The Threadpool

The pool is implemented by a queue. When there is a new job to perform, it should be place in the queue. When there will be available thread (can be immediately), it will handle this job.

You should implement the functions in threadpool.h.

The main should first init the thread pool by calling the function create_threadpool(int). This function gets the size of the pool.

create_threadpool should:

1. Check the legacy of the parameter.
2. Create threadpool structure and initialize it:
    a. num_thread = given parameter
    b. qsize=0
    c. threads = pointer to <num_thread> threads
    d. qhead = qtail = NULL
    e. Init lock and condition variables.
    f. shutdown = dont_accept = 0
    g. Create the threads with do_work as execution function and the pool as an argument.

do_work should run in an endless loop and:

1. If destruction process has begun (shutdown=1), exit thread
2. If the queue is empty, wait (no job to make)
3. Check again destruction flag after wake-up.
4. Take the first element from the queue (*work_t)
5. If the queue becomes empty and destruction process waits to begin (dont_accept=1), signal destruction process.
6. Call the thread routine.

dispatch gets the pool, pointer to the thread execution routine and argument to thread execution routine. dispatch should:

1. Create work_t structure and init it with the routine and argument.
2. If destroy function has begun (dont_accept=1), don't accept new item to the queue
3. Add item to the queue

destroy_threadpool

1. Set don’t_accept flag to 1
2. Wait for queue to become empty
3. Set shutdown flag to 1
4. Signal threads that wait on empty queue, so they can wake up, see shutdown flag and exit.

5. Join all threads
6. Free whatever you have to free.

Program flow:

1. Main creates pool of threads, threads wait for jobs.
2. Main accept a new job
3. Main dispatch a job - call dispatch with the main operation function and index as a parameter.
1. dispatch will add work_t item to the queue.
2. When there will be an available thread, it will takes a job from the queue and run the operation function.

Error handling:

1. In any case of wrong command usage, print

"Usage: threadpool <pool-size> <max-number-of-jobs>\n"

2. In any case of a failure, use perror(“error: <sys_call>\n”) and exit the program.

In order to check your pool, create a main file with a function that you can send as a “work” for the threads. This operation function in the main should (print the job index, sleep for 1 second) * 3