Lessons for some daunting subjects, simplified for a smooth learning journey.

LINUX KERNEL

- LINUX KERNEL BOOTING SEQUENCE (LINUX-KERNEL-BOOTING-SEQUENCE.PHP?COURSE=LINUX KERNEL)
- PROCESS MANAGEMENT PART 1 (PROCESS-MANAGEMENT-PART-1.PHP?COURSE=LINUX KERNEL)
- PROCESS MANAGEMENT PART 2 (PROCESS-MANAGEMENT-PART-2.PHP?COURSE=LINUX KERNEL)
- PROCESS SCHEDULING (PROCESS-SCHEDULING.PHP?COURSE=LINUX KERNEL)
- SYSCALLS (SYSCALLS.PHP?COURSE=LINUX KERNEL)
- BREAKPOINTS -STEP BY STEP (BREAKPOINTS-STEP-BY-STEP.PHP?COURSE=LINUX KERNEL)
- INTERRUPTS (INTERRUPTS.PHP?COURSE=LINUX KERNEL)
- HOW BREAKPOINTS ARE IMPLEMENTED (HOW_BREAKPOINTS_ARE_IMPLEMENTED.PHP?COURSE=LINUX KERNEL)
- TOP HALVES AND BOTTOM HALVES IN INTERRUPT (TOP_HALVES_AND_BOTTOM_HALVES_IN_INTERRUPT_HANDLING.PHP?COURSE=LINUX KERNEL)
- **SOFTIRQ AND TASKLETS (SOFTIRQ-AND-TASKLETS.PHP?COURSE=LINUX KERNEL)**
- LINUX MEMORY MANAGEMENT - PAGE ALLOCATION (LINUX-MEMORY-MANAGEMENT-PAGE-ALLOCATION.PHP?COURSE=LINUX KERNEL)
- LINUX MEMORY MANAGEMENT- KMALLOC AND VMALLOC (LINUX-MEMORY-MANAGEMENT-KMALLOC-AND-VMALLOC.PHP?COURSE=LINUX KERNEL)
- LINUX INTERRUPT HANDLING-WORK QUEUES (LINUX-INTERRUPT-HANDLING-WORK-QUEUES.PHP?COURSE=LINUX KERNEL)
- UNDERSTANDING KERNEL SYNCHRONIZATION (UNDERSTANDING-KERNEL-

SYNCHRONIZATION.PHP?COURSE=LINUX KERNEL)

LINUX MEMORY MANAGEMENT - BUDDY SYSTEM ALGORITHM AND SLAB ALLOCATOR (LINUX-MEMORY-MANAGEMENT-BUDDY-SYSTEM-ALGORITHM-AND-SLAB-ALLOCATOR.PHP?COURSE=LINUX KERNEL)

Softirqs and Tasklets

In last chapter we have discussed about interrupts (http://www.tutorialsdaddy.com/2015/09/interrupts/) and interrupt handling , bottom halves and top halves (http://www.tutorialsdaddy.com/2015/09/top_halves_and_bottom_halves_in_interrupt_handling/).In this chapter we will discuss different types of bottom halves available in Latest linux kernel.

As bottom halves perform the task which was deferred by top halves and during bottom halves all the interrupt lines are enabled. To perform bottom halves task latest kernel provide following bottom halves techniques.

1) Softirq
2) Tasklets
3) Work Queues

Lets discuss each of these types of bottom halves in details.

1.Softirq

a) Softirqs are a set of statically defined bottom halves which can run simultaneously on any of the processor.
b) Even two of the same type can run on any processor concurrently.
c) Because two softirqs can run concurrently , It is the responsibility of programmer to take care of synchronization between softirqs .
d) A softirq never preempts other softirq.
e) Only interrupt handler can preempt softirq.
f) Softirq runs with interrupt enabled and can't sleep as it runs in interrupt context.

Implementation

Softirq statically created at compile time.

