Aaron Attarzadeh
Cullen Coyle
CSCI 350 Assignment 2
9/30/19

Task 1.1

1. In exit, checked to see if the thread isn't running and that it is used
2. If that is the case, then set it to ZOMBIE state

Task 1.2

1. Defined kthread functions in kthread.h with appropriate parameters
2. Created system calls for each of the kthread functions and linked them appropriately (kthread_create, kthread_id, kthread_exit, kthread_join) so they would be able to get called
3. Created kthread.c file with all these functions
4. Developed kthread_create() function
    a. Used allocthread that was already provided to allocate a thread under the proc struct
    b. With allocthread returning a new thread, we change the state to runnable
    c. Set the required stack entries to the proper configuration (eip, esp, ebp)
5. Developed kthread_id() function
    a. This first does an empty check for if there are zero threads or processes
        i. Returns -1 if an error
    b. Then returns the tid of the thread
6. Developed kthread_exit() function
    a. Acquires a lock so it can properly terminate a thread without issues
    b. This checks the state of the thread with function kthreadstate()
        i. Returns if it's a valid thread
    c. Once that happens, it turns the thread into a zombie state and releases the lock
7. Developed kthread_join() function
    a. Located the thread within the stack
    b. Ensure that the thread is a ZOMBIE
    c. Holds the current running thread
    d. Conjoins the new thread
    e. Releases the running thread
    f. Returns

Task 2

1. Create all the functions in kthread.h (kthread_mutex_alloc, kthread_mutex_dealloc, kthread_mutex_lock, kthread_mutex_unlock)
2. Created system calls for each of the mutex functions
3. Developed kthread_mutex_alloc
    a. Allocates a new kthread_mutex in memory
    b. Properly initializes stack within memory
4. Developed kthread_mutex_dealloc

    a. Locates the kthread_mutex by id
    b. Makes sure it is locked
    c. Sets the state to UNUSED to be reallocated at another time
    d. Sets the id to -1

5. Developed kthread_mutex_lock
    a. Locks the mutex table
    b. Locates the thread in the queue
    c. Increments the last thread pointer
    d. Declares its new invalid state
    e. Releases the mutex table lock
6. Developed kthread_mutex_unlock
    a. Ensures that the mutex is in use
    b. Ensures that the thread is not in the queue
    c. Decrements the last thread pointer