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

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, батіківський 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 Потоку

1.1 Code

```c
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#define TEXT "Test text"

void * printLines(void * param);
int main(int argv, char * argc[])
{
    pthread_attr_t attr;
    pthread_t p1;

    pthread_attr_init(&attr);
    int * thread_number = (int*)malloc(sizeof(int));
    *thread_number = 0;
    pthread_create(&p1, &attr, printLines, (void*)thread_number);

    printf("%s\n", "Parent process:");
    for(int i = 0; i < 10; i++)
        printf("%s\n", TEXT);
    sleep(1);
    return 0;

}

void * printLines(void * number)
{
    printf("Child process: %d\n", *(int*)number);
    for(int i = 0; i < 10; i++)
        printf("%s\n", TEXT);
    pthread_exit(0);
}
```

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

1.2 Output

```
./main
Parent process:
Test text
Test text
Test text
Test text
Test text
Test text
Test text
Test text
Test text
Test text
Test text
Child process: 0
Test text
Test text
Test text
Test text
Test text
Test text
Test text
Test text
Test text
```

2 Очікування потоку

2.1 Code

```c
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#define TEXT "Test text"

void * printLines(void * param);
int main(int argv, char * argc[])
{
    pthread_attr_t attr;
    pthread_t p1;

    pthread_attr_init(&attr);
    int * thread_number = (int*)malloc(sizeof(int));
    *thread_number = 0;
    pthread_create(&p1, &attr, printLines, (void*)thread_number);

    pthread_join(p1, NULL);

    printf("%s\n", "Parent process:");
    for(int i = 0; i < 10; i++)
        printf("%s\n", TEXT);

    return 0;

}

void * printLines(void * number)
{
    printf("Child process: %d\n", *(int*)number);
    for(int i = 0; i < 10; i++)
        printf("%s\n", TEXT);
    pthread_exit(0);
}
```

2.2 Output

```
./main
Child process: 0
Test text
Test text
Test text
Test text
Test text
Test text
Test text
Test text
Test text
Test text
Parent process:
Test text
Test text
Test text
Test text
Test text
Test text
Test text
Test text
Test text
```

3 Параметри потоку

3.1 Code

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

void * printLines(void * param);
int main(int argv, char * argc[])
{
    pthread_t threads[4];

    char ** text = (char**)malloc(3*sizeof(char*));
    text[0] = "thread";
    text[1] = "first";
    text[2] = "Heh Ipsum";
    pthread_create(&(threads[0]), NULL, printLines, (void*)text);

    text = (char**)malloc(3*sizeof(char*));
    text[0] = "thread";
    text[1] = "Second";
    text[2] = "Avadacedavra Lorem";
    pthread_create(&(threads[1]), NULL, printLines, (void*)text);

    text = (char**)malloc(3*sizeof(char*));
    text[0] = "thread";
    text[1] = "third";
    text[2] = "Realy SOme ";
    pthread_create(&(threads[2]), NULL, printLines, (void*)text);

    text = (char**)malloc(3*sizeof(char*));
    text[0] = "thread";
    text[1] = "fourth";
    text[2] = "PrintLines function";
    pthread_create(&(threads[3]), NULL, printLines, (void*)text);

    printf("%s\n", "Parent process here!");

    pthread_join(threads[0], NULL);
    return 0;

}

void * printLines(void * text)
{
    printf("Child process: %c", '\n');
    for(int i = 0; i < 3; i++)
        printf("%s\n", ((char**)text)[i]);
    pthread_exit(0);
}
```

3.2 Output

```
./main
Child process:
Child process:
thread
Second
Avadacedavra Lorem
Child process:
thread
third
Child process:
thread
fourth
PrintLines function
Parent process here!
Realy SOme
thread
first
```

```
18 Heh Ipsum
```

4 Примусове завершення потоку

4.1 Code

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

void * printLines(void * param);
int main(int argv, char * argc[])
{
    pthread_t thread;

    char ** text = (char**)malloc(3*sizeof(char*));
    text[0] = "thread";
    text[1] = "first";
    text[2] = "Heh Ipsum";
    pthread_create(&thread, NULL, printLines, (void*)text);
    sleep(1);

    pthread_cancel(thread);
    printf("Child process canceled\n");
    return 0;
}

void * printLines(void * text)
{
    sleep(2);
    printf("Child process: %c", '\n');
    for(int i = 0; i < 3; i++)
        printf("%s\n", ((char**)text)[i]);
    pthread_exit(0);
}
```

4.2 Output

```
./main
Child process canceled
```

5 Обробка завершення потоку

5.1 Code

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


static int cleanup_pop_arg = 0;

void * printLines(void * param);
int main(int argv, char * argc[])
{
    pthread_t thread;

    char ** text = (char**)malloc(3*sizeof(char*));
    text[0] = "thread";
    text[1] = "first";
    text[2] = "Heh Ipsum";
    pthread_create(&thread, NULL, printLines, (void*)text);

    pthread_cancel(thread);
    printf("Child process canceled\n");
    sleep(1);
    return 0;
}

static void onCancel(void * arg)
{
    printf("Child process says that it was canceled by someone =(;\n");
}

void * printLines(void * text)
{
    pthread_cleanup_push(onCancel, NULL);
    int done = 0;
    int i = 0;
    while(!done)
    {

        pthread_testcancel();

        printf("%s\n", ((char**)text)[i]);
        i++;
        if(i == 3)
            done = 1;
        sleep(1);
    }
    pthread_cleanup_pop(cleanup_pop_arg);
}
```

5.2 Output

```
./main
thread
Child process canceled
Child process says that it was canceled by someone =(;
```

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.