一、上半部与下半部

起源：

1. 中断处理程序执行时间过长引起的问题
2. 有些设备的中断处理程序必须要处理一些耗时操作

二、下半部机制之tasklet ---- 基于软中断

6.1 结构体

```
struct tasklet_struct
{
 struct tasklet_struct *next;
 unsigned long state;
 atomic_t count;
 void (*func)(unsigned long);
 unsigned long data;
};
```

6.2 定义tasklet的中断底半部处理函数

```
void tasklet_func(unsigned long data);
```

6.3 初始化tasklet

```
DECLARE_TASKLET(name, func, data);
/*
定义变量并初始化
参数: name:中断底半部tasklet的名称
      Func:中断底半部处理函数的名字
      data:给中断底半部处理函数传递的参数
*/
```

```
void tasklet_init(struct tasklet_struct *t,void (*func)(unsigned long), unsigned
long data)
```

6.4 调度tasklet

```
void tasklet_schedule(struct tasklet_struct *t)
//参数:t:tasklet的结构体
```

三、按键驱动之tasklet版

四、下半部机制之workqueue ----- 基于内核线程

8.1 工作队列结构体：

```
typedef void (*work_func_t)(struct work_struct *work)

struct work_struct {
 atomic_long_t data;
 struct list_head entry;
 work_func_t func;
#ifdef CONFIG_LOCKDEP
 struct lockdep_map lockdep_map;
#endif
};
```

8.2 定义工作队列底半部处理函数

```
void work_queue_func(struct work_struct *work);
```

8.3 初始化工作队列

```
struct work_struct work_queue;
```

初始化：绑定工作队列及工作队列的底半部处理函数

```
INIT_WORK(struct work_struct * pwork, _func) ;
```

参数：pwork：工作队列

func：工作队列的底半部处理函数

8.4 工作队列的调度函数

```
bool schedule_work(struct work_struct *work);
```

五、按键驱动之workqueue版

六、下半部机制比较

任务机制

workqueue ----- 内核线程 能睡眠 运行时间无限制

异常机制 -------- 不能睡眠 下半部执行时间不宜太长 （ < 1s)

软中断 ---- 接口不方便

tasklet ----- 无具体延后时间要求时

定时器 -----有具体延后时间要求时