CS167 Project 3: UThread

Zhang, Shu (szhang)

March 4, 2013

1 Function Implementations

uthread_init() Besides the three init subfunctions, it also clear the memory of uthread id bitmap and the global uthread array.

uthread_create() First get the thread ID , then get the corresponding thread structure , then initialize all its members. At last put the structure into the runnable queue.

uthread_exit() First set the state to Zombie and set the exit value. Then, if the thread has not been detached, see if there is already a waiter thread waiting , if yes, then wake it up, otherwise, switch the thread and never put it into the runnable queue. If the thread has been detached, then wake up the reaper by calling make_reapable().

uthread_join() First check if the join operation is allowed, a series of error will be set if there is something wrong, the error numbers majorly refer to the pthread_join(). If the join operation is allowed, then set the caller as the waiter of the target thread, if the thread is already a zombie thread, then reallocate it direcly. Else block() until the thread calls exit() to wake the caller up.

uthread_detach() If the target thread is a zombie thread, then deallocate it directly, else set the thread as detached. If there is a thread joining the thread, just return and does not set the detached flag.

uthread_destroy() Free the stack of the thread, and memset all the contents of the uthread struct to zero.

uthread_alloc() Find the first unallocated uthread ID in the bitmap.

uthread_add_to_runnable_queue() Since the runq_table is static, I write a function as an interface for functions in other files to use to add a thread into the runnable queue.

uthread_yield() Set the current thread to be runable and call switch() to swap to another thread.

utrhead_block() Set the current thread to be waiting and call switch() to swap to another thread, inside switch(), if the function sees the state of current thread is UT_WAIT, it will not put the thread into the runnable queue.

uthread_wake() Put the current thread to the runnable queue and set the current thread to the target and swap the context to the context of target thread.

uthread_setprio() Remove the target thread from the previous runnable queue of previous priority (If it is in the RUNNABLE state) and enqueue it to the target queue.

uthread_switch() First put the current thread into runnable queue if it is previously set to UT_RUNNABLE state or just don't put it into the runnable queue if it is not in the RUNNABLE state. Then traverse the runnable queue table from the largest priority to get the first thread to run. If there is no available runnable thread, then call uthread_idle() once. Then start the loop again until we find another thread to switch.

uthread_sched_init() Initialize all runnable queues.

uthread_mtx_init() Memset the content of the mutex to zero, and initialize the waiter queue of the mutex.

uthread_mtx_lock() If the mutex has no owner, then set the owner as the caller. If it is locked, then set the thread to the waiter queue and then call block to switch.

uthread_mtx_trylock() If the mutex has an owner, then return directly without blocking.

uthread_mtx_unlock() If the owner of the mutex is not the caller, it will cause error. Then, if there is some thread waiting ,then dequeue the waiter thread and set it as the ownner, and remove it from the blocking state to RUNNABLE.

uthread_cond_init() Initialize the cond's waiter queue.

uthread_cond_wait() Unlock the mutex first, and block the caller by inserting it into the cond's waiting queue. If block() is returned, then lock the mutex again.

uthread_cond_broadcast() Dequeue all the waiters and set them as Runnable and put them into runnable queue.

uthread_cond_signal() Dequeue a waiter and set it as Runnable and put it into runnable queue.