1.驱动开发（13道）

1.1 Linux驱动程序的功能是什么?

（1）对设备初始化和释放。
（2）进行内核与硬件的数据交互。
（3）检测和处理设备出现的错误。

1.2内核程序中申请内存使用什么函数?

答案：kmalloc()、kzalloc()、vmalloc()。

解读：

（1）void *kmalloc(size_t size, gfp_t flags);

①申请连续的物理内存，这对于要进行DMA的设备十分重要，但大小不能超过128KB，其中有16B是被页描述符占用了。

②较常用的flag有GFP_ATOMIC（分配内存的过程是一个原子过程）、GFP_KERNEL（正常分配内存）、GFP_DMA（给DMA控制器分配内存）。

③对应的内存释放函数为void kfree(const void *objp)。

（2）void *kzalloc(size_t size, gfp_t flags);

①kzalloc()相对kmalloc()只是额外增加了__GFP_ZERO标志，除了申请内存外，还会对申请到的内存内容清零。

②对应的释放函数也是kfree()。

（3）void *vmalloc(unsigned long size);

①申请虚拟地址连续的内存空间，但其对应的物理内存不一定连续，因此对申请的内存大小没有限制。

②对应的内存释放函数为void free(const void *addr)。

③注意：vmalloc()和vfree()可以睡眠，因此不能在中断上下文调用。

1.3内核程序中申请内存和应用程序时申请内存有什么区别?

答案: 内核中申请内存空间用的是函数kmalloc、kzalloc、vmalloc，应用程序申请内存用的函数是malloc。

解读:

（1）kmalloc/kzalloc直接分配连续的物理地址（虚拟地址也是连续的）。

（2）vmalloc分配连续的虚拟地址，但物理地址不一定连续。分配时实际分配了物理内存，不过这个物理内存页面是在公共的页表进行了映射，并没有在本进程的页表进行映射，当访问这段内存时，触发do_page_fault异常（缺页中断）才完成页表的同步工作。

（4）malloc是用户空间申请内存的方法，分配连续的虚拟地址，物理地址一般不会连续。在分配时并没有做实际物理页的分配动作，实际分配物理页的动作是在do_page_fault异常（缺页中断）处理中完成的。

1.4自旋锁和信号量在互斥使用时需要注意什么？在中断服务程序里面的互斥是使用自旋锁还是信号量?

（1）使用自旋锁的进程不会睡眠， 而使用信号量的进程会睡眠。

（2）中断服务程序使用的是自旋锁，原因是中断服务程序处于中断上下文，中断上下文是不参与调度的，也就没有保护现场与恢复现场，一旦睡眠就回不来了。

1.5驱动卸载异常可能是由什么原因引起的?

可能是因为有进程在使用该模块。

1.6 Linux中引入模块机制有什么好处?

（1）应用程序在退出时，可以不管资源的释放或者其他的清除工作，而把这些任务交给模块退出函数(exit)。

（2）模块机制有助于缩短模块的开发周期，因为模块的安装和卸载都很方便。

1.7 Linux设备驱动程序中，使用哪两个函数进行中断处理程序的注册和注销?

（1）注册中断：

```
int request_irq(insigned int irq, irqreturn_t (*handler)
(int, void *, struct pt_regs *), unsigned long flag, const char *dev_name, void *dev_id);
```

参数的意义依次是中断号，中断处理函数，中断管理有关掩码，中断请求设备名，中断信号线。

（2）注销中断：

```
void free_irq(insigned int irq, void *dev_id);
```

参数的意义分别是中断号和中断信号线。

1.8写一个中断服务程序需要注意哪些地方?

（1）中断服务程序应该尽可能短，把能放在底半部（tasklet、workqueue）的任务尽量放在底半部。

（2）中断服务程序中不能有阻塞操作，因为中断期间是完全占用CPU的，不存在内核调度，中断被阻塞住，其他进程将无法推进。

（3）中断服务程序的返回值要用操系统定义的宏，而不能用自己定义的。

（4）中断服务程序不能有不可重入的操作，如浮点数计算、printf()等。

1.9 Linux系统打开设备文件，进程可能处于三种基本状态，如果多次打开设备文件，驱动程序应该实现什么?

互斥。

1.10简述static对于工程模块化的作用。

static可以让全局变量或函数的作用域限制在当前模块，不会与其他模块发生冲突。因为在嵌入式系统中，一个程序可能是很多程序员共同完成的，在定义变量及函数的过程中，可能会重命名，给系统集成带来麻烦。

1.11并发是什么？驱动里面为什么要有互斥控制？如何实现？

（1）并发是指多个执行单元同时、并行地被执行，而并发的执行单元对共享资源（硬件资源和软件上的全局变量、静态变量）的访问很容易导致竞态。

（2）解决竞态问题的途径是保证对共享资源的互斥访问。

（3）访问共享资源的代码区域被称为临界区，临界区需要用某种互斥机制加以保护，中断屏蔽、原子操作、信号量、自旋锁都是Linux设备驱动中可以采取的互斥机制。

1.12 Linux内核有哪些同步方式？

原子操作、信号量、自旋锁、读写锁、顺序锁等。

1.13在一个多任务嵌入式系统中，有一个CPU可直接寻址的32位寄存器REGn，地址为0x1F000010，编写一个安全的函数将寄存器REGn的指定位反转？

答案：

```
void bit_reverse(uint32_t nbit)
{
    *((volatile unsigned int *)0x1F000010) ^= (0x01 << nbit);
}
```

注意：

（1）指定位反转用异或^。

（2）由于是寄存器地址，因此强制类型转换的时候要加上volatile。

讨论 评论

帖子还没人回复，快来抢沙发！