User Level Thread Library

mythread_t imitates pthread lib. Following functions are implemented.

1. create
2. yield
3. join
4. exit

Methodology

A circular Queue is implemented using pointers *head, *tail. The circular linked list stores information about a node using Struct node. Node contains two pointers, one pointing to the next node, named *next, two pointing to the previous node, named *last and a store to the *mypthread_t, named n. It also stores 'th_id' to uniquely identify the threads. Queue is implemented to store all the threads in the order of creation. Queue implements following functions:

1. enqueue() - Adds Node into circular queue
2. searchNextActiveThread() - Searches next active thread
3. returnHead() - Returns Head node
4. returnTail - Returns Tail node
5. searchThread - Searches for thread in the queue

create()

In this function we create threads by calling functions *getcontext*() to initialize and *makecontext*(). Before we call *makecontext*() we dynamically allocate context and update the uc_stack.ss_sp and uc_stack.ss_size.Since, main thread is considered to be at first thread, we add it to the head of the queue and give it thread id (th_id) of 1. Rest of the threads are given th_id incrementally when called to create thread.Each thread created has ACTIVE state by default.

yield()

function call to yield finds a next ACTIVE state thread in the queue to swap the context with the currently running thread. Using the *swapcontext*() method of *ucontext.h.*

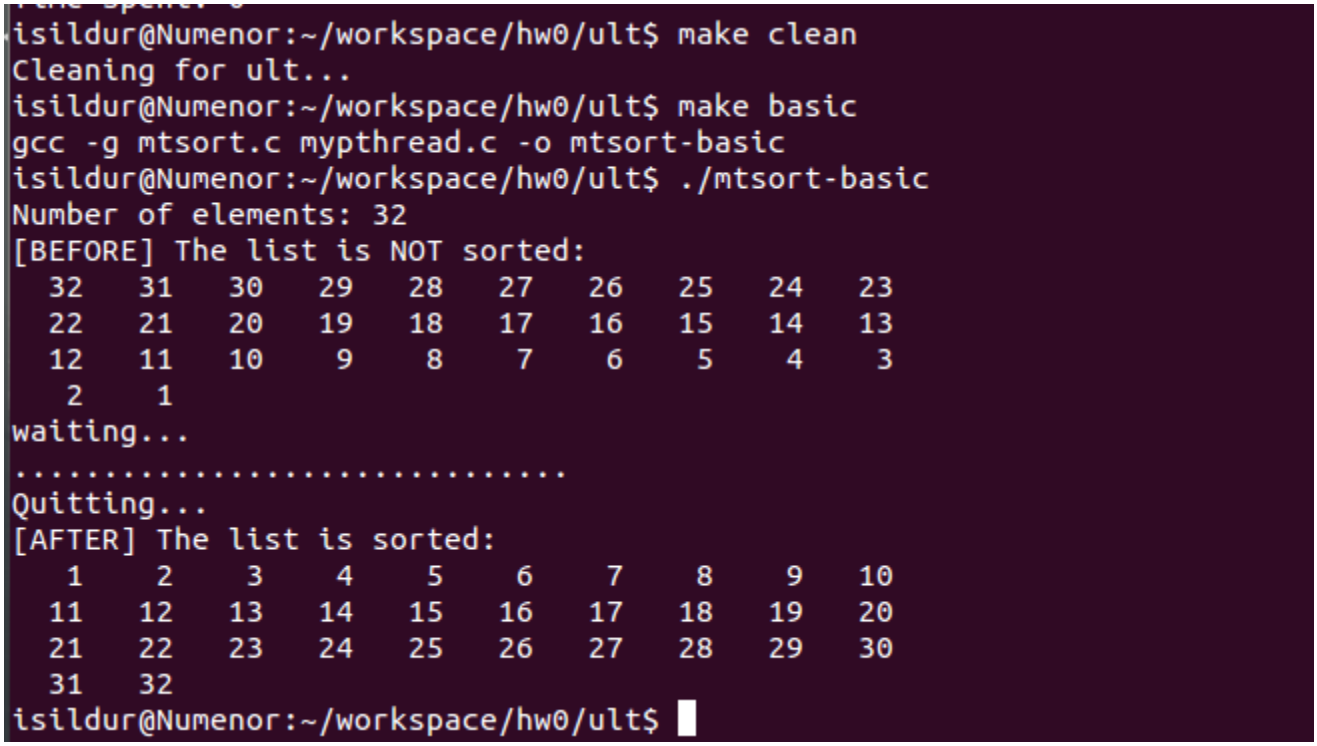
join()

Join like Yield searches for the given thread in the parameter and switches the context with the current running thread using *swapcontext*(). However, the running thread's state is changed to BLOCKED. Also, the thread calling the join function is now added to the joinfrom_th attribute, to ensure that this thread runs as soon as the the joined thread finishes its process.

exit()

The state of current running thread is changed to DEAD. The memory allocated to the context of current thread is freed here. However, to check for which thread should be run next, the state of current thread's attribute of joinfrom_th is changed to ACTIVE and the context of the next ACTIVE thread is set using the *setcontext*().

Result



Note:- Result is obtained by removing the sleep() in the mtsort.c file. Though, result can be obtained even with the sleep. The method only delays the result by 8 minutes.