CS39002: Operating Systems Lab
Spring 2017

Assignment 5
Due: March 28th, 11 pm

In this assignment you will be working with the Pintos OS that was discussed in class. Specifically, your job in this assignment is to:

1. Installing Pintos on your laptop/PC, and running user programs on it
2. Understand the source codes, especially the files thread.c, palloc.c, malloc.c, sync.c in the src/threads directory, and any other files that functions in these files call. Also take a look at the src/lib directory. You should try to understand how the thread management (data structures kept, creation, scheduling, blocking/unblocking, context switch, termination) and memory management work in Pintos.
3. Write code to replace the default Pintos scheduler with the following scheduler:
    a. A multilevel feedback queue with two levels, Level 1 and 2, that works as follows. Let the time quanta in the default round robin scheduler in Pintos be T.
        i. Threads are added on creation to Level 1 queue.
        ii. Level 2 queue threads are scheduled only if Level 1 queue is empty.
        iii. Level 1 scheduling policy is round-robin with time quanta equal to T. However, if a thread runs for two time quanta and still does not finish, it is pushed down to the end of the Level 2 queue.
        iv. Level 2 scheduling is round-robin with time quanta equal to 2T. However, if a thread in Level 2 waits for 6T and still does not get the CPU (because Level 1 queue is not empty), it is pushed up to the end of the Level 1 queue.

    The 6T count in Level 2 queue is the waiting time in ready queue WHEN THE LEVEL 1 QUEUE IS NOT EMPTY. If the Level 1 queue is empty, waiting time does not count because that is between Level 2 processes only. So think of it this way: When a process goes to Level 2 queue for the first time, its time count of 6T starts. If Level 1 queue becomes empty, the time count stops where it is. It starts again when there is another process in Level 1 queue. If the process gets scheduled in Level 2 queue, the timer gets reset to 0 (the 6T period starts again, after it finishes its time quanta) and the same above rule applies.

    Note that a running process, be it in Level 1 or Level 2 queues, can block while running. In that case, you should remember which queue it was in, and when it is unblocked, it should be put back in the same queue.

4. Write code to replace the default implementations of malloc/realloc/free in Pintos with a memory management system that uses the **buddy memory allocation algorithm** (read from the text). Use the buddy algorithm to allocate memory inside each page, getting a new page from the page allocator if sufficient memory is not available in the existing pages.

   You will also have to write a function void printMemory(void) (in malloc.c only) to check your memory management, the TAs will also use that. The function, when called from a user program, will print the lists for each page in the following format. You can assume that the minimum allocation size will be 16 (so you don't need to keep track of lists for 2, 4, 8. Note that the maximum allocation size is 2KB in Pintos (you do not have to handle allocations spanning multiple pages). You can call the function after malloc/free to check if the list changed as it should.

```
---------------------------------------------------------------------
No. of pages allocated:

Page 1:
Size 16: <list starting addresses of free blocks of size 16,
sorted in increasing order)
Size 32: Size 16: <list starting addresses of free blocks of size
32, sorted in increasing order)
.
.
Size 2048: <list starting addresses of free blocks of size 2048,
sorted in increasing order)

Page2:
Size 16:.....
Size 32:.....
.
.
Size 2048:...

and so on for all page
---------------------------------------------------------------------
```

Submit the following:

1. A design document (nicely formatted) explaining (i) how the existing Pintos Scheduler works (the scheduling policy is simple, explain how a context switch happens stating the functions called in sequence and what does each function do (ii) data structures you have added/modified (iii) which functions

you have added/modified (for added function (if any), explain what it does; for modified functions explain what you modified) (iv) which files you have modified.

2. A design document with the same things as above for the memory management part. Explain the within-page memory management, not the page allocator).
3. ONLY the files that you have changed in Pintos. You must NOT add any new files. Pintos should build with just the default files replaced with your files. Minimize the number of files that you have to change to do steps 3 and 4.