LAB7 田新韬 2012011268

Ex0

先 diff 修改后的 lab6 与原 lab6，再 patch 到 lab7 中
更新了 trap.c 中时钟中断的处理

不知道为什么在做到 lab678 的时候编译经常会报空间不足..跑个测例要一顿删东西..

Ex1 理解内核级信号量的实现

Make grade 除了 priority 通不过其他都可以通过（调度还是用的 round-robin 没用 stride）

```
priority:                 (16.4s)
  -check result:                                WRONG
   !! error: missing 'sched class: stride_scheduler'
   !! error: missing 'stride sched correct result: 1 2 3 4 5'

  -check output:                                OK
sleep:                    (12.2s)
  -check result:                                OK
  -check output:                                OK
sleepkill:                (3.1s)
  -check result:                                OK
  -check output:                                OK
matrix:                   (16.1s)
  -check result:                                OK
  -check output:                                OK
Total Score: 183/190
make: *** [grade] Error 1
```

请在实验报告中给出内核级信号量的设计描述，并说明其大致执行流程。

信号量定义在 sem.c/sem.h 中，

```
typedef struct {
    int value; //信号量的值
    wait_queue_t wait_queue; //信号量对应的等待队列
} semaphore_t;
void sem_init(semaphore_t *sem, int value); //初始化
void up(semaphore_t *sem); //V()
```

void down(semaphore_t *sem); //P()

bool try_down(semaphore_t *sem); //不进入等待队列的 P 操作

其中，V/P/不进入等待队列的 P 操作，均采用禁用中断的方式保证原子性。

初始化信号量将 value 初始化为给定值，等待队列初始化为只有一个成员的队列。P 操作若等待队列为空则信号数值加一，若不为空则唤醒队首成员。V 操作若信号值大于 0 则减一，若等于 0 则将当前进程放入等待队列并调度到其他进程，同时使用 wakeup_flag 作为判断该进程是否消耗信号量的判断标志位。

请在实验报告中给出给用户态进程/线程提供信号量机制的设计方案，并比较说明给内核级提供信号量机制的异同。

信号量机制（申请，初始化，V 操作，P 操作，释放）的设计与内核态相同，只是无法通过禁用中断来保证原子性，可以通过系统调用的方式把信号量操作转移到内核态中。

Ex2 完成内核级条件变量和基于内核级条件变量的哲学家就餐问题

按照注释翻译代码即可，完成了 monitor 与 check_sync 中编码。

请在实验报告中给出内核级条件变量的设计描述，并说明其大致执行流程。

条件变量的数据结构：对应信号量，进程数量，所属管程

管程的数据结构：mutex 信号量，next 信号量，next_count 等待唤醒的进程数量，cv 条件变量。

在管程中，使用信号量机制来保证互斥。

条件变量在管程初始化（设置 next，mutex，创建条件变量）时进入其生命周期，其主要操作是 cond_signal 和 cond_wait。Cond_signal 用于激活等待管程的进程，首先判断 cv.count，若不大于 0，表示没有因为该条件变量 cond_wait 而 sleeping 的进程，直接返回；若大于 0，唤醒 cv.sem 里的线程，同时把自身进程挂在 monitor.next 上，并操作 monitor.next_count++。对于 Cond_wait，cv.count 加一，同时判断 monitor.next_count，若不大于 0，唤醒 monitor.mutex，同时该进程挂在 cv.sem；若大于 0，则唤醒 monitor.next，同时该进程挂在 cv.sem。

请在实验报告中给出给用户态进程/线程提供条件变量机制的设计方案，并比较说明给内核级提供条件变量机制的异同。

在用户态实现条件变量与实现信号量的原理相同，通过系统调用转移到内核态，即把所有对管程的操作均封装成系统调用即可。

请在实验报告中回答：能否不用基于信号量机制来完成条件变量？如果不能，请给出理由，如果能，请给出设计说明和具体实现。

能，把管理管程的 mutex 和 next 信号量变为互斥锁即可，互斥锁在管程初始化时申请。