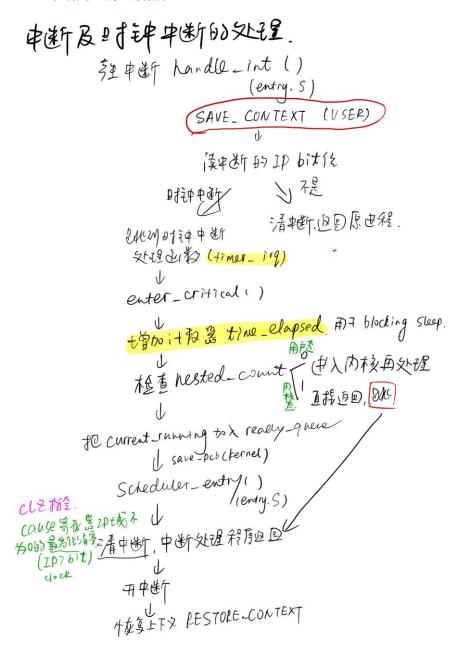
Project3 Preemptive Kernel 设计文档

中国科学院大学 李静逸 2017.11.8

1. 时钟中断与 blocking sleep 设计流程

(1) 中断处理的一般流程



(2) 你所实现的时钟中断的处理流程,如何处理 blocking sleep 的 tasks;如何处理用户 态 task 和内核态 task

Blocking sleep 的 tasks 的处理方法: 时钟中断会把用户态 tasks 阻塞,然后调用 scheduler_entry(),每次调用 scheduler 函数时,都在开头调用 check_sleeping 函数,check_sleeping 函数会通过 get_timer()函数得到当前时刻的时间,存到 current_time 变量里,然后遍历 sleep_wait_queue 队列,把所有 deadline 超过 current_time 的 tasks 唤醒,放入就绪队列 ready_queue 里。

用户态 task 的处理方法:通过检查 task 的 nested_count 判断是用户态还是内核态,如果是用户态,则进入内核态,并阻塞当前 task,把它放入就绪队列,调用 scheduler 函数获取下一个 task,最后清中断,中断处理程序返回。

内核态 task 的处理方法:通过检查 task 的 nested_count 判断是用户态还是内核态,如果是内核态,直接清中断,中断处理程序返回。

(3) blocking sleep 的含义, task 调用 blocking sleep 时做什么处理? 什么时候唤醒 sleep 的 task?

Blocking sleeep 的含义为: tasks 主动交出 CPU,开始 sleep。

Task 调用 blocking sleep 时做的处理: 关中断,设置自己的 deadline,即什么时候醒来,然后把自己的状态设为 SLEEPING,把自己放入 sleep_wait_queue 队列,调用 scheduler_entry 执行就绪队列里的下一个 task。

唤醒 sleep task 的时间: 当前时间超过 deadline 时,唤醒 task,把它放入就绪队列。

(4)设计或实现过程中遇到的问题和得到的经验(如果有的话可以写下来,不是必需项)

设计和实现过程中全是问题。

在实现 check_sleeping 函数时也出现了一些错误,因为忽略了 sleep_wait_queue 队头元素是没有存 pcb 的,队头元素的

2. 基于优先级的调度器设计

(1) priority-based scheduler 的设计思路,包括在你实现的调度策略中优先级是怎么定义的,如何给 task 赋予优先级,调度与测试用例如何体现优先级的差别

调度策略中的优先级定义为pid×30, 所有 tasks 的 pid 号依次为 1,2,3... task 的 priority 越大,优先级越高。

每次进入 scheduler 函数,遍历就绪队列,找出 priority 最大的 task,赋给 current_running,然后将这个 task 的 priority 减 5, 若 task 的 priority 减为 0 了,并不马上重新赋上pid×30,而是等所有的 task 的 priority 都减为 0,再把 tasks 的 priority 赋为pid×30。这样不同 task 占用 CPU 的次数比为 1:2:3:...

通过上板时不同 tasks 执行次数的不同来体现优先级的差别,执行次数越多,优先级越

高。

(2)设计或实现过程中遇到的问题和得到的经验(如果有的话可以写下来,不是必需项)

基于优先级的调度器实现并没有遇到太大问题,因为都是 C 语言实现,只要想清楚了很好调试。

3. 关键函数功能

请列出上述各项功能设计里, 你觉得关键的函数或代码块, 及其作用

```
NESTED(handle_int,0,sp)
  /* TODO: timer_irq */
/* read int IP and handle clock interrupt or just call do_nothing */
  SAVE_CONTEXT(USER)
  /* test IP bit *
  mfc0 k0, CP0_CAUSE
  andi
        k0, k0, CAUSE_IPL
  andi k0, k0, 0x8000
         k0, zero, l1 //it is not clock interrupt, jump
  nop
  mtc0 zero, CP0_COUNT
  li 
         k1, 150000000
  mtc0 k1, CP0_COMPARE
  jal time_irq
  nop
11:
 mfc0 k0, CP0_CAUSE
lui t0, 0xffff
ori t0, t0, 0x00ff
and k0, k0, t0
  mtc0 k0, CP0_CAUSE
  nop
  RESTORE_CONTEXT(USER)
  STI
  eret
  /* TODO:end */
END(handle int)
```

中断处理函数

```
rvoid time_irq(){
    static int i = 1;
    ++time_elapsed;
    if(current_running->nested_count == 0){
        enter_critical();
        current_running->nested_count++;
        current_running->status = READY;
        enqueue(&ready_queue, (node_t *)current_running);
        current_running->nested_count--;
        scheduler_entry();
    }
}
```

