CEng 536 Advanced Unix
Fall 2011
HW3
Due: 19/12/2011

In this homework you will write a thread pool based server library named tpool. Library will have capabilities of mapping tasks to threads in the pool and communication through channels.
The following definitions are given in tpool.h.

```
struct Task {
        int (*action)(void *);
        void *params;
};
#define CHANBROAD       0

#define NOTIMEOUT       0
#define RCVBLOCK        1

void starttpool(int numthreads);
void waitcomplete();
int newtask(struct Task *task);

int sendmess(int channel, int length, void *messbody, int timeout);
int rcvmess(int channel, int maxlength, void *messbody, int block);
```

The definitions and their meanings are given as:

`startpool(numthreads)`
: Creates and inits numthreads threads for processing the tasks. Threads take tasks to execute from a task queue. As soon as a task is inserted in the queue, it should be mapped to one of the free threads in the pool and the task->action(task->params) call should be made by the thread. When all threads are busy, task waits in the queue until one of the threads is available.

  A thread body is as simple as a loop of getting a mapped task and calling the function. Threads never terminate but iterate on the same loop unless waitcomplete() is called.

`waitcomplete()`
: It is called by the main thread to check if it is ok to exit. It signals the end of the program, threads terminate if task queue is empty. waitcomplete() blocks until the last thread in the pool terminates. Then it returns meaning the program can exit gracefully.

`newtask(task)`
: Inserts a new task in the task queue. There is no limit in the number of tasks in the task queue. As long as task queue not empty, task is mapped to a thread and started executing.

`sendmess(channel, length, messbody, timeout)`
: All threads in the system can communicate through message passing. channel is the id of the channel to message on. The message is received by the first receiver in the same channel. Messages are queued and processed in FIFO manner. One exception is the timeout value. The message expires after timeout seconds from the time of insertion. It is removed from the queue or ignored afterwards.

  length is the length of the message body given in messbody in bytes.

`rcvmess(channel, maxlength, messbody, block)`
: Receive a messages on channel. First maxlength bytes of the message in the queue is copied on messbody. If actual message body is shorter only that part is copied. If it is longer, the remainder is ignored. Returns number of bytes copied on success, otherwise -1 returned.

block parameter is a flag denoting if receive should wait until a message in the same channel exists. If 0, function will not block, immediately return 0 if no message exists. When it is RCVBLOCK, it will wait until there is a message in the given channel and read it.

Receiving a message consumes (delete message from queue) the message on all channels but CHANBROAD. This channel is the broadcast channel where all threads in the task pool might receive. A broadcast message is removed from the queue only if it is received by all numthreads threads or by expiration. It is important if the receiving threads are the distinct threads. A double rcvmess() call on the broadcast channel by the same thread never reads the same message.

You are asked to implement this library `libtpool`. You are not allowed to use any IPC mechanism like semaphores, message queues or pipes. It should be a pure pthread based implementation. Initial set of tasks can be added by the main thread and each task can call newtask() to introduce new tasks.

Please provide your implementation on 3 files in a tar.gz archive (no RAR, no ZIP please):

- A header file tpool.h that can be used as include file for library users as well.
- Library implementation on a C or C++ file. It should not have main() implementation. All functions and global variables that are for library internal use should be defined as `static`.
- A Makefile compiling your homework as libtpool.so.

Please ask all questions to:
`news://news.ceng.metu.edu.tr:2050/metu.ceng.course.536/`
`https://cow.ceng.metu.edu.tr/News/thread.php?group=metu.ceng.course.536`