# 第二周试题

本试题测验范围：课时《临界区保护》至《任务的状态查询》

用户名：\_\_\_sharemon+139\_\_\_ 联系邮箱：\_\_\_\_\_sharemon@163.com\_\_\_\_\_\_

（用户名格式： 学号+电子发烧友用户名）

**温馨提示：**

亲爱的同学，你好！

为了帮助你测试自己对课程内容的掌握程度，特地编写了此试题。在做题之前，请摒弃学生时代的那种答题方式，不要为了分数而答题，而是以自我检测为目标。答题时如遇难题可随时翻阅课件资料，编写答案时可自由发挥、多写多画图，以便更好地检验自己的不足！

希望能够通过视频课程以及试题检测，更好地帮你掌握这些技术知识。如果通过课程学习带来的收益能够帮助你提升技术水平， 甚至获得工作提升，那将是我莫大的荣幸！

@01课堂 李述铜

**注意事项：**

1. 所有试题均基于《自己动手从0到1写嵌入式操作》课程中所用的软硬件完成，除非特别指出，否则试题内容的各项基础条件均与课程中所指一致
2. 试题类型包含选择题、简单题、程序设计题。选择题和简答题请直接在此文档中填写；程序设计题请在文档中说明设计原理，随同工程一起提交
3. 本试题非100分制，而是答对获得相应的分数，答错不计分，最后一起累计总分
4. 请在截止时间前提交试题，逾期不批改，也不计入相应分数
5. 完成试题后，提交至course@ilishutong.com和[qianshanshan@elecfans.com](mailto:qianshanshan@elecfans.com)。分别添加两个附件：文档不压缩，工程文件压缩，文件命名: 第?周试题 用户名.扩展名。

# 选择题

1、单选题。假设内核的系统时钟节拍为10ms。现有任务延时，首先考虑提升系统效率，其次兼顾延时精度。不考虑其它要求，请选择合适的延时接口。（8分）

1）、延时0.1ms，选择接口\_B\_\_

2）、延时1ms，选择接口\_B\_\_

3)、延时100ms，选择接口\_A\_\_

4)、延时1000ms，选择接口\_A\_

请选择以下延时接口，A或B。

A：tTaskDelay(ticks)：延时指定ticks。

B：软件延时delay(ms)：延时指定的ms数。

2、多选题。假设系统时钟节拍为10ms。任务A调用了tTaskDelay(1)。请问当任务从tTaskDelay()返回时，实际的延时时间可能是\_\_\_3,4\_\_\_\_（8分）

1)、不多不少, 正好10ms

2)、少于10ms，比如1.5ms

3)、比10ms略多，但比20ms少

4)、多于10ms，比如甚至是1s。

3、单选题。假设系统中有资源A，只有任务B和中断C对资源A写访问。为了避免写冲突，需要使用tTaskEnterCritical()/tTaskExitCritical()进行保护。请问中断C中是否需要在写过程中添加这种保护？（B） A、需要； B、不需要

# 简答题

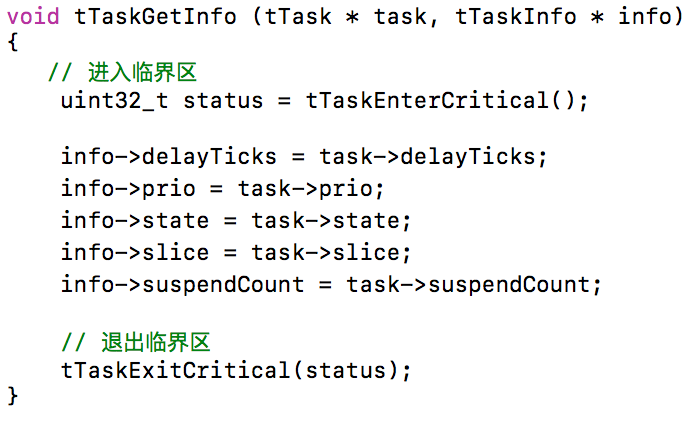
1、请简述tinyOS中提供了哪几种任务删除方式（2分），并解释删除过程原理？（5分）为什么要提供这两种删除方式？（5分）

有两种删除方式:

1. 强制删除：直接删除指定任务，并调用任务清除回调函数清理被清除任务使用到的资源。
2. 请求删除：不直接删除任务，而是通过标志位向任务发出请求。当被删除任务执行时，检查请求标志位，如果被请求删除，则可以先对自己使用的资源进行清除，然后调用清除回调函数对系统资源进行清理，最后删除自己。