It is represented by softirq_action structure defined in

```
struct softirq_action{
void (*action) (struct softirq_action *);
}
```

A 32 bit entry of this structure is defined in kernel/softirq.c

Kernel enforce a limit of 32 registered softirq in the latest versions.

Softirq Handler

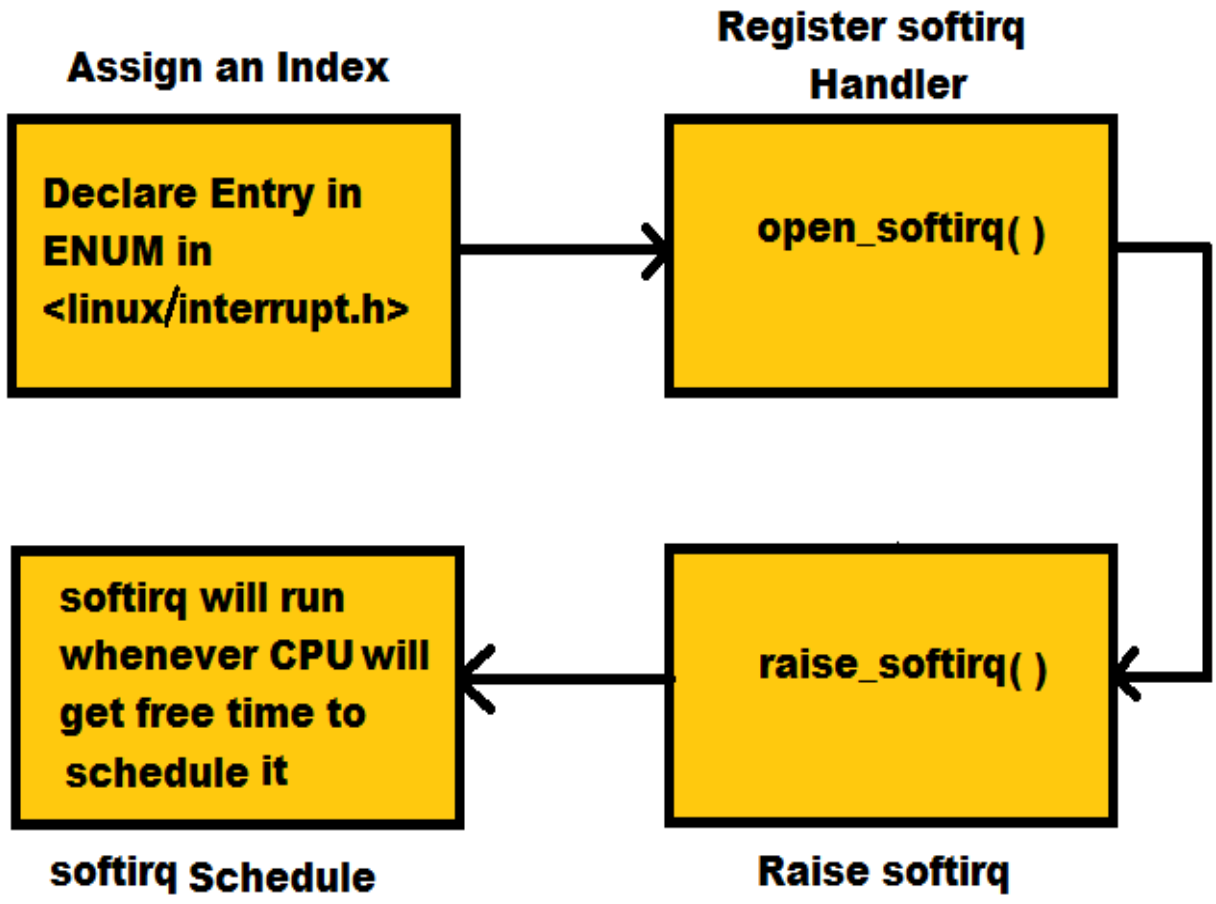
Softirq handler defines a function which will run when kernel schedule the softirq to run. below is the prototype of softirq handler.

```
void softirq_handler(struct softirq_action *);
```

action is the function which will actually run on scheduling of softirqs and perform all task of bottom halves.

