Лабораторная 2

Порядок выполнения (кто первее из потоков выполнит свою работу) потокв не определён и нам необходим инструмент, для управления этим порядком. Для этого используется, например, функция pthread_join

Сигнатура функции :

```
#include <pthread.h>

int pthread_join(pthread_t thread, void **retval);
```

Использование в программе:

- **return_code = pthread_join(thread_id, NULL);**

Функция ожидает завершения потока с указанным идентификатором. Если указанный поток уже завершил выполнение, то функция возвращается немедленно.

После успешного вызова функции pthread_join() вызывающему гарантируется, что указанный поток завершен.

Соединение с потоком, с идентификатором которого уже была выполнена функция pthread_join, приводит к неопределенному поведению.

аргументы функции:

1. первый - идентификатор потока
2. область памяти, куда размещается возвращаемое потоком значение (оно было передано либо через pthread_exit либо для всех кроме main потока через return (неявно вызывается pthread_exit))

   Так как в данной задаче нам неинтересно возвращаемое значение, то мы указываем в этом параметре NULL

возвращаемое значение :

0 - успешное завершение

код ошибки - ошибка при выполнении функции

возможные ошибки :

EDEADLK — deadlock так как два потока попытались соединиться друг с другом; или идентификатор потока является идентификатором вызывающего потока.

EINVAL — указанный поток не является joinable.

EINVAL — другой поток уже ждет, чтобы присоединиться к указанному потоку

ESRCH — не удалось найти поток с идентификатором thread.

Не рекомендуется не делать join с joinable потоками, так как это приводит к созданию “зомби потоков” которые потребляют системные ресурсы, и когда накопилось достаточное количество потоков зомби, больше не будет возможности создавать новые потоки (ограничение в /proc/sys/kernel/threads-max).

Помимо joinable threads существуют ещё detached threads

Когда detached поток завершается, его ресурсы автоматически высвобождаются обратно в систему без необходимости соединения другого потока с завершенным потоком

Сделать поток detahed можно с помощью фукнции pthread_detach

Сигнатура фукнции :

```
#include <pthread.h>

int pthread_detach(pthread_t thread);
```

Функция pthread_detach() помечает поток, идентифицированный thread, как отсоединенный.

Попытка отсоединить уже отсоединенный поток приводит к неопределенному поведению.

аргументы функции :

1. индектификатор потока, который помечается как detached

Возможные ошибки :

EINVAL - указанный поток не является joinable

ESRCH -нет потока с таким идентификатором

Как только поток был отсоединен, он не может быть соединен с pthread_join(3) или снова стать соединяемым.

Новый поток может быть создан в detached состоянии с помощью pthread_attr_setdetachstate(3) для установки аргумента аттрибутов pthread_create(3).

Следует вызвать либо pthread_join(3), либо pthread_detach() для каждого поток, который создает приложение, так как иначе ресурсы любых потоков, для которых одно из этих действий не было выполнено, будут будет освобожден только когда завершится процесс.

Остальные используемые в данной задаче функции были использлованы и описаны в задаче 1