时钟中断处理函数

```
void do_sleep(int milliseconds){
    ASSERT(!disable_count);

    enter_critical();
    // TODO
    current_running->deadline = time_elapsed*1000 + milliseconds;
    current_running->status = SLEEPING;
    enqueue(&sleep_wait_queue, (node_t *)current_running);
    scheduler_entry();
}
```

Do_sleep 函数

```
void check_sleeping(){
    uint64_t current_time = get_timer();
    node_t *pp;
    pcb_t *temp;

for(pp = peek(&sleep_wait_queue); pp != &sleep_wait_queue && pp != NULL;){
    temp = (pcb_t*)pp;
    if(current_time * 1000 >= temp->deadline){
        temp->status = READY;
        pp = pp->next;
        temp->node.prev->next = temp->node.next;
        temp->node.next->prev = temp->node.prev;
        enqueue(&ready_queue, (node_t *)temp);
    }
    else
        pp = pp->next;
}
```

check sleeping 函数

```
for(pp=peek(&ready_queue); pp!=&ready_queue && pp!=NULL; pp=pp->next){
    if(temp->priority < ((pcb_t*)pp)->priority){
         temp = (pcb_t*)pp;
temp->node.prev->next = temp->node.next;
temp->node.next->prev = temp->node.prev;
temp->node.next =
temp->node.prev = |
current running = temp;
if(current_running->priority>0)
        current_running->priority-=5;
else{
        for(pp=peek(&ready_queue);pp!=&ready_queue&&pp!=NULL;pp=pp->next){
    if(((pcb_t*)pp)->priority!=0)
        prio=1;
        }
if(prio==0){
        current_running->priority=(current_running->pid)*30;
for(pp=peek(&ready_queue);pp!=&ready_queue&&pp!=NULL;pp=pp->next)
                  ((pcb_t*)pp)->priority=30*((pcb_t*)pp)->pid;
```

基于优先级的调度函数

```
.macro SAVE_CONTEXT offset
 * TODO: need add
  la
           k0, current_running
           k0, 0(k0)
  1w
  addi
           k0, k0, \offset
           zero, 0(k0)
  SW
           AT, 4(k0)
  SW
           v0, 8(k0)
  SW
           v1, 12(k0)
           a0, 16(k0)
           a1, 20(k0)
  SW
           a2, 24(k0)
  SW
           a3, 28(k0)
  SW
           t0, 32(k0)
  SW
           t1, 36(k0)
  SW
           t2, 40(k0)
  SW
           t3, 44(k0)
  SW
           t4, 48(k0)
  SW
           t5, 52(k0)
  SW
           t6, 56(k0)
  SW
           t7, 60(k0)
s0, 64(k0)
  SW
  SW
           s1, 68(k0)
  SW
           s2, 72(k0)
s3, 76(k0)
s4, 80(k0)
  SW
  SW
  SW
           s5, 84(k0)
  SW
           s6, 88(k0)
```

```
s7, 92(k0)
SW
         t8, 96(k0)
SW
         t9, 100(k0)
SW
         gp, 112(k0)
SW
         sp, 112(k0)
sp, 116(k0)
fp, 120(k0)
ra, 124(k0)
SW
SW
SW
mfc0
         k1, CP0_STATUS
         k1, 128(k0)
SW
mfhi
         t1
         t1, 132(k0)
SW
mflo
         t1
         t1, 136(k0)
SW
mfc0
         k1, CP0_BADVADDR
         k1, 140(k0)
SW
         k1, CP0_CAUSE
mfc0
         k1, 144(k0)
SW
mfc0
         k1, CP0_EPC
         k1, 148(k0)
SW
```

SAVE_CONTEXT

```
.macro RESTORE_CONTEXT offset
  TODO: need add */
  1a
             k0, current_running
  1w
             k0, 0(k0)
  addi
             k0, k0, \offset
             zero, 0(k0)
  lw
             AT, 4(k\hat{\theta})
  lw
            AT, 4(k0)

v0, 8(k0)

v1, 12(k0)

a0, 16(k0)

a1, 20(k0)

a2, 24(k0)

a3, 28(k0)

t0, 32(k0)
  lw
  1w
  lw
  lw
  1w
  lw
  lw
             t1, 36(k0)
  lw
             t2, 40(k0)
  lw
             t3, 44(k0)
  lw
             t4, 48(k0)
  lw
             t5, 52(k0)
  lw
  lw
             t6, 56(k0)
  1w
             t7, 60(k0)
             s0, 64(k0)
  lw
             s1, 68(k0)
  lw
  1w
             s2, 72(k0)
             s3, 76(k0)
  1w
             s4, 80(k0)
  lw
             s5, 84(k0)
  lw
             s6, 88(k0)
s7, 92(k0)
t8, 96(k0)
  lw
  lw
  lw
  lw
             t9, 100(k0)
```

```
t9, 100(k0)
gp, 112(k0)
  1w
  1w
               sp, 116(k0)
fp, 120(k0)
ra, 124(k0)
  lw
  1w
  1w
               k1, 128(k0)
  lw
               k1, CP0_STATUS
  mtc0
               k1, 132(k0)
  1w
  mthi
               k1
  1w
               k1, 136(k0)
  mtlo
               k1
               k1, 140(k0)
k1, CP0_BADVADDR
k1, 144(k0)
k1, CP0_CAUSE
k1, 148(k0)
k1, CP0_EPC
  1w
  mtc0
  lw
  mtc0
  lw
  mtc0
/* TODO: end */
```

RESTORE_CONTEXT

参考文献

无