5 lab_7_5.cpp
nazar11@ubuntu:~/lab_7$ ./lab_7_5
Parent thread started
Child thread started
(Child) Line number: 1
Someone canceling me!
Child thread canceled by parent
nazar11@ubuntu:~/lab_7$
```

Висновок

ОС Linux надає гарну підтримку потоків та бібліотеки для роботи з ними. Працюючи з потоками з Windows можу сказати, що всі необхідні методи, такі як створення, передача параметрів, очікування та безпечне переривання дуже схоже реалізовані. Звичайно, зараз вже є більш потужні бібліотеки, такі як <thread>, наприклад, але основні методи працюють за однаковими принципами.