

w



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

# Операційні системи

## Лабораторна №8

Виконав:  
Студент групи ФБ-82  
**Козачок Вячеслав**  
Перевірив:  
Кіреєнко О.В.

Київ - 2020

# Зміст

<b>1</b>	<b>Створення Потoku</b>	<b>3</b>
1.1	Code . . . . .	3
1.2	Output . . . . .	3
<b>2</b>	<b>Очікування потоку</b>	<b>4</b>
2.1	Code . . . . .	4
2.2	Output . . . . .	4
<b>3</b>	<b>Параметри потоку</b>	<b>5</b>
3.1	Code . . . . .	5
3.2	Output . . . . .	5
<b>4</b>	<b>Примусове завершення потоку</b>	<b>7</b>
4.1	Code . . . . .	7
4.2	Output . . . . .	7
<b>5</b>	<b>Обробка завершення потоку</b>	<b>8</b>
5.1	Code . . . . .	8
5.2	Output . . . . .	8
<b>6</b>	<b>Висновки</b>	<b>8</b>

1. **Створення потоку.** Напишіть програму, що створює потік. Застосуйте атрибути за умовчанням. Батьківський і дочірній потоки мають роздрукувати по десять рядків тексту.
2. **Очікування потоку.** Модифікуйте програму п. 1 так, щоби батьківський потік здійснював роздрукування після завершення дочірнього (функція `pthread_join()`).
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.
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()`).
5. **Обробка завершення потоку.** Модифікуйте програму п. 4 так, щоби дочірній потік перед завершенням роздруковував повідомлення про це (`pthread_cleanup_push()`).

# Хід роботи

## 1 Створення Потону

### 1.1 Code

```
1 #include <stdio.h>
2 #include <pthread.h>
3 #include <unistd.h>
4 #define TEXT "Test text"
5
6 void * printLines(void * param);
7 int main(int argv, char * argc[])
8 {
9     pthread_attr_t attr;
10    pthread_t p1;
11
12    pthread_attr_init(&attr);
13    int * thread_number = (int*)malloc(sizeof(int));
14    *thread_number = 0;
15    pthread_create(&p1, &attr, printLines, (void*)thread_number);
16
17    printf("%s\n", "Parent process:");
18    for(int i = 0; i < 10; i++)
19        printf("%s\n", TEXT);
20    sleep(1);
21    return 0;
22 }
23
24 void * printLines(void * number)
25 {
26    printf("Child process: %d\n", *(int*)number);
27    for(int i = 0; i < 10; i++)
28        printf("%s\n", TEXT);
29    pthread_exit(0);
30 }
31 }
```

Компіляцію треба здійснювати з флагом `-lpthread`, щоб працювала бібліотека `<pthread.h>`. Також

### 1.2 Output

```
1 ./main
2 Parent process:
3 Test text
4 Test text
5 Test text
6 Test text
7 Test text
8 Test text
9 Test text
10 Test text
11 Test text
12 Test text
13 Child process: 0
14 Test text
15 Test text
16 Test text
17 Test text
18 Test text
19 Test text
20 Test text
21 Test text
22 Test text
23 Test text
```

## 2 Очікування потоку

### 2.1 Code

```
1 #include <stdio.h>
2 #include <pthread.h>
3 #include <unistd.h>
4 #define TEXT "Test text"
5
6 void * printLines(void * param);
7 int main(int argv, char * argc[])
8 {
9     pthread_attr_t attr;
10    pthread_t p1;
11
12    pthread_attr_init(&attr);
13    int * thread_number = (int*)malloc(sizeof(int));
14    *thread_number = 0;
15    pthread_create(&p1, &attr, printLines, (void*)thread_number);
16
17    pthread_join(p1, NULL);
18
19    printf("%s\n", "Parent process:");
20    for(int i = 0; i < 10; i++)
21        printf("%s\n", TEXT);
22
23    return 0;
24 }
25
26 void * printLines(void * number)
27 {
28     printf("Child process: %d\n", *(int*)number);
29     for(int i = 0; i < 10; i++)
30         printf("%s\n", TEXT);
31     pthread_exit(0);
32 }
33 }
```

### 2.2 Output

```
1 ./main
2 Child process: 0
3 Test text
4 Test text
5 Test text
6 Test text
7 Test text
8 Test text
9 Test text
10 Test text
11 Test text
12 Test text
13 Parent process:
14 Test text
15 Test text
16 Test text
17 Test text
18 Test text
19 Test text
20 Test text
21 Test text
22 Test text
23 Test text
```

