



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 бібліотеки 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.

```
1  #include <iostream>
2  #include <string.h>
3  #include <pthread.h>
4  #include <stdlib.h>
5  #include <unistd.h>
6
7  using namespace std;
8
9  pthread_t tid[2];
10
11 void* func(void *arg) {
12     cout << "\nChild thread started" << endl;
13     for (int i = 1; i <= 10; i++) {
14         usleep(500);
15         cout << "(Child) Line number: " << i << endl;
16     }
17 }
18
19 int main(void) {
20     cout << "\nParent thread started" << endl;
21     tid[0] = pthread_self();
22
23     pthread_create(&(tid[1]), NULL, &func, NULL);
24
25     for (int i = 1; i <= 10; i++) {
26         usleep(500);
27         cout << "(Parent) Line number: " << i << endl;
28     }
29
30     return 0;
31 }
```

```
nazar11@ubuntu:~/lab_7$ g++ -pthread -o lab_7_1 lab_7_1.cpp
nazar11@ubuntu:~/lab_7$ ./lab_7_1

Parent thread started

Child thread started(Parent) Line number: 1
(Parent) Line number: 2

(Parent) Line number: 3
(Child) Line number: 1
(Parent) Line number: 4
(Child) Line number: 2
(Parent) Line number: 5
(Child) Line number: 3
(Parent) Line number: 6
(Child) Line number: 4
(Child) Line number: 5
(Parent) Line number: 7
(Child) Line number: 6
(Parent) Line number: 8
(Child) Line number: 7
(Parent) Line number: 9
(Child) Line number: 8
(Parent) Line number: 10
nazar11@ubuntu:~/lab_7$
```

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

```
1  #include <iostream>
2  #include <string.h>
3  #include <pthread.h>
4  #include <stdlib.h>
5  #include <unistd.h>
6
7  using namespace std;
8
9  pthread_t tid[2];
10
11 void* func(void *arg) {
12     cout << "\nChild thread started" << endl;
13     for (int i = 1; i <= 10; i++) {
14         usleep(500);
15         cout << "(Child) Line number: " << i << endl;
16     }
17 }
18
19 int main(void) {
20     cout << "\nParent thread started" << endl;
21     tid[0] = pthread_self();
22
23     pthread_create(&(tid[1]), NULL, &func, NULL);
24     pthread_join(tid[1], NULL);
25
26     for (int i = 1; i <= 10; i++) {
27         usleep(500);
28         cout << "(Parent) Line number: " << i << endl;
29     }
30
31     return 0;
32 }
```

```
nazar11@ubuntu:~/lab_7$ g++ -pthread -o lab_7_2 lab_7_2.cpp
nazar11@ubuntu:~/lab_7$ ./lab_7_2

Parent thread started

Child thread started
(Child) Line number: 1
(Child) Line number: 2
(Child) Line number: 3
(Child) Line number: 4
(Child) Line number: 5
(Child) Line number: 6
(Child) Line number: 7
(Child) Line number: 8
(Child) Line number: 9
(Child) Line number: 10
(Parent) Line number: 1
(Parent) Line number: 2
(Parent) Line number: 3
(Parent) Line number: 4
(Parent) Line number: 5
(Parent) Line number: 6
(Parent) Line number: 7
(Parent) Line number: 8
(Parent) Line number: 9
(Parent) Line number: 10
nazar11@ubuntu:~/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.

```

1  #include <iostream>
2  #include <pthread.h>
3  #include <stdlib.h>
4  #include <unistd.h>
5  #include <string.h>
6  #include <vector>
7  #include <cstdlib>
8
9  using namespace std;
10
11 void* func(void *arg) {
12     string str = *reinterpret_cast<string*>(arg);
13     for (int i = 1; i <= 2; i++) {
14         usleep(500);
15         cout << str << " and " << i << endl;
16     }
17     pthread_exit(NULL);
18 }
19
20 int main(void) {
21     cout << "Parent thread started" << endl;
22
23     pthread_t tid[4];
24     vector<string> text {
25         "Amazing first text",
26         "Second line here",
27         "Why we need third text?",
28         "Ohh! That is the last 4 string"
29     };
30
31     for (int i = 0; i < 4; i++) {
32         pthread_create(&(tid[i]), NULL, &func, &text[i]);
33     }
34     for (int i = 0; i < 4; i++) {
35         pthread_join(tid[i], NULL);
36     }
37
38     return 0;
39 }

```

```

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()`).

```

1  #include <iostream>
2  #include <string.h>
3  #include <pthread.h>
4  #include <stdlib.h>
5  #include <unistd.h>
6
7  using namespace std;
8
9  void* func(void *arg) {
10     cout << "Child thread started" << endl;
11     for (int i = 1; i <= 25; i++) {
12         usleep(500);
13         cout << "(Child) Line number: " << i << endl;
14     }
15     pthread_exit(NULL);
16 }
17
18 int main(void) {
19     cout << "Parent thread started" << endl;
20
21     pthread_t thread;
22     pthread_create(&thread, NULL, &func, NULL);
23
24     usleep(5000);
25     pthread_cancel(thread);
26     cout << "Child thread canceled by parent" << endl;
27
28     return 0;
29 }

```

```

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()`).


```

1  #include <iostream>
2  #include <string.h>
3  #include <pthread.h>
4  #include <stdlib.h>
5  #include <unistd.h>
6
7  using namespace std;
8
9  void cleanup_handler(void *arg) {
10     cout << "Someone canceling me!" << endl;
11 }
12
13 void* func(void *arg) {
14     pthread_cleanup_push(&cleanup_handler, NULL);
15     cout << "Child thread started" << endl;
16     for (int i = 1; i <= 25; i++) {
17         usleep(500);
18         cout << "(Child) Line number: " << i << endl;
19     }
20     pthread_cleanup_pop(1);
21 }
22
23 int main(void) {
24     cout << "Parent thread started" << endl;
25
26     pthread_t thread;
27     pthread_create(&thread, NULL, &func, NULL);
28
29     usleep(5000);
30     pthread_cancel(thread);
31     cout << "Child thread canceled by parent" << endl;
32
33     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.