1 APR::ThreadMutex - Perl API for APR thread mutexes
1.1 Synopsis

use APR::ThreadMutex ();

my $mutex = APR::ThreadMutex->new($r->pool);
$mutex->lock;
$mutex->unlock;
$mutex->trylock;

1.2 Description

APR::ThreadMutex interfaces APR thread mutexes.

1.3 API

APR::ThreadMutex provides the following functions and/or methods:

1.4 Unsupported API

APR::ThreadMutex also provides auto-generated Perl interface for a few other methods which aren’t
tested at the moment and therefore their API is a subject to change. These methods will be finalized later
as a need arises. If you want to rely on any of the following methods please contact the the mod_perl
development mailing list so we can help each other take the steps necessary to shift the method to an offi-
cially supported API.

1.4.1 DESTROY

META: Autogenerated - needs to be reviewed/completed

Destroy the mutex and free the memory associated with the lock.

$mutex->DESTROY();

- obj: $mutex (APR::ThreadMutex object)
  - the mutex to destroy.

- ref: no return value
- since: subject to change

1.4.2 lock

META: Autogenerated - needs to be reviewed/completed
Acquire the lock for the given mutex. If the mutex is already locked, the current thread will be put to sleep until the lock becomes available.

```
$ret = $mutex->lock();
```

- **obj**: `$mutex` (APR::ThreadMutex object)

  the mutex on which to acquire the lock.

- **ret**: `$ret` (integer)
- **since**: subject to change

### 1.4.3 new

Create a new mutex

```
my $mutex = APR::ThreadMutex->new($p);
```

- **obj**: APR::ThreadMutex (class name)
- **arg1**: `$p` (APR::Pool object)
- **ret**: `$mutex` (APR::ThreadMutex object)
- **since**: subject to change

### 1.4.4 pool_get

META: Autogenerated - needs to be reviewed/completed

META: should probably be renamed to pool(), like all other pool accessors

Get the pool used by this thread_mutex.

```
$ret = $obj->pool_get();
```

- **obj**: `$obj` (APR::ThreadMutex object)
- **ret**: `$ret` (APR::Pool object)

  apr_pool_t the pool

- **since**: subject to change

### 1.4.5 trylock

META: Autogenerated - needs to be reviewed/completed

Attempt to acquire the lock for the given mutex. If the mutex has already been acquired, the call returns immediately with APR_EBUSY. Note: it is important that the APR_STATUS_IS_EBUSY(s) macro be used to determine if the return value was APR_EBUSY, for portability reasons.
$ret = $mutex->trylock();

- **obj:** $mutex (APR::ThreadMutex object)
  the mutex on which to attempt the lock acquiring.

- **ret:** $ret (integer)
  - **since:** subject to change

### 1.4.6 `unlock`

META: Autogenerated - needs to be reviewed/completed

Release the lock for the given mutex.

```perl
$ret = $mutex->unlock();
```

- **obj:** $mutex (APR::ThreadMutex object)
  the mutex from which to release the lock.

- **ret:** $ret (integer)
  - **since:** subject to change

### 1.5 See Also

mod_perl 2.0 documentation.

### 1.6 Copyright

mod_perl 2.0 and its core modules are copyrighted under The Apache Software License, Version 2.0.

### 1.7 Authors

The mod_perl development team and numerous contributors.
# Table of Contents:

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>APR::ThreadMutex - Perl API for APR thread mutexes</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Synopsis</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Description</td>
<td>2</td>
</tr>
<tr>
<td>1.3 API</td>
<td>2</td>
</tr>
<tr>
<td>1.4 Unsupported API</td>
<td>2</td>
</tr>
<tr>
<td>1.4.1 DESTROY</td>
<td>2</td>
</tr>
<tr>
<td>1.4.2 lock</td>
<td>2</td>
</tr>
<tr>
<td>1.4.3 new</td>
<td>3</td>
</tr>
<tr>
<td>1.4.4 pool_get</td>
<td>3</td>
</tr>
<tr>
<td>1.4.5 trylock</td>
<td>3</td>
</tr>
<tr>
<td>1.4.6 unlock</td>
<td>4</td>
</tr>
<tr>
<td>1.5 See Also</td>
<td>4</td>
</tr>
<tr>
<td>1.6 Copyright</td>
<td>4</td>
</tr>
<tr>
<td>1.7 Authors</td>
<td>4</td>
</tr>
</tbody>
</table>