## 3 Параметры потока

### 3.1 Code

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <pthread.h>
4 #include <unistd.h>
5
6 void * printLines(void * param);
7 int main(int argv, char * argc[])
8 {
9     pthread_t threads[4];
10
11     char ** text = (char**)malloc(3*sizeof(char*));
12     text[0] = "thread";
13     text[1] = "first";
14     text[2] = "Heh Ipsum";
15     pthread_create(&(threads[0]), NULL, printLines, (void*)text);
16
17     text = (char**)malloc(3*sizeof(char*));
18     text[0] = "thread";
19     text[1] = "Second";
20     text[2] = "Avadacedavra Lorem";
21     pthread_create(&(threads[1]), NULL, printLines, (void*)text);
22
23     text = (char**)malloc(3*sizeof(char*));
24     text[0] = "thread";
25     text[1] = "third";
26     text[2] = "Realy S0me ";
27     pthread_create(&(threads[2]), NULL, printLines, (void*)text);
28
29     text = (char**)malloc(3*sizeof(char*));
30     text[0] = "thread";
31     text[1] = "fourth";
32     text[2] = "PrintLines function";
33     pthread_create(&(threads[3]), NULL, printLines, (void*)text);
34
35     printf("%s\n", "Parent process here!");
36
37     pthread_join(threads[0], NULL);
38     return 0;
39 }
40
41 void * printLines(void * text)
42 {
43     printf("Child process: %c", '\n');
44     for(int i = 0; i < 3; i++)
45         printf("%s\n", ((char**)text)[i]);
46     pthread_exit(0);
47 }
48

```

### 3.2 Output

```

1 ./main
2 Child process:
3 Child process:
4 thread
5 Second
6 Avadacedavra Lorem
7 Child process:
8 thread
9 third
10 Child process:
11 thread
12 fourth
13 PrintLines function
14 Parent process here!
15 Realy S0me
16 thread
17 first

```

18 Ней Ipsum



## 4 Примусове завершення потоку

### 4.1 Code

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <pthread.h>
4 #include <unistd.h>
5
6 void * printLines(void * param);
7 int main(int argv, char * argc[])
8 {
9     pthread_t thread;
10
11     char ** text = (char**)malloc(3*sizeof(char*));
12     text[0] = "thread";
13     text[1] = "first";
14     text[2] = "Heh Ipsum";
15     pthread_create(&thread, NULL, printLines, (void*)text);
16     sleep(1);
17
18     pthread_cancel(thread);
19     printf("Child process canceled\n");
20     return 0;
21 }
22
23 void * printLines(void * text)
24 {
25     sleep(2);
26     printf("Child process: %c", '\n');
27     for(int i = 0; i < 3; i++)
28         printf("%s\n", ((char**)text)[i]);
29     pthread_exit(0);
30 }
```

### 4.2 Output

```
1 ./main
2 Child process canceled
```

## 5 Обробка завершення потоку

### 5.1 Code

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <pthread.h>
4 #include <unistd.h>
5
6
7 static int cleanup_pop_arg = 0;
8
9 void * printLines(void * param);
10 int main(int argv, char * argc[])
11 {
12     pthread_t thread;
13
14     char ** text = (char**)malloc(3*sizeof(char*));
15     text[0] = "thread";
16     text[1] = "first";
17     text[2] = "Heh Ipsum";
18     pthread_create(&thread, NULL, printLines, (void*)text);
19
20     pthread_cancel(thread);
21     printf("Child process canceled\n");
22     sleep(1);
23     return 0;
24 }
25
26 static void onCancel(void * arg)
27 {
28     printf("Child process says that it was canceled by someone =(\n");
29 }
30
31 void * printLines(void * text)
32 {
33     pthread_cleanup_push(onCancel, NULL);
34     int done = 0;
35     int i = 0;
36     while(!done)
37     {
38
39         pthread_testcancel();
40
41         printf("%s\n", ((char**)text)[i]);
42         i++;
43         if(i == 3)
44             done = 1;
45         sleep(1);
46     }
47     pthread_cleanup_pop(cleanup_pop_arg);
48 }
```

### 5.2 Output

```
1 ./main
2 thread
3 Child process canceled
4 Child process says that it was canceled by someone =(\n
```

## 6 Висновки

В цій лабораторній роботі я отримав базові навички роботи з потоками, такі як створення потоку, примусове завершення потоку, обробка примусового завершення потоку, очікування завершення потоку, передача потоку додаткових параметрів.