Task scheduling in real-time systems is to determine an execution order such that all tasks can meet the corresponding deadlines. To do this, we often need an estimate of the worst-case execution time (WCET) and then reserve that amount of CPU time accordingly. An overrun occurs if the real execution time exceeds the estimate WCET. When it is undetected, subsequent tasks may miss their deadline as a consequence of overspending CPU time. One solution to the task overrun problem is to lower the priority of the overrunning task once it consumes the reserved CPU time and only execute it in the background mode. This requires a measurement of task execution time at run time.

In real-time systems, a task may be preempted (by high priority tasks) or blocked (due to unavailable resources locked by low priority tasks). The measurement of execution time at user application level is difficult, since we cannot predict the instances of preemption and blocking. On the other hand, it can be done easily at kernel level. As shown in the following diagram, a task is created, dispatched to run, switched off to waiting or ready state, and deleted (terminated). The measurement of execution time can be done once we mark the instances it is dispatched and accumulate the time it has spent when it is switched off or terminated.

vxWorks allows user tasks to add callback functions (hooks) to kernel at the instances of task creation, task deletion, and context switching. When a hook is called, the pointer to the task’s TCB is passed as a parameter. Thus, we can access TCB and use the spare space in TCB to record additional task information.

What you will do in the assignment:
You need to develop a task set where tasks in the task set simulate operations of an integer binary search tree. A binary search tree is a binary tree data structure which has the following properties:

- Each node has a value.
- A total order is defined on these values.
- The left subtree of a node contains only values less than the node's value.
- The right subtree of a node contains only values greater than or equal to the node's value.

It must contain the following tasks to run in vxWorks environment at the target Intel ARM-Xscale processor:

1. **Searching.** Searching a binary tree for a specific value.
2. **Insertion.** Inserting a specific integer into the tree.
3. **Deleting.** There are several cases to be considered:
   - **Deleting a leaf:** Deleting a node with no children is easy, as we can simply remove it from the tree.
   - **Deleting a node with one child:** Delete it and replace it with its child.
   - **Deleting a node with two children:** Suppose the node to be deleted is called $N$. We replace the value of $N$ with either its in-order successor (the left-most child of the right subtree) or the in-order predecessor (the right-most child of the left subtree).

4. **Traversal.** Elements retrieved in order by recursively traversing the left subtree, visiting the root, then recursively traversing the right subtree. The tree may also be traversed in pre order or post order traversals.

(Go over the knowledge of binary search tree you learned before if necessary.)
Be aware of the issues of data consistency. You need to run the tasks several times (by generating different data-set) during your experiment. Every time after finishing, please show the data set and result(s) on the console.

3. Add vxWorks hooks for task execution time measurement.

4. Feel free to add any tasks to make the above operations more convenient. Assign the tasks with reasonable priorities.

5. Report the accumulated execution times of terminated tasks (the execution times should be printed out on console, and a host task should be developed to inquire the task execution time measured at the target.

6. (Required!) Develop a test case that contains multiple concurrent tasks (e.g., these tasks can be performing the printing of a message on the console) to test your measurement and reporting facility. Note that you can design your own format of reporting and how it is done. Please make reasonable assumptions and don’t use any excessive buffers.

**Relevant vxWorks facilities for this project**

The vxWorks' libraries are more than the followings for the given purpose and of course you're not restricted to use just those. They are listed to give you a basic idea for the project.

- Generating tasks: `taskSpawn()` with task's name, a priority, etc., as arguments can be used to create new task.

- Delaying tasks: `taskDelay()` or `nanosleep()` may be used to put a task into the delayed state.

- Checking an execution time: You may need to handle a timer or times to measure the execution time. However, a simple way would be using `tickSet()` and `tickGet()` to check a relative time elapse.

- Tracking the execution time of a task: There are diverse ideas regarding how to keep track of the execution time for a task which is preempted multiple times. One could be to use VxWorks' task hooks which allow additional routines to be invoked whenever a task is created, switched and deleted. Those libraries are `taskCreateHookAdd()`, `taskCreateHookDelete()`, `taskSwitchHookAdd()`, `taskSwitchHookDelete()`, `taskDeleteHookAdd()`, `taskDeleteHookDelete()` etc. that are
included in taskHookLib. You must be careful to use hook routines since there is a limit of facilities that can be called in Hook routines.

- TCB(Task Control Block) which keeps a context of a task has spare fields for these extensions. TCB's data structure is defined in taskLib.h.

Reading References

- Reading A quick intro to Wind River Workbench development environment would be a great help to start this project.
- Refer to TaskHookLib Document to get used to callback functions.
- All these manuals can be accessed through Wind River Workbench’s HELP menu.

You report should contain:

1. Listing of your source code (including measurement and reporting facility, and test case).
2. Explicit description of your design and the criteria used to decide if the design works correctly. Be careful and comprehensive.
3. The report of task execution time.

Turn in a hardcopy of your listing and any description. Also, you need to email a zipped file to xzou@uh.edu. The zipped file should contain all files and subdirectories in your project directory. Make sure to name your zipped file as following format:

FIRST NAME_LAST NAME_COSC6384 Assignment3.zip