Міністерство освіти і науки України
Національний технічний університет України
"Київський політехнічний інститут імені Ігоря Сікорського"
Фізико-технічний інститут

ОПЕРАЦІЙНІ СИСТЕМИ

Комп’ютерний практикум

Робота № 7

Виконав
студент гр. ФЕ-01 Дорошенко В. О.
Перевірив
Кіреєнко О. В.

Київ - 2022

Застосуванням системних викликів fork() і exec()

Мета

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

Варіант 6 Залікова книжка ФЕ-0108

Завдання до виконання

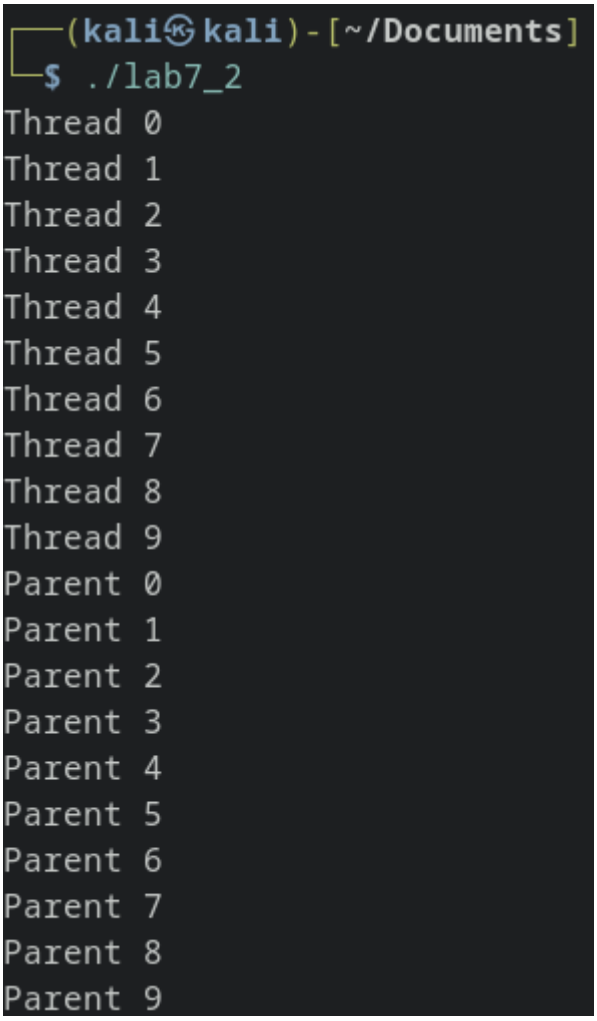
1. Створення потоку. Напишіть програму, що створює потік. Застосуйте атрибути за умовчанням. Батьківський і дочірній потоки мають роздрукувати по десять рядків тексту.

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

void *myThreadFun(void *vargp)
{
        for (int i=0; i<10; ++i)
        {
         printf("Thread %i \n", i);
        }
        pthread_exit(NULL);
}

int main()
{
    pthread_t thread_id;
    pthread_create(&thread_id, NULL, myThreadFun, NULL);
   // pthread_join(thread_id, NULL);
        for (int i=0; i<10; ++i)
        {
          printf("Parent %i \n", i);
        }
        pthread_exit(NULL);
}
```

```
┌──(kali㉿kali)-[~/Documents]
└─$ gcc -pthread -o lab7_2 lab7_2.cpp

┌──(kali㉿kali)-[~/Documents]
└─$ ./lab7_2
Parent 0
Parent 1
Parent 2
Parent 3
Parent 4
Parent 5
Parent 6
Parent 7
Parent 8
Parent 9
Thread 0
Thread 1
Thread 2
Thread 3
Thread 4
Thread 5
Thread 6
Thread 7
Thread 8
Thread 9
```

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

```
int main()
{
    pthread_t thread_id;
    pthread_create(&thread_id, NULL, myThreadFun, NULL);
    pthread_join(thread_id, NULL);
        for (int i=0; i<10; ++i)
        {
```



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.

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

void *myThread(void *vargp)
{
        char *msg = (char *) vargp;
        printf("%s \n", msg);
        pthread_exit(NULL);
}

int main()
{
    pthread_t thread_id_1, thread_id_2, thread_id_3, thread_id_4;
    const char *msg1 = "Thread msg 1";
    const char *msg2 = "Thread msg 2";
    const char *msg3 = "Thread msg 3";
    const char *msg4 = "Thread msg 4";

    pthread_create(&thread_id_1, NULL, myThread, (void*) msg1);
    pthread_create(&thread_id_2, NULL, myThread, (void*) msg2);
    pthread_create(&thread_id_3, NULL, myThread, (void*) msg3);
    pthread_create(&thread_id_4, NULL, myThread, (void*) msg4);
        pthread_exit(NULL);
}
```

```
┌──(kali㉿kali)-[~/Documents]
└─$ g++  -pthread -o lab7_2_2 lab7_2_2.cpp

┌──(kali㉿kali)-[~/Documents]
└─$ ./lab7_2_2
Thread msg 1
Thread msg 4
Thread msg 3
Thread msg 2
```

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()).

```c
void *myThread(void *vargp)
{
        char *msg = (char *) vargp;
        for( ; ;){
        printf("%s \n", msg);
        sleep(1);
        }

        pthread_exit(NULL);
}

int foo()
{
    pthread_t thread_id_1;
    const char *msg1 = "Thread msg 1";
    pthread_create(&thread_id_1, NULL, &myThread, (void*) msg1);
    sleep(2);
    pthread_cancel(thread_id_1);
    pthread_exit(NULL);
}

int main(){
    foo();
    return 0;
}
```

```
┌──(kali㉿kali)-[~/Documents]
└─$ g++ -pthread -o lab7_3 lab_7_3.cpp

┌──(kali㉿kali)-[~/Documents]
└─$ ./lab7_3
Thread msg 1
Thread msg 1
```

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

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

void stopthread(void* arg){
        printf("%s stop. \n", (char *)arg);
}

void *myThread(void* arg){
        pthread_cleanup_push(stopthread, (void*)"Thread 1");
        for( ; ;){
                printf("%s \n", (char *) arg);
                sleep(1);
        }
        pthread_cleanup_pop(0);
        pthread_exit((void*)1);
}

void foo(){
        pthread_t ptid1;
        const char *msg1 = "Thread massege 1";
        pthread_create(&ptid1, NULL, &myThread, (void*) msg1);
        sleep(2);
        pthread_cancel(ptid1);
        pthread_exit(NULL);
}

int main(){
        foo();
        return 0;
}
```

```
┌──(kali㉿kali)-[~/Documents]
└─$ g++ -pthread -o lab_71111 lab_71111.cpp

┌──(kali㉿kali)-[~/Documents]
└─$ ./lab_71111
Thread massege 1
Thread massege 1
Thread massege 1
Thread 1 stop.
```

Висновки

В результаті виконання даної лабораторної роботи я ознайомився з навичками роботи бібліотеки pthread. Ознайомився з базовими функціями для керування потолками pthread_create(), pthread_join(), pthread_exit(), pthread_cancel(), pthread_cleanup_push(), pthread_cleanup_pop().