考试科目名称 操作系统原理与实践 II （A 卷）

考试方式：闭卷 考试日期：2007 年 月 日 教师：

系（专业）： 年级： 班级：

学号： 姓名： 成绩：

题号	一	二	三	四	五
分数					

一、名词解释（共 5 题，每题 5 分，计 25 分）

1、封装例程

屏蔽底层复杂性，将系统调用封装成应用程序能够直接调用的函数(库函数)，供用户在用户空间编程使用。

2、实时信号

实时信号也称"可靠信号"，支持排队。当一个实时信号发送给一个进程时，不管该信号是否已经在进程中注册，都会被再注册一次。其信号值位于信 SIGRTMIN 和 SIGRTMAX 之间。

3、工作队列

是另外一种将工作推后执行的形式，其核心思想是将推后工作交由一个内核线程去执行，允许被重新调度甚至是睡眠。因此，通过工作队列执行的代码能占尽进程上下文的所有优势。

4、IPC 键

IPC 对象的外部表示，可由程序员选择。如果键是公用的，则系统中所有进程通过权限检查后，均可找到和访问相应 IPC 对象。如果键是私有的，则键值为 0。

5、物理地址扩展

采用 36 位地址线，支持对大于 4GB 的内存空间的寻址的技术，最大寻址能力可达到 64GB。

二、计算题（共 1 题，每题 15 分，计 15 分）

某基于 i386 体系结构的逻辑地址的段标志符（即段选择子）为 0x001F，其段内偏移为 0x00000005，CR3 的地址信息为 0x00102000：

A）试根据下述 GDT 及 LDT 信息，计算出线性地址。

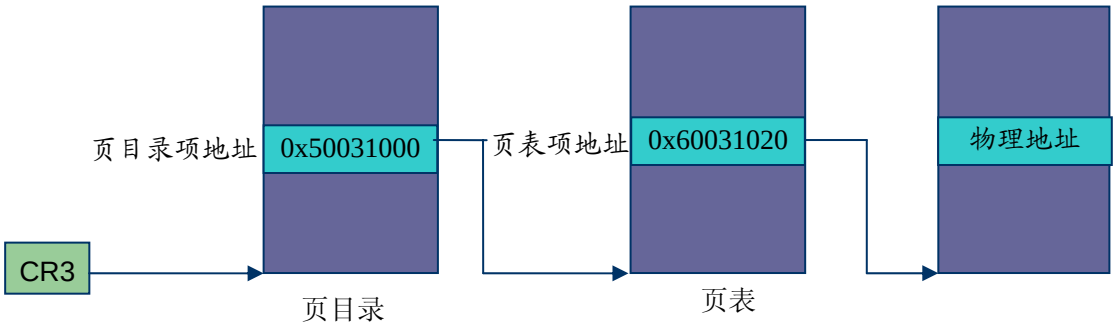
B）试结合下图中的页目录及页表信息，计算出与该线性地址对应的页目录项地址、页表项地址及物理地址。

INDEX	基地址
0	NULL
1	0x20021400
2	0x20021401
3	0x20021402
4	0x20021403
5	0x20021404
6	0x20021405

GDT 中各段描述符的基地址信息

INDEX	基地址
0	NULL
1	0x30021401
2	0x30021402
3	0x30021403
4	0x30021404
5	0x30021405
6	0x30021406

LDT 中各段描述符的基地址信息



段选择子的二进制形式：0000 0000 0001 1111，可得 T = 1，索引值为 3。因此，应从 LDT 中获取基地址信息，基地址值为 0x30021403。

线性地址为 0x30021403+0x00000005 = 0x30021408，其二进制格式为

0011 0000 0000 0010 0001 0100 0000 1000

其页目录索引为 0xc0，页表索引为 0x21，页内偏移是 0x408。

页目录项地址：0x00102000 + 0xc0*4 = 0x00102300。

页表项地址：0x50031000 + 0x21*4 = 0x 50051084

物理地址：0x60031020 + 0x408 = 0x60031428

三、简答题（共 4 题，每题 8 分，计 32 分）

1、试述消息队列、共享主存及信号量通信机制的特点及在应用场景上的主要差别。

消息队列是一个格式化的可变长信息单位，允许一个进程向任何其他进程发送一个消息。当一个进程收到多个消息时，可将它们排成一个消息队列。消息队列以异步方式为通信频繁、但数据量少的进程通信提供服务。

共享主存的核心思想是让多个进程共享的一块内存区域，不同进程可把共享内存映射到自己的一块地址空间，处于共享内存区的进程对该区域的操作是互见的。共享内存主要用于为数据量大的进程间通信提供服务。

信号量是具有整数值的对象，表示可用资源的数量。申请资源时减 1，资源不足时进程可以睡眠等待、也可以立即返回。信号量主要用于实现与其他进程同步和互斥。

2、试述软中断、tasklet 的特点及在实现机制上关系。

软中断是一组静态定义的下半部接口，共有 32 个，必须在编译阶段静态注册。软中断可以在所有处理器上同时执行（即使两个类型相同的软中断），一个软中断不会抢占另一个软中断。在一个软中断处理程序运行的时候，当前处理器上的软中断被禁止。

tasklet 是一种基于软中断实现的、动态创建的下半部机制。与软中断不同，类型相同的 tasklet 不能同时执行，但两种不同类型的 tasklet 可以在不同处理器上同时执行。实际上，tasklet 由 HI_SOFTIRQ 及 TASKLET_SOFTIRQ 两类软中断代表组成，前者的优先级高于后者。