Executing softirq

An interrupt handler or we can say top halves marks its softirq before returning. When CPU will get time , softirq will get chance to execute.



(http://www.tutorialsdaddy.com/wp-content/uploads/2015/10/asas.png)

Declaring a Softirq

Softirq are statically declared at compile time via an enum in . This enum index is used by kernel as a relative priority. This index starts from zero. softirq with the index 0 has highest priority . For creating a softirq we need to add our entry to this enum.

Registering softirq
Softirq handler is registered at run time by open_softirq() , which takes two arguments:

softirq index and its handler function.

Raising Softirq
After a handler is added to the enum list and registered via open_softirq() , it is ready to run .softirq are raised by using raise_softirq() call. after raising softirq will get chance to execute whenever system is idle.

Real usage of softirq
Softirq are used in time critical and important bottom halves processing. We use softirq in mostly networking devices like Ethernet , NIC and block devices like USB, Hard disk etc.

2.Tasklets

a) Tasklets is another bottom-half mechanism which is built on top of softirq.
b) Same tasklets cannot run on more than one processor at same time.
c) So in tasklets we need not to worry about synchronization among them.
d) Tasklets has very simple interface as compared to softirq.
e) Tasklets cannot sleep as it runs in interrupt context.

Tasklets Implementation

As already discussed that tasklets are implemented on top of softirq. Tasklets are represented by two softirqs : **Hi_SOFTIRQ** and **TASKLET_SOFTIRQ**.

HI_SOFTIRQ has high priority than TASKLET_SOFTIRQ.

Taslklets are defined by tasklet_struct structure which is declared in

```
struct tasklet_struct {
        struct tasklet_struct *next;  /* next tasklet in the list */
        unsigned long state;          /* state of the tasklet */
        atomic_t count;               /* reference counter */
        void (*func)(unsigned long);  /* tasklet handler function */
        unsigned long data;           /* argument to the tasklet function */
};
```

(http://www.tutorialsdaddy.com/wp-content/uploads/2015/10/tasklets.png)

Reference :- Linux Kernel Development by Robert Love

Scheduling Tasklets

Tasklets are scheduled via the tasklets_schedule() and tasklets_hi_schedule() functions which receives pointer to tasklet_struct structure as an argument.

Declaring your tasklets
Tasklets can be created dynamically and statically depending upon requirement.

Statically Creation
DECLARE_TASKLET(name,func,data)

Dynamically Creation
tasklet_init(t,tasklet_handler,dev);

Writing Tasklet Handler
Tasklet handler is the function which runs upon scheduling of tasklets. below is the prototype for tasklet handler.

void tasklet_handler(unsigned long data)

Scheduling Tasklet
Tasklets are scheduled tasklet_schedule() which takes pointer to tasklet_struct.
After tasklet is scheduled , it runs once at some time in the near future.

Real usage of Tasklets
Tasklets are the most widely used bottom-half. Its a most preferred way for implementing bottom half for a normal hardware .

That's all was about tasklets and softirqs.We will discuss work queues in next chapter.

Search here... fl

0 (/#facebook) 0 (/#linkedin) G+1 0 (/#twitter)

Subscribe here

Name

Email Id

Subscribe

Quick Links

Contact Us (contactus.php?page_id=54)

About Us (aboutus.php?page_id=55)

Contact Us

- info@tutorialsdaddy.com
- +91- 8860992443

Follow Us

(http://www.facebook.com/tutorialsdaddy/)

(https://twitter.com/tutorialsdaddy)

(https://www.youtube.com/channel/UCCSJG3ecVfqG38FbQOP_HNA)

We can be supported here!