因为这两种删除方式各有缺点，要根据情况使用。

1. 强制删除不能保证被删除任务所使用的资源完全被释放。
2. 请求删除不能保证被删除任务立即被删除。

2、任务状态查询接口实现如下。请说明为什么要额外传入一个tTaskInfo结构变量，然后在tTaskEnterCritical()/tTaskExitCritical()将需要的信息从tTask结构中拷贝出来，而不是直接让调用者访问tTask结构；除了向调用者隐藏tTask的实现细节外，是否有其它原因？（10分）

可以避免用户的误操作改变tTask中变量的值。

3、请绘制一幅图，描述当前课程内容中各内核数据结构的组织关系示意图。请至少包含以下内容：

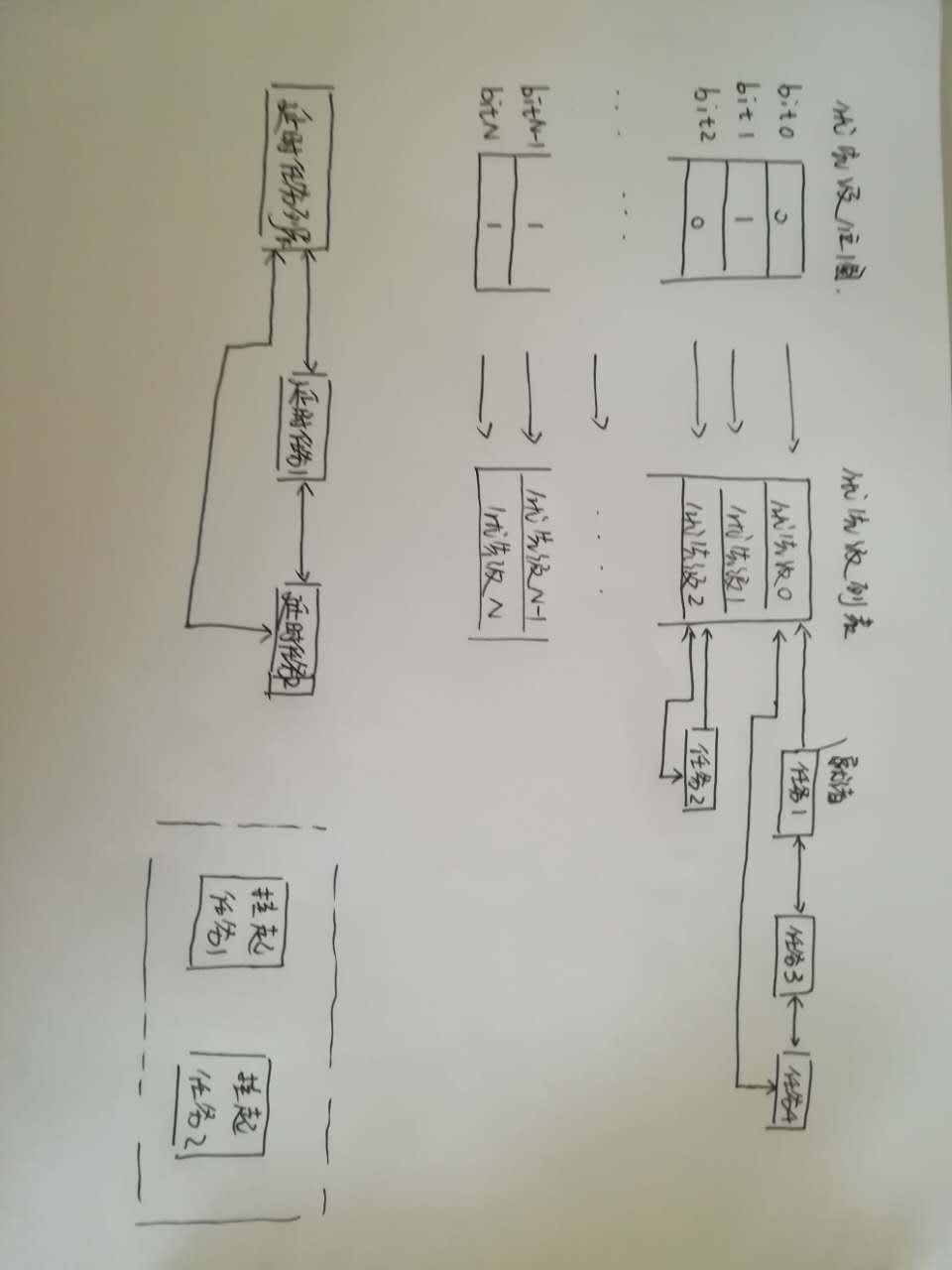
1）、所有就绪任务的组织和连接；（5分）

2）、延时列表中各个任务的组织与连接；（5分）

3）、优先级位图和任务的多优先级队列之间的联系（5分）

在课程中实际上有类似的图，但请参考之前先自行回顾。本题主要目的是让你理一下rtos中的数据结构组织。实际上在rtos的知识重点中，数据结构的组织是一个非常有意思且重要的知识点。理解其组织方式，可以说已经对这个rtos的核心就已经掌握了80%以上。

