Linux Interrupts: The Basic Concepts

Paula Sandoval

Abril 2025

1. Resumen

Una interrupción es un evento **asíncrono** generado por hardware que detiene el flujo normal de la CPU, comunmente genereado por dispositivos de entrada y salida.
Una excepción es un evento **síncrono** que ocurre cuando la CPU detecta un error (división por cero, fallo de página, etc.).
Las interrupciones son importantes ya que se ejecutan en tiempo real, mejoran la eficiencia de procesos multitarea y saltan el polling que se organiza los demas procesos.

2. Tipos de interrupciones y excepciones

- **Maskables:** se pueden bloquear temporalmente.
- **No maskables (NMI):** no se pueden bloquear, se usan en errores críticos.
- **Excepciones:** fallos de página, segmentación, etc.

Linux asigna vectores:

- 0–31: excepciones
- 32–47: interrupciones de hardware (IRQs)
- 128: interrupciones de software

3. Exception Handlers

:
Son rutinas que manejan excepciones, que guardan el estado del CPU, ejecutan código de manejo y hace el envío de señales como `SIGSEGV`, `SIGFPE`.

4. Interrupciones de software

Se generan por instrucciones int y son son usadas para:

- Llamadas al sistema (syscalls).
- Simulaciones y pruebas.

5. Tabla de señales y excepciones comunes

Vector	Excepción	Señal
0	Divide Error	SIGFPE
3	Breakpoint	SIGTRAP
6	Invalid Opcode	SIGILL
14	Page Fault	SIGSEGV

6. Estructuras de datos clave en Linux

- `irq_desc_t`: gestiona cada IRQ.
- `irqaction`: lista de manejadores registrados.
- `hw_interrupt_type`: funciones del controlador.

7. Proceso de manejo de interrupción

1. CPU detecta interrupción.
2. Accede al IDT.
3. Cambia al modo kernel.
4. Ejecuta manejador.
5. Procesa `softirqs` si los hay.
6. Vuelve con `iret`.

8. Softirqs, Bottom Halves y Tasklets

Permiten diferir el trabajo pesado:

Tipo	Multiprocesamiento	Reentrante
Bottom half	No	No
Softirq	Sí	Sí
Tasklet	Sí	No

9. IDT: Interrupt Descriptor Table

Tabla de hasta 256 entradas. Cada entrada contiene el handler para una interrupción o excepción específica.

10. IRQ (interrupt request)Sharing y manejo dinámico

- Soporta IRQs compartidas.
- Múltiples dispositivos pueden usar la misma IRQ.
- Cada manejador verifica si es su interrupción.

11. SMP y APIC

En sistemas multiprocesador:

- Se usan APICs (Advanced PIC).
- Permiten distribuir interrupciones entre CPUs.
- Usan IPI (Inter-Processor Interrupts) para comunicación interna.

12. Conclusión

Linux tiene un sistema robusto y escalable para el manejo de interrupciones. Soporta hardware moderno, interrupciones compartidas, y ejecución diferida mediante softirqs/tasklets, con estructuras optimizadas para rendimiento y concurrencia.

13. Preguntas:

13.1. ¿Qué es una interrupción?, ¿por qué son necesarias?, ¿qué tipos existen y cuáles son sus características?

Una **interrupción** es un evento asíncrono externo al procesador, generado por dispositivo de entrada o salida que provoca suspender temporalmente la ejecución del proceso ejecutado para atender la ocurrencia.
las interrupciones permiten abordar los dispositivos externos de manera eficiente, sin comprobar continuamente su estado (sondeo), lo que ahorra tiempo de CPU y, por lo tanto, aumenta el rendimiento general del sistema.
Tipos de interrupciones:

- **Interrupciones de hardware:** Generadas por externos (teclados, discos duros, etc.).

- **Interrupciones de software:** Generadas por instrucciones del programa (por ejemplo, llamadas al sistema).
- **Interrupciones no enmascarables (NMI):** Usualmente no pueden ser bloqueadas y se utilizan en fallos mayores, como por ejemplo, fallos de hardware.

Características:

- Deben ser manejadas mediante tablas (IDT: Interrupt Descriptor Table).
- Se puede configurar su prioridad.
- Pueden estar anidadas
- Pueden ser ejecutdas en tiempos diferidos (softirqs, tasklets) para mejorar la respuesta del sistema.

13.2. ¿Qué son los .exception handlers¿

Es una funcion del sistema que esta a cargo de manejar y gestionar los eventos sincrinizados por la CPU.
Principales características:

- Recolectan la información de anómalias del sistema detectadas durante la ejeción de instrucciones.
- Manejan errores como división por cero, violaciones de segmentación o fallos de página.
- Generalmente envían señales a los procesos afectados (`SIGFPE`, `SIGSEGV`, etc.).
- Algunos handlers intentan recuperar la ejecución, mientras que otros terminan el proceso fallido.

Los exception handlers también pueden invocar mecanismos de bajo nivel como la recuperación de memoria (page fault recovery) en sistemas operativos modernos.

13.3. ¿Qué son las interrupciones generadas por software?, ¿para qué sirven?, ¿qué algoritmos/diagramas se utilizan?

Interrupciones generadas por software son interrupciones provocadas explícitamente por instrucciones en el código del programa, como el uso de `INT n` en x86.
Utilidad:

- Permiten acceder a servicios del sistema operativo sin contacto directo con hardware.
- Son el mecanismo base para realizar **llamadas al sistema** en Linux (por ejemplo, `int 0x80` en arquitecturas x86).
- Facilitan la comunicación entre espacio de usuario y espacio de kernel.

Flujos y algoritmos:

- `do_IRQ()`: Función encargada de delegar el procesamiento a los controladores de interrupciones apropiados.
- `ret_from_intr()`: Restablece el contexto después de atender una interrupción.
- `softirqs` y `tasklets`: Mecanismos que permiten diferir trabajo no urgente a un contexto menos crítico.

En sistemas SMP, el flujo se complica mediante la coordinación de múltiples CPUs usando controladores APICs.

13.4. ¿Qué son las IRQ y las estructuras de datos relacionadas?

IRQ (Interrupt Request) es una línea física o virtual mediante la cual un dispositivo solicita atención al CPU.
Estructuras asociadas en Linux:

- `irq_desc_t`: Contiene la información de gestión de una IRQ (estado, controladores asociados, locks de sincronización).
- `irqaction`: Lista encadenada de handlers asociados a un número de IRQ.
- `hw_interrupt_type`: Agrupa funciones específicas para cada tipo de controlador de hardware (start, enable, ack, end).

Estas estructuras permiten:

- Gestionar múltiples handlers por IRQ (interrupciones compartidas).
- Aplicar técnicas de enmascaramiento dinámico de interrupciones.
- Optimizar el despacho de interrupciones en sistemas multiprocesador.