CSP 701: Assignment 2(MyThread)

Implementing User-Level Thread Library and its Synchronization

Implement a user thread library **"MyThread"** with a Round-Robin Scheduler. It should support following functionalities (remember, name of functions should exactly match the one mentioned here; or no evaluation for your code!).

1. It should have a system timer/alarm driven scheduler that preempts execution of a thread and elects a new one in round robin fashion.

2. Every thread should do some log keeping; i.e. Store its CPU (execution) time, number of burst etc.; so you should maintain data structure for each thread.

3. Following functionalities:

(a) int create(void (*f)(void)): creates a thread, whose execution starts from function **f** which has no parameter and no return type; however **create** returns thread ID of created thread.
(b) int getID(): called from inside a thread. Should return thread ID of current thread.
(c) void dispatch(int sig): A scheduler for your thread library.
(d) void start(): Called by your main function. One called,starts running of thread and never return.(so at least one of your thread should have infinite loop.)
(e) void run(int threadID): Submits particular thread to scheduler for execution. Could be called from any other thread.
(f) void suspend(int threadID): Suspends a thread till resume is not called for that thread.
(g) void resume(int threadID): resumes the particular thread.
(h) void yield(): calling thread passes control to another next thread.
(i) void delete(int threadID): deletes a particular thread.
(j) void sleep(int sec): Sleeps the calling thread for sec seconds.
(k)struct statistics* getStatus(int threadID) : returns the status of threaded by returning pointer to its statistics or NULL.
(l) int createWithArgs(void *(*f)(void *), void *arg) f is a ptr to a function which takes (void *) and returns (void *). Unlike the threads so far, this thread takes arguments, and instead of uselessly looping forever, returns a value in a (void *). This function returns the id of the thread created.

(m) void clean(): stop scheduler, frees all the space allocated, print statistics per thread.

BONUS modules:

(a) Implement **void JOIN(int threadID):** calling thread blocks itself for a thread to finishes.

(b) Implement **void *GetThreadResult(int threadID)** waits till a thread created with the above function returns, and returns the return value of that thread. This function, obviously, waits until that thread is done with.

Statistics per thread:

1. thread id
2. state (running, ready, sleeping, suspended).
3. number of bursts (none if the thread never ran).
4. total execution time in msec(N/A if thread never ran).
5. total requested sleeping time in msec (N/A if thread never slept).
6. average execution time quantum in msec (N/A if thread never ran).
7. average waiting time (status = READY) (N/A if thread never ran).

NOTE: This assignment need to be done on Linux OS only.