

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

中断及时钟中断的处理.

时钟中断 `handle-int()`
(entry.S)

SAVE_CONTEXT (USER)

↓

读中断的 IP bit 位

时钟中断

不是

清中断, 返回原进程.

非时钟中断

处理函数 (timer_irq)

↓

`enter-critical()`

↓

增加计数器 `time-elapsed` 用于 blocking sleep.

↓

检查 `nested-count` (中入内核再处理)

↓

把 `current-running` 加入 `ready-queue`↓ `save-pcb(kernel)``Scheduler-entry()`
(entry.S)

↓

CLZ 指令.

cause 寄存器 IP 域不

为 0 的最高位清零

(IP bit)

clock

开中断

恢复上下文 `RESTORE-CONTEXT`

直接返回, 跳

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
beq 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
l1:
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

```

void 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 = NULL;
temp->node.prev = NULL;
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
lw      k0, 0(k0)

addi    k0, k0, \offset
sw      zero, 0(k0)
sw      AT, 4(k0)
sw      v0, 8(k0)
sw      v1, 12(k0)
sw      a0, 16(k0)
sw      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)
sw      s0, 64(k0)
sw      s1, 68(k0)
sw      s2, 72(k0)
sw      s3, 76(k0)
sw      s4, 80(k0)
sw      s5, 84(k0)
sw      s6, 88(k0)

```

```

sw      s7, 92(k0)
sw      t8, 96(k0)
sw      t9, 100(k0)
sw      gp, 112(k0)
sw      sp, 116(k0)
sw      fp, 120(k0)
sw      ra, 124(k0)

mfc0    k1, CP0_STATUS
sw      k1, 128(k0)
mfhi    t1
sw      t1, 132(k0)
mflo    t1
sw      t1, 136(k0)
mfc0    k1, CP0_BADVADDR
sw      k1, 140(k0)
mfc0    k1, CP0_CAUSE
sw      k1, 144(k0)
mfc0    k1, CP0_EPC
sw      k1, 148(k0)

```

SAVE_CONTEXT

```

.macro RESTORE_CONTEXT offset
/* TODO: need add */
la      k0, current_running
lw      k0, 0(k0)

addi    k0, k0, \offset
lw      zero, 0(k0)
lw      AT, 4(k0)
lw      v0, 8(k0)
lw      v1, 12(k0)
lw      a0, 16(k0)
lw      a1, 20(k0)
lw      a2, 24(k0)
lw      a3, 28(k0)
lw      t0, 32(k0)
lw      t1, 36(k0)
lw      t2, 40(k0)
lw      t3, 44(k0)
lw      t4, 48(k0)
lw      t5, 52(k0)
lw      t6, 56(k0)
lw      t7, 60(k0)
lw      s0, 64(k0)
lw      s1, 68(k0)
lw      s2, 72(k0)
lw      s3, 76(k0)
lw      s4, 80(k0)
lw      s5, 84(k0)
lw      s6, 88(k0)
lw      s7, 92(k0)
lw      t8, 96(k0)
lw      t9, 100(k0)

```

```
lw    t9, 100(k0)
lw    gp, 112(k0)
lw    sp, 116(k0)
lw    fp, 120(k0)
lw    ra, 124(k0)

lw    k1, 128(k0)
mtc0  k1, CP0_STATUS
lw    k1, 132(k0)
mthi  k1
lw    k1, 136(k0)
mtlo  k1
lw    k1, 140(k0)
mtc0  k1, CP0_BADVADDR
lw    k1, 144(k0)
mtc0  k1, CP0_CAUSE
lw    k1, 148(k0)
mtc0  k1, CP0_EPC

/* TODO: end */
```

RESTORE_CONTEXT

参考文献

无

■