一 造成函数重入的主要原因，以及保护不可安全重入代码的方法。

由于终端的异步属性，因此终端可能会打断正在被执行的每个函数A而将程序的控制权交给相应的中断服务程序，这是会有两种可能性出现：

1. 中断服务程序中又重新调用了刚才被中断的函数A
2. 在一个抢占式多任务的RTOS内核中，中断服务程序激活了一个更高优先级的任务，并且在中断返回时由RTOS内核的调度器将控制权交给了这个高优先级任务，这个任务接着又重新调用了刚才被中断的函数A
3. 递归函数的自我递归调用

二 堆栈缓冲区溢出一般的原理是什么？造成的原因？可能的后果？

[堆栈溢出](http://baike.baidu.com/subview/770499/770499.htm)就是不顾[堆栈](http://baike.baidu.com/subview/93201/93201.htm)中分配的局部[数据块](http://baike.baidu.com/view/702806.htm)大小，向该数据块写入了过多的数据，导致数据越界，结果覆盖了老的堆栈数据。

循环的递归调用，如果使用的大数据结构的局部变量，也可能导致堆栈溢出。

轻则造成程序运行失败，系统崩溃，系统重新启动，重则可以利用它执行非授权指令，取得系统特权，进行非法操作。

三 全局数据区，堆与栈的区别，举例说明C语言中变量如何在这三种内存区域进行分配？

堆被称为动态内存，由堆管理器（系统里的大人物，山高皇帝远不用去管它）管理，程序中可以使用malloc函数来（向堆管理器）申请分配堆内存，使用完后使用free函数释放（给堆管理器回收）。堆内存的特点是：在程序运行过程中才申请分配，在程序运行中即释放（因此称为动态内存分配技术）。  
栈是C语言使用的一种内存自动分配技术（注意是自动，不是动态，这是两个概念），自动指的是栈内存操作不用C程序员干预，而是自动分配自动回收的。C语言中局部变量就分配在栈上，进入函数时局部变量需要的内存自动分配，函数结束退出时局部变量对应的内存自动释放，整个过程中程序员不需要人为干预。

1、栈区（stack）— 由编译器自动分配释放 ，存放函数的参数值，局部变量的值等。

2、堆区（heap） — 一般由程序员分配释放，例如malloc函数分配的数据区， 若程序员不释放，程序结束时可能由OS回收 。

3、全局区（静态区）（static）—，全局变量和静态变量的存储是放在一块的，初始化的全局变量和静态变量在一块区域， 未初始化的全局变量和未初始化的静态变量在相邻的另一块区域。 - 程序结束后有系统释放

四 在C语言中，关键字static有三个明显的作用：  
1). 在函数体， 一个被声明为静态的变量在这一函数被调用过程中维持其值不变。  
2). 在模块内（但在函数体外），一个被声明为静态的变量可以被模块内所用函数访问，但不能被模块外其它函数访问。它是一个本地的全局变量。  
3). 在模块内， 一个被声明为静态的函数只可被这 模 一模块内的其它函数调用。那就是，这个函数被限制在声明它的模块的本地范围内使用。

五 CONST的优点  
关键字const的作用是为给读你代码的人传达非常有用的信息， 实际上， 声明一个参数为常量是为了告诉了用户这个参数的应用目的。如果你曾花很多时间清 其它人留下的 圾 理其它人留下的垃圾， 你就会很快学会感谢这点多余的信息。（当然，懂得用const的程序员很少会留下的垃圾让别人来清理的。）通过给优化器 些附加的信息 一些附加的信息， 使用关键字const也许能产生更紧凑的代码。合理地使用关键字const可以使编译器很自然地保  
护那些不希望被改变的参数，防止其被无意的代

六 一个定义为volatile的变量是说这变量可能会被意想不到地改变， 这样， 编译器就不会去假设这个变量的值了。精确地说就是，优化器在用到这个变量时必须每次都小 地重新读取这个变量的值 心地重新读取这个变量的值，而不是使用保存在寄存器里的备份。下面是volatile变量的几个例子：  
1). 并行设备的硬件寄存器（ 如： 状态寄存器）  
2). 一个中断服务子程序中会访问到的非自动变量(Non-automatic variables)  
3). 多线程应用中被几个任务共享的变量

七 static用法

在函数体，一个被声明为静态的变量在这一函数被调用过程中维持其值不变在模块内（但在函数体外），一个被声明为静态的变量可以被模块内所用函数访问， 但不能被模块外其它函数访问。它是一个本地的全局变量在模块内，一个被声明为静态的函数只可被这一模块内的其它函数调用。那就是，这个函数被限制在声明它的模块的本地范围内使用。

八 编译器和汇编器的作用  
编译器将C文件转换成为汇编文件汇编器将汇编文件转换成为二进制指令流\*.o文件（ 目标文件）每个目标文件是独立编址的，也就是说每个目标文件的第一条指令都从相同的地址开始存放

九 链接器的作用

将多个目标文件或库文件按照各文件中段进行统一编址生成 个完整的统一的地址映像 一个完整的统一的地址映像嵌入式系统中一般生成一个绝对地址映像在有MMU的系统中可以为每个任务单独分配一个地址空间

十 函数指针的用途，主要为以下三者：  
多态 (polymorphism)：指用一个名字定义不同函数，这函数执行不同但又类似的操作，从而实现“一个接口，多种方法”。

回调 (call-back)： 由操作系统来调用用户编写的函数，或者由底层函数调用上层函数。由于操作系统的代码在用户代码之前就已经编译完成， 因此由操作系统发起的回调 般 一般都必须通过将用户编写函数的函数指针传递给操作系统，再由操作系统实现调用

多线程 (multithreading)：将函数指针传进负责建立多线程的API 中： 例如 Win32 的 CreateThread( pF )。

十一 栈的作用  
⎫利用堆栈传递函数调用的参数  
⎫利用堆栈保存函数调用的返回地址（对于中断处理程序还包括程序状态字寄存器）  
⎫利用堆栈保存在被调函数中需要使用的寄存器的值

⎫利用堆栈实现局部变量

十二 内存泄漏的原因  
⎫忘记释放已分配内存  
⎫程序员A分配了内存，被程序员B使用，但没人释放  
⎫由malloc分配的内存地址只能由free释放，如果指针被改变，free将失效  
⎫缓冲区溢出损害了被分配内存的头信息