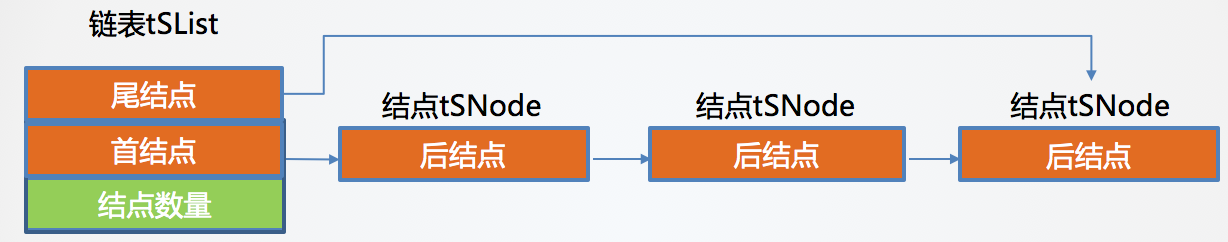
注：如不方便，图可手绘，拍照贴上来。如画不出来，请参考课程中的图，然后再自行绘制。



# 程序设计

以下设计题，任选1个完成。如有兴趣可全部完成，在同一份工程上完成，得分可累加。

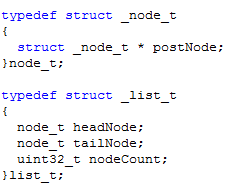
请在完成代码实现后，简要说明其实现原理，无原理说明的算0分。

1. tinyOS中实现了一种双向链表，但是这种链表比较复杂，且每个结点包含了两个指针，占用较多资源。有些情况下，一个简单的单链表结构可以就可以满足需求，如下所示。（15分）

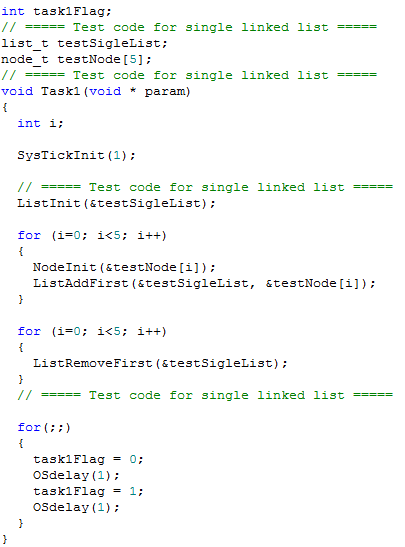
请实现这种数据结构，至少包含以下操作接口。

* 初始化链表
* 获取链表中结点数量
* 获取链表中第一个结点
* 获取链表中最后一个结点
* 将指定结点插入到链表头部
* 将指定结点插入到链表尾部
* 移除第一个结点

