Pithikos Remove obsolete documentation after commit @860303e 55fb891 on 11 Dec 2015

2 contributors

53 lines (32 sloc) 2.62 KB Raw Blame History

build passing

C Thread Pool

This is a minimal but advanced threadpool implementation.

- ANCI C and POSIX compliant
- Pause/resume/wait as you like
- Simple easy-to-digest API
- Well tested

The threadpool is under MIT license. Notice that this project took a considerable amount of work and sacrifice of my free time and the reason I give it for free (even for commercial use) is so when you become rich and wealthy you don't forget about us open-source creatures of the night. Cheers!

Run an example

The library is not precompiled so you have to compile it with your project. The thread pool uses POSIX threads so if you compile with gcc on Linux you have to use the flag `-pthread` like this:

```
gcc example.c thpool.c -D THPOOL_DEBUG -pthread -o example
```

Then run the executable like this:

```
./example
```

Basic usage

1. Include the header in your source file: `#include "thpool.h"`
2. Create a thread pool with number of threads you want: `threadpool thpool = thpool_init(4);`
3. Add work to the pool: `thpool_add_work(thpool, (void*)function_p, (void*)arg_p);`

The workers(threads) will start their work automatically as fast as there is new work in the pool. If you want to wait for all added work to be finished before continuing you can use `thpool_wait(thpool);` . If you want to destroy the pool you can use `thpool_destroy(thpool);` .

API

For a deeper look into the documentation check in the thpool.h file. Below is a fast practical overview.

Function example	Description
thpool_init(4)	Will return a new threadpool with `4` threads.
thpool_add_work(thpool, (void*)function_p, (void*)arg_p)	Will add new work to the pool. Work is simply a function. You can pass a single argument to the function if you wish. If not, `NULL` should be passed.
thpool_wait(thpool)	Will wait for all jobs (both in queue and currently running) to finish.
thpool_destroy(thpool)	This will destroy the threadpool. If jobs are currently being executed, then it will wait for them to finish.
thpool_pause(thpool)	All threads in the threadpool will pause no matter if they are idle or executing work.
thpool_resume(thpool)	If the threadpool is paused, then all threads will resume from where they were.