**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

Варіант 8

**Виконав:**

Студент групи ФБ-95

Прохоренко Я.Ю.

**Основи роботи з потоками у Linux з використанням бібліотеки pthread**

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

Operating System: Ubuntu 20.10

Kernel: Linux 5.8.0-43-generic

Architecture: x86-64

Дата виконання: 19.02.2021

#include <iostream>

#include <pthread.h>

#include <string>

#include <unistd.h>

using namespace std;

string parent[] = { "I", "draw", "ten", "lines", "of" ,"text","right","now", "and", "here"};

string child[] = { "I`m", "a", "child", "and", "i", "am", "drawing", "10", "lines", "now" };

void \*functionChild(void\*)

{

for(int i = 0; i < 10; i++)

{

cout << child[i] << "\n";

}

return 0;

}

void \*functionParent(void\*)

{

pthread\_t child\_thread;

pthread\_create(&child\_thread, NULL, \*functionChild, NULL);

sleep(1);

for(int i = 0; i < 10; i++)

{

cout << parent[i]<< "\n";

}

return 0;

}

int main()

{

pthread\_t parent\_thread;

pthread\_create(&parent\_thread, NULL, \*functionParent, NULL);

sleep(8);

return 0;

}



Sleep необхідно було викликати для того щоб потоки встигли виконати свої задачі, до завершення main потоку.

#include <iostream>

#include <pthread.h>

#include <string>

#include <unistd.h>

using namespace std;

string parent[] = { "I", "draw", "ten", "lines", "of" ,"text","right","now", "and", "here"};

string child[] = { "I`m", "a", "child", "and", "i", "am", "drawing", "10", "lines", "now" };

void \*functionChild(void\*)

{

for(int i = 0; i < 10; i++)

{

cout << child[i] << "\n";

}

return 0;

}

void \*functionParent(void\*)

{

pthread\_t child\_thread;

pthread\_create(&child\_thread, NULL, \*functionChild, NULL);

pthread\_join(child\_thread,NULL);

for(int i = 0; i < 10; i++)

{

cout << parent[i]<< "\n";

}

return 0;

}

int main()

{

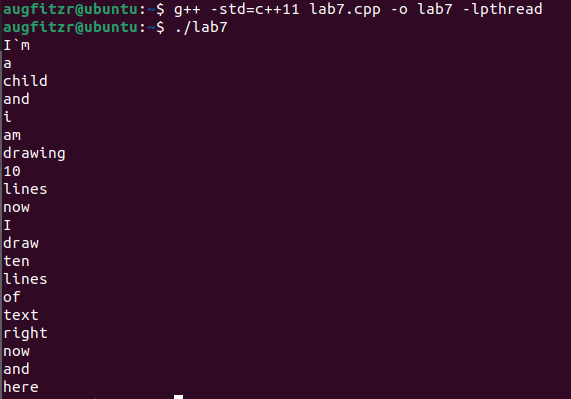
pthread\_t parent\_thread;

pthread\_create(&parent\_thread, NULL, \*functionParent, NULL);

pthread\_join(parent\_thread,NULL);

return 0;

}



Pthread\_join очікує завершення потоку і тому «сон» використовувати не потрібно, «рух» починається одразу як потік завершить свою роботу.

#include <iostream>

#include <pthread.h>

#include <string>

#include <unistd.h>

using namespace std;

#define count\_threads 4

void\* output\_text\_line(void\* tl)

{

string \*getted = (string\*)tl;

for(int i = 0; i < 2; i++)

{

cout << getted[i] << "\n";

}

return NULL;

}

int main()

{

string text\_lines[] = {

"text\_line\_1","text\_line\_2","text\_line\_3","text\_line\_4","text\_line\_5","text\_line\_6","text\_line\_7","text\_line\_8"};

pthread\_t threads[count\_threads];

for(int i = 0; i < count\_threads; i++)

{

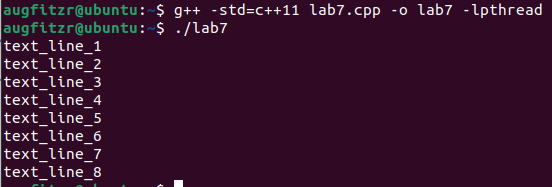
pthread\_create(&threads[i], NULL, output\_text\_line, (void\*)(text\_lines+(i\*2)));

pthread\_join(threads[i], NULL);

}

return 0;

}



#include <iostream>

#include <pthread.h>

#include <string>

#include <unistd.h>

using namespace std;

string text[] = {"asd", "vxc", "thd", "324", "xcv"};

void\* functionChild(void\*)

{

for(int i = 0; i < 5; i++)

{

sleep(1);

cout << text[i] << "\n";

}

return 0;

}

void\* functionParent(void\*)

{

pthread\_t child\_thread;

pthread\_create(&child\_thread, NULL, functionChild, NULL);

sleep(3);

if(pthread\_cancel(child\_thread) == 0)

{

cout << "Child thread cancelled\n";

}

return 0;

}

int main()

{

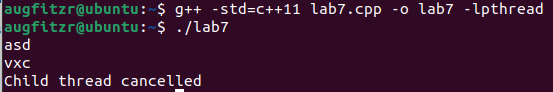
pthread\_t parent\_thread;

pthread\_create(&parent\_thread, NULL, functionParent, NULL);

pthread\_join(parent\_thread, NULL);

return 0;

}



#include <iostream>

#include <pthread.h>

#include <string>

#include <unistd.h>

using namespace std;

string text[] = {"asd", "vxc", "thd", "324", "xcv"};

void func(void\*)

{

cout << "Something\n";

}

void\* functionChild(void\*)

{

pthread\_cleanup\_push(func, NULL);

for(int i = 0; i < 5; i++)

{

sleep(1);

cout << text[i] << "\n";

}

pthread\_cleanup\_pop(1);

return 0;

}

void\* functionParent(void\*)

{

pthread\_t child\_thread;

pthread\_create(&child\_thread, NULL, functionChild, NULL);

sleep(3);

pthread\_cancel(child\_thread);

pthread\_join(child\_thread, NULL);

return 0;

}

int main()

{

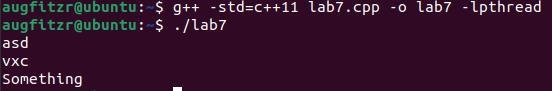
pthread\_t parent\_thread;

pthread\_create(&parent\_thread, NULL, functionParent, NULL);

pthread\_join(parent\_thread, NULL);

return 0;

}



**Висновки:** за допомогою потоків ми можемо виконувати якісь дії паралельно, а не одна за одною, передаючи різні параметри та коректно обробляючи.