在实际应用能够，通常应优先考虑使用 tasklet，软中断适用于执行频率很高且连续性要求很高的情况。

3、试述 Linux 中的线程实现机制的特点，并说明如何区别内核线程与普通进程。

从内核角度，Linux 中没有线程概念内核没有针对所谓线程的调度算法或数据结构来表征线程，把所有线程当进程来实现。线程被视为与其他进程共享某些资源的进程，都有自己的进程描述符 task_struct。在 linux 中“线程”，仅仅是表示多个进程共享资源的一种说法。

内核线程与普通进程的区分主要通过进程描述符中的 mm 和 active_mm 来实现。 对普通进程而言，这两个字段存放相同的指针。对内核线程而言，由于不拥有任何内核描述符，因此，mm 字段总为 NULL，当内核线程运行时，active_mm 被初始化为前一个运行进程的 active_mm 的值。

4、试述 Linux 内存管理中 Slab 分配器的实现机制。

slab 分配器将内存区看成对象，并将对象按照类型分组成不同的高速缓存，每个高速缓存都是同种类型内存对象的一种“储备”。每个 slab 由一个或多个连续的页框组成。每个 slab 有三种状态：1）全满状态。全满意味着 slab 中的对象全部已被分配出去；2）半满状态，半满介于两者之间；3）全空状态。全空意味着 slab 中的对象全部是可用的。

slab 对象分配优先从半满的 slab 满足请求；如果没有半满状态 slab，则从全空的 slab 中取一个对象满足请求；如果没有空的 slab，则向伙伴系统申请页面生成一个新的 slab。

Linux 的 slab 分配器的高速缓存包括普通高速缓存和专用高速缓存两种类型。其中普通高速缓存共 2 组 26 个（一组用于 DMA 分配，另一组用于常规分配），由 slab 分配器自己使用，可根据大小分配内存。专用高速缓存(special cache)由内核其余部分使用，可根据类型分配。

根据类型分配

四、程序分析题（共 1 题，每题 14 分，计 14 分）

结合__do_softirq()及 ksoftirqd 内核线程的核心代码，分析软中断的核心处理机制（可在关键代码处注解说明或在代码后面的空白处结合关键代码解释说明）。

tasklet_action(struct softirq_action *a)

```
static void tasklet_action(struct softirq_action *a){
    struct tasklet_struct *list;
    local_irq_disable();
    list = __get_cpu_var(tasklet_vec).head;
    __get_cpu_var(tasklet_vec).head = NULL;
    __get_cpu_var(tasklet_vec).tail = &__get_cpu_var(tasklet_vec).head;
    local_irq_enable();
    while (list) {
        struct tasklet_struct *t = list;
        list = list->next;
        if (tasklet_trylock(t)) {
            if (!atomic_read(&t->count)) {
                if (!test_and_clear_bit(TASKLET_STATE_SCHED, &t->state))
                    BUG();
                t->func(t->data);
                tasklet_unlock(t);
                continue;
            }
            tasklet_unlock(t);
        }
        local_irq_disable();
        t->next = NULL;
        *__get_cpu_var(tasklet_vec).tail = t;
        __get_cpu_var(tasklet_vec).tail = &(t->next);
        __raise_softirq_irqoff(TASKLET_SOFTIRQ);
        local_irq_enable();
    }
}
```

tasklet_action()的执行过程如下:

- 禁止中断（此时无须保存其状态），并为当前 CPU 检索 tasklet_vec/task_hi_vec
- 将当前处理器上的该链表设置为 NULL，达到清空效果
- 打开中断
- 循环遍历获得链表上的每个待处理 tasklet
  - 如果是多处理系统，通过检查 TASKLET_STATE_RUN 状态标志来判断这个 tasklet 是否正在其他处理器上运行
    - ✓ 若设置，将任务描述符重新插入到 tasklet_vec/task_hi_vec 链表，再次激活 TASKLET__SOFTIRQ 或 TASKLET__HI_SOFTIRQ
    - ✓ 若未设置，则将其状态标志为 TASKLET_STATE_RUN，在本 CPU 执行

- 检查 count 是否为 0，确保 tasklet 没有被禁止
  - ✓ 若禁止，则清除 TASKLET_STATE_RUN 状态，将任务描述符重新插入到 tasklet_vec/task_hi_vec 链表，随后再次激活 TASKLET__SOFTIRQ 或 TASKLET__HI_SOFTIRQ
- 如果 tasklet 被激活，则清除 TASKLET_STATE_SHED 标志，执行对应处理程序

五、程序设计题（共 1 题，每题 14 分，计 14 分）

基于消息队列通信机制，采用 C 语言编写两个程序，两程序功能描述如下：

1）程序 A 接收用户输入的加减法运算表达式，并通过消息队列将运算表达式发送给程序 B。加减法运算表达式格式如下：

a) 加法运算：op1 + op2

b) 减法运算：op1 – op2

2）程序 B 从消息队列中读取程序 A 发送的加减法运算表达式，并将计算结果通过同一消息队列返回给用户。若用户输入表达式有错，则返回“表达式错误”提示信息给用户。

请根据上述功能描述，给出程序 A 及程序 B 的代码。

3）程序实现中可能涉及到的字符串处理函数说明如下：

a) *strchr(char *str, char c);

在字符串 str 中查找给定字符 c 的第一个匹配之处。查找成功时，将返回该字符所对应的内存地址；否则，返回 NULL。

b）float atof(const char *nptr);

将字符串 nptr 转换成符点数。

答案略。