

OS Lab 2

Как завершить поток?

SYNOPSIS

```
#include <pthread.h>

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

Compile and link with -pthread.
```

DESCRIPTION

ERRORS

EDEADLK

A deadlock was detected (e.g., two threads tried to join with each other); or thread specifies the calling thread.

EINVAL thread is not a joinable thread.

EINVAL Another thread is already waiting to join with this thread.

ESRCH No thread with the ID thread could be found.

EDEADLK - поток ожидает завершения самого себя, тем самым порождая мёртвую блокировку.

EINVAL - целевой поток, имеет значение атрибута detachstate равное PTHREAD_CREATE_DETACHED.

ESRCH - указан некорректный идентификатор потока thread.

Когда нить завершается, то связанные с ней ресурсы существуют до того момента, пока какаято другая нить не вызовет `pthread_join(3C)`. Однако к тому моменту, когда `pthread_join` завершается, все ресурсы, занятые нитью (стек, `thread local data`, дескриптор нити) уничтожаются.

Еще одна важная функция, связанная с ожиданием завершения нити – это функция `pthread_detach(3C)`. Эта функция указывает, что все ресурсы, связанные с нитью, необходимо уничтожить сразу после завершения этой нити. При этом уничтожается и код возврата такой нити – при попытке сделать `pthread_join(3C)` на нить, над которой перед этим сделали `pthread_detach(3C)`, возвращается код ошибки `EINVAL`.

В руководстве по `pthread_detach(3C)` в системе Solaris 10 сказано, что главное применение `pthread_detach(3C)` – это ситуация, когда родитель, ожидавший завершения дочерней нити, получает `pthread_cancel(3C)`. В действительности, существуют и другие применения "отсоединенных" нитей.

If two or more threads wait for the same thread to complete, all will suspend processing until the thread has terminated, and then one thread will return successfully and the others will return with an error of **ESRCH**. The `pthread_join()` function will not block processing of the calling thread if the target thread has already terminated.

If a `pthread_join()` call returns successfully with a non-null status argument, the value passed to `pthread_exit(3C)` by the terminating thread will be placed in the location referenced by status.

SYNOPSIS

```
#include <pthread.h>

void pthread_exit(void *retval);

Compile and link with -pthread.
```

Вызывающий поток будет заблокирован, пока указанный поток не вызовет функцию `pthread_exit`, не вернет управление из запускающей процедуры или не будет принудительно завершен другим потоком. Если поток просто выйдет из запускающей процедуры, `retval` будет содержать возвращаемое значение. Если поток был принудительно завершен, по адресу `retval` будет записано значение `PTHREAD_CANCELED`. Вызов функции `pthread_join` автоматически переводит поток в обособленное состояние (вскоре мы обсудим это), которое позволяет вернуть ресурсы потока обратно. Если он уже находится в обособленном состоянии, поток,

вызвавший `pthread_join` , получит код ошибки `EINVAL` . Если нас не интересует возвращаемое значение потока, мы можем передать пустой указатель в аргументе `retval` . В этом случае обращение к функции `pthread_join` позволит дождаться завершения указанного потока, но не вернет код его завершения.

Когда нить завершается, то связанные с ней ресурсы существуют до того момента, пока какаято другая нить не вызовет `pthread_join(3C)` . Однако к тому моменту, когда `pthread_join` завершается, все ресурсы, занятые нитью (стек, `thread local data`, дескриптор нити) уничтожаются.

По умолчанию код завершения потока сохраняется, пока для этого потока не будет вызвана функция `pthread_join`. Основная память потока может быть немедленно освобождена по его завершении, если поток был обособлен. Когда поток обособлен, функция `pthread_join` не может использоваться для получения его кода завершения, потому что в этом случае ее поведение не определено. Обособить поток можно с помощью функции `pthread_detach`.