'Frequently Asked Questions for Lab 1'

Homework 1 FAQs

Q. For question 2, 3 and 4, do we have to implement and run them on a specific target?

No. Actually you can complete these questions, using your favorite C compiler.

Q. I keep hearing I need to write a driver function to test this, what is a driver function?

It is a test program you will write and use to test your own code. It basically calls the functions for question 2, 3, 4 that you wrote and test them. Do NOT confuse this with device driver or 'modules' as they are called in linux. You will learn about device drivers in later lab(s).

Q. For question 2, 3 and 4, do we need to spawn a number of tasks and then calculate it's C, T, D values and then calculate blah blah ... ?

No. You can create three seperate arrays in your test function and make up your C, T, D values and call your functions.

Q. For question 2, 3 and 4, can I assume the tasks in the array have priority order?

Yes. The *tid values can all have priorities starting with 1 and ending with Ntasks, from highest to lowest priority.

Q. This is how I have written the driver function, is it ok?

It doesn't really matter how you have written it, as the TA grading it will have his own driver to test it. As long as you follow the exact C function prototypes.

Lab 1 FAQs

Q. How do I get started on Question 1

I would recommend that you start understanding POSIX example code. You will need to understand different libraries, structures and APIs for this lab.
The Open Group has good documentation for signal.h and time.h The VxWorks reference manual that I have attached will be useful in understanding the libraries and APIs.

Q. How do I get started on Question 2

Note: Read questions 2, 3, 4 and 5 before starting on 2, so you can design your fibbocacci function such that the execution time on it is easily computable. Use a large enough data type. Avoid using globals so that two different tasks can call the same fibbonacci function. Just a suggestion, you can implement it how ever you like. If a variable overflow happens, don't worry about it, it's the scheduling theory we are trying to understand using fibbonacci computation, otherwise fib computation isn't that exciting.

Read up on taskSpawn, how and what type of parameters it allows to be passed in. Maybe you can use large enough global and pass a pointer to it to your fib task to compute? Again, implementation details, many ways to skin a cat.

Q. For question 2 and 3, I wrote a recursive Fibbonacci sequence function and I get weird behavior. Sometimes the system reboots etc.

Do **NOT** use recursive function to compute the fibbonacci, use a for loop. Why? Because recursion uses stack, loads of it. What is the stack size of your task? (Hint: Check the parameters passed to taskSpawn) Remember, you don't have unlimited resources in real time.

Q. For question 2 and 3, do I need to save every number of the fibbonacci sequence?

No, absolutely not.

Q. Do I need to print out all the Fibbonacci Sequence?

No. In fact your fibbonacci computing function should not have any printf(s) at all (unless you are trying to debug it). Because when you will be computing the 10/20 millisecond measurement, you want only the time it took for the computation, not any I/O time etc.

Q. After I stop my System Viewer, the trace does not pop up?

Most likely you have killed the target (congratulations). Either you were calculating too big of a number at a very high priority, or you didn't have enough buffer space. Try to learn the options in the System Viewer Configuration. What do the options under Upload Mode do? Change them to see if it helps.

Q. For question 3, I downloaded the code to the simulator, and my timings are inconsistent. Why?

Do **NOT** use the simulator for timing measurements in this or any of the following labs, use the lab targets. The timings are inconsistent because the simulator is running as a application on Windows, which is far from real time.

Q. I have to calculate the Fibbonacci sequence upto "x" terms to achieve 10 ms duration while my friend has to calculate the sequence upto "y" terms to achieve 10 ms duration. Is something wrong with my code?

Everyone in the lab will have different values of the number of terms upto which they will need to calculate the Fibonacci sequence to achieve 10 ms duration.

General Tips

To re-use the function in questions 2-3, it is helpful to write it such that any global variables are not accessed directly by the function. In other words, two of these functions should be able to run with seperate instances of each other and not corrupt one another's execution.

Take a look at the taskSpawn function and note the type of parameters which can be passed to the function. Then consider the size of variable you will need for a large number of Fibbonacci terms. It may be helpful to pass a pointer to a global variable of sufficient size to specify the number of terms to compute in the sequence.

To make the function more re-usable in questions 2-3, it may be nice not to print the result directly in your function. One way to print the result outside of your function would be to pass a pointer to a location to store the result. You also may need to consider the priority of the function spawning the task to determine if the Fibonnacci function will complete before or after the spawning function. See taskPrioritySet and taskIdSelf to manipulate calling function priority if necessary.

You might note that an actual Fibbonacci number at the sequences necessary to give desired execution times of 10 and 20ms would require a number larger than 4 bytes as storage. It is fine to use a 4 byte storage value and ignore the variable overflow. We are not concerned with the actual number, we are just using the Fibbonacci routine as a loading function to test scheduling theory.

Common Pitfalls to Avoid:

1) Do not use printf inside your fibbonacci function. This will make your function timing unpredictable. Use an external function to spawn your fibbonacci calculation function and print the value from this function.

2) Do not try to save every number in the fibbonacci sequence... or if you do make sure you have buffering necessary to do so. It is not required that you save (or show) every number, you just want to calculate the Nth term. All other values are intermediate.

3) If you are unable to download your windview capture (the program hangs), you can suspect that the CPU on the target is most likely overloaded. This is most likely because your code has run amok. Look for things like infinite loops, uninitialized pointers, or exceeded array bounds. Think about the size of variables declared in the function and the stack size specified by taskSpawn. Another possibility is that someone left a task running on this target that is bogging it down. It is a good idea to always restart the target before you try your code the first time.

A good way to setup this test would be to call a test function that spawns two different task.. a high and low priority Fibbonacci function spawning task. The high and low priority tasks would spawn the Fibbonacci functions at the given rate using delays between taskSpawn calls. Think hard about the priority of each of these functions and the priority of the spawned Fibbonacci computing functions. How long should the delays be between function spawns?

Use your knowledge of scheduling theory to determine the number of times each of the low priority and high priority functions should be spawned in order to adequately illustrate system scheduling success or failure. Success would mean that there should be idle execution time during the execution of these tasks. Failure would mean that these tasks would overlap and miss their deadlines. Either of these scenarios will be easy to show using a Windview plot.

Refer to the "Scheduling Point Test" section in the Real Time Embedded Components and Systems Online text for more information about our friends Lehoczky, Shah, and Ding.

This page is last updated on September 24, 2013

<< Back