DragonFlyBSD - Bug #2763

pthread_mutex_destroy fails with error EINVAL(22) when run from main thread

01/08/2015 02:43 AM - mneumann

Status:	Resolved	**Start date:**	01/08/2015
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

The following program when compiled with -pthread on DragonFly fails with assertion 22, while it works on Linux:

```
#include <assert.h>
#include <pthread.h>

int main() {
pthread_mutex_t lock = PTHREAD_MUTEX_INITIALIZER;
assert(pthread_mutex_destroy(&lock) == 0); // returns EINVAL (22)
}
```

While when it goes through a lock/unlock cycle or pthread_mutex_init() is called before,
the pthread_mutex_destroy() does not fail with EINVAL:

```
int main() {
pthread_mutex_t lock = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_init(&lock, NULL);
assert(pthread_mutex_destroy(&lock) == 0); // OK
}
```

I propose the attached patch to return success if pthread_mutex_destroy() is called
for the PTHREAD_MUTEX_INITIALIZER case.

Associated revisions

Revision 146da5fc - 01/08/2015 03:40 PM - mneumann

Make pthread_*_destroy() more standards compliant

Function pthread_{mutex,cond,rwlock}_destroy() returned EINVAL when the
mutex/cond/rwlock was initialized statically via one of the
PTHREAD_*_INITIALIZER macros and not being used before destruction. We now
return success (0) instead, as it would have been the case when the *_init()
function were used for initialization. This is also the behaviour Linux
exhibits.

Note that we now can no longer detect multiple calls to *_destroy(). Multiple
calls will do no harm, but return success.

While there, fix some potential null pointer derefs in cond and rwlock.

Fixes: #2763

History

#1 - 01/08/2015 04:05 AM - swildner

Correct me if I'm wrong, but it looks as with this patch, pthread_mutex_destroy()ing a mutex twice would no longer result in EINVAL. Do we want that?

#2 - 01/08/2015 10:42 AM - mneumann

Am 08.01.2015 um 13:05 schrieb bugtracker-admin@leaf.dragonflybsd.org:
> Issue #2763 has been updated by swildner.
>
>
> Correct me if I'm wrong, but it looks as with this patch,
pthread_mutex_destroy()ing a mutex twice would no longer result in

EINVAL. Do we want that?

Agreed! It's better to leave as is unless we want to use a sentinel
value of say (void*)1 to detect an already destroyed mutex. Btw, same
problem occurs for condvar and rwlock.

Regards,

`Michael`

#3 - 01/08/2015 03:44 PM - mneumann

- Status changed from New to Resolved

Fixed with commit http://gitweb.dragonflybsd.org/dragonfly.git/commit/146da5fcd4be3d6b3e74514ab38418637b600540.

Files

diff-thread.txt	493 Bytes	01/08/2015	mneumann