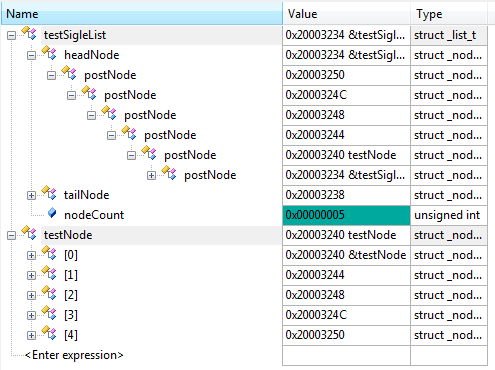
请实现上述接口，并自行添加测试代码验证正确性。

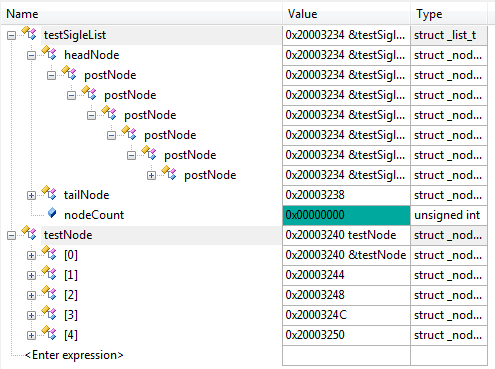
对node和list进行了重新的定义：

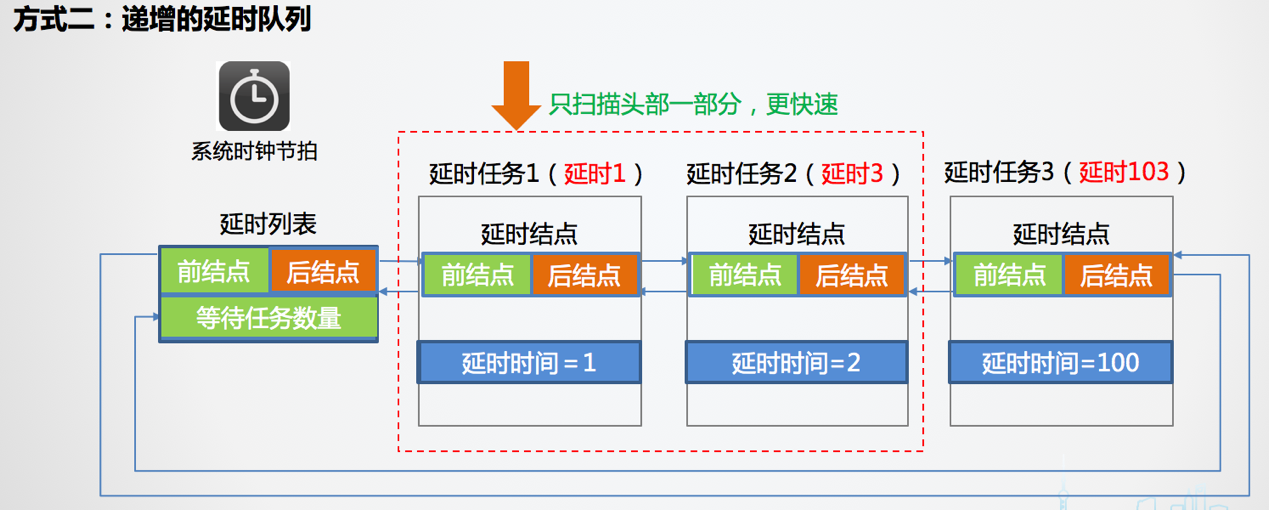
列表操作函数见附件工程的list.c文件

测试函数：

测试结果：

在第一个for循环之后，

在第二个for循环之后，

1. 在【任务延时队列】中，我们提到了更为快速的第二种延时处理方案，请实现该方案替换掉原有的低效方案。或者，如果您有更好的方案，请设计出来。对该方案不熟悉的同学，请看【任务延时队列】第4分30秒。

请实现上述机制，并简要说明你是如何测试确认完成了这样一种方案。（15分）

具体实现见附件TimeTaskWait函数和TaskSystemTickHandler函数。

新建三个任务，task1，task2，task3，优先级分别为0，1，2，标志位反转延时分别为4，7，6。设置断点调试程序。

当三个任务都进入延时列表的时候，可以看到延时列表的顺序是headNode->task1node->task3node->task2node。此时，task1的delayTick值为4，task3的delayTick为2（表示延时为4+2），task2的delayTick为1（表示延时为4+2+1）。符合要求。

在SysTick\_Handler开头设置断点，可以看到，每次执行完SysTick\_Handler，只有task1的delayTick减1，其他task的delayTick不变。

当task1延时结束后，task1被移出delay链表。task3变成延时链表的第一个节点。

当task1执行完task1Flag的翻转之后，再次进入延时链表时，task1被放在延时链表的最后，且delayTick为1，表示task1的延时为（2+1+1=4）

由这些现象可见，上图中所示的延时方案已被实现。