pthread_cancel

Cancels the execution of a thread. When used, sends a cancellation request to the thread thread. If the cancellation is succesfull it returns zero.

SYNOPSIS

#include<pthread.h>

int pthread_cancel(pthread_t thread);

USAGE

If you use it with pthread_cancel (NULL) the thread will terminate itself.

If you use with with pthread_cancel(pthread_t another_thread) the thread will terminate other thread (if it is cancellable.)

LIBRARY

POSIX thread library: -lpthread

pthread_setcancelstate

Sets whether threa is cancellable or not.

SYNOPSIS

```
#include<pthread.h>

int pthread_setcancelstate(int state, int *oldstate);
```

Returns zero if succesful.

USAGE

```
pthread_setcancelstate(PTHREAD_CANCEL_ENABLE)

pthread_setcancelstate(PTHREAD_CANCEL_DISABLE)
```

LIBRARY

POSIX thread library: -lpthread

pthread_setcanceltype

Sets whether the thread is asynchronously cancellable or deferred cancellable.

SYNOPSIS

#include<pthread.h>

int pthread_setcancelstate(int state, int *oldstate);

Returns zero if succesful.

USAGE

pthread_setcanceltype(PTHREAD_CANCEL_ASYNCHRONOUS)

pthread_setcanceltype(PTHREAD_CANCEL_DEFERRED)

LIBRARY

POSIX thread library: -lpthread

pthread_cleanup_push

Cleanup handlers are specific for each thread. A thread might have multiple cleanup handlers and the cleanup handlers are stored in a thread-specific LIFO stack.

Assume that you have an asynchronous cancellation thread and allocated some memory and opened a file. Asynchronous can cancel a thread immediately causing memory leaks, unclosed file handlers etc. So, after doing this important things we push to the stack so that if the thread is terminated, we can free the memory, close the file handler etc.

SYNOPSIS

```
#include<pthread.h>

void pthread_cleanup_push (void (*routine)(void *), void *arg);
```

USAGE

```
pthread_cleanup_push(memory_cleaner, array);

pthread_cleanup_push(file_cleaner, my_stream);
```

LIBRARY

POSIX thread library: -lpthread

pthread_cleanup_pop

It removes the routine from the top of the stack.

If the execution of pthread_cleanup_push wasn't necessary, the developer has a responsibility to pop the stack. And s/he does it via pthread_cleanup_pop.

Assume that you have pushed into stack two times. You should pop it two times with pthread_cleanup_pop(0).

SYNOPSIS

```
#include<pthread.h>

void pthread_cleanup_pop (int execute);
```

USAGE

```
pthread_cleanup_push(0);
```

LIBRARY

POSIX thread library: -lpthread