Project 5 Device Driver 设计文档

中国科学院大学
秦宏
2019.1.24

1. 网卡驱动

（1） 任务一初始化了 64 个接收描述符，每个接收描述符中都设置了 RDES0，RDES1，RDES2，RDES3 的值，其中 RDES0 的值为 0x00000000，当 RDES0 的值为 0x80000000 时代表 DMA 寄存器的值有效。RDES1 的值包含了数据接收 buffer 的大小以及描述符的位置，前 63 个描述符第 24 位置 1 表示表示 RDES3 存储的是下一个描述符地址，第 25 位置 1 表示该描述符为最后一个。RDES2 存储的是数据接收 buffer1 的物理地址，要求减去 0xa0000000。RDES3 存储下一个描述符的物理地址，最后一个存储的是第一个描述符地址。

（2） 任务一中的发送描述符同接收描述符一样，设置了 64 个 RDES0 到 RDES3 的寄存器的值，同样是采用环形链表结构。基本内容同接收描述符内容一致，但是发送描述符的 RDES1 中 29 和 30 位要设置为 1，目的是表示 buffer1 中的数据为完整的一帧数据。

（3） 任务二中判断是否有数据包到达网卡时是在时钟中断时判断的，

```
    if(!queue_is_empty(&recv_block_queue)){
    pcb_t *item = recv_block_queue.head;
    Recv_desc = (uint32_t*)(mac_ptr->rd + (ch_flag) * 16);
    if (((*Recv_desc) & 0x80000000) != 0x80000000)
    {
        ch_flag++;
        item->status = TASK_READY;
        pcb_t *temp = queue_remove(&recv_block_queue, item);
        queue_push(&ready_queue, item);
    }
}
```

如果最后一个描述符接收到数据，则将阻塞队列释放进入就绪队列。

（4） 未完成任务三。

（5） 环形链表的描述符实际上没有实际的第一个描述符，描述符的地址连续使得描述符在被使用后可以被再次调用。

2. Bonus 设计

未完成 bonus。

3. 关键函数功能

发送初始化函数

```
static void send_desc_init(mac_t *mac)
{
    //send memory init
    int i = 0;
    uint32_t *buffer_temp = (uint32_t*)SEND_BUFFER;
    for(i = 0; i < mac->psize; i++){
        buffer_temp[i] = buffer[i];
    }
    // printf("test1\n");
    //send desc init
    uint32_t *desc_addr = (uint32_t*)SEND_DESC;
    for(i = 0; i < mac->pnum-1; i++){
        desc_addr[0] = 0x00000000;
        desc_addr[1] = 0x61000400;
        desc_addr[2] = SEND_BUFFER - 0xa0000000;
        desc_addr[3] = (uint32_t)(desc_addr + 4) - 0xa0000000;
        desc_addr = desc_addr + 4;
    }
    // printk("test2\n");
    desc_addr[0] = 0x00000000;
    desc_addr[1] = 0x63000400;
    desc_addr[2] = SEND_BUFFER - 0xa0000000;
    desc_addr[3] = SEND_DESC - 0xa0000000;

    mac->td     = SEND_DESC;
    mac->td_phy = SEND_DESC - 0xa0000000;
}
```

检查是否完成 64 个包传输

```
void check_recv(mac_t *test_mac)
{
    int i = 0;
    while(ch_flag <= 64){
        sys_move_cursor(1,8);
        printf("package number:%d\n",ch_flag);
        for(i = 0; i < 64; i++){
            printf("%x  ",*((uint32_t*)RECV_BUFFER + i +
(ch_flag-1)*0x400/4));
        }
        if(ch_flag != 64)
            sys_wait_recv_package(test_mac);
        else
            break;
    }
```

```
    printf("\n64 packages finished\n");
}
```

参考文献

[1] 无