- **INTERRUPCIONES Y MANEJADORES DE INTERRUPCIONES**

Periódicamente, el núcleo puede comprobar el estado del hardware en el sistema y responder en consecuencia. Sin embargo, el polling incurre en una sobrecarga, ya que debe producirse repetidamente independientemente de si el hardware está activo o listo. Una mejor solución es proporcionar un mecanismo para que el hardware señale al núcleo cuando necesita atención. Este mecanismo se llama interrupción.

- **INTERRUPCIONES**

Las interrupciones permiten al hardware enviar una señal al procesador. El procesador recibe la interrupción y envía una señal al sistema operativo para que éste pueda responder a los nuevos datos. El kernel puede interrumpirse en cualquier momento para procesar las interrupciones.

Al recibir una interrupción, el controlador de interrupciones envía una señal al procesador. El procesador detecta esta señal e interrumpe su ejecución actual para gestionar la interrupción. El procesador puede entonces notificar al sistema operativo que se ha producido una interrupción, y el sistema operativo puede manejar la interrupción adecuadamente.

- **MANEJADORES DE INTERRUPCIONES**

La función que el kernel ejecuta en respuesta a una interrupción específica se llama rutina de servicio de interrupción (ISR). Cada dispositivo que genera interrupciones tiene un manejador de interrupciones asociado.

Lo que diferencia a los manejadores de interrupción de otras funciones del kernel es que el kernel los invoca en respuesta a las interrupciones y que se ejecutan en un contexto especial llamado contexto de interrupción.

Lo que diferencia a los manejadores de interrupción de otras funciones del núcleo es que el núcleo los invoca en respuesta a las interrupciones y que se ejecutan en un contexto especial llamado contexto de interrupción.

- **TOP HALVES VS BOTTOM HALVES**

El manejador de la interrupción es top halve. Top halve se ejecuta inmediatamente después de la recepción de la interrupción y realiza sólo el trabajo que es crítico para el tiempo. El trabajo que se puede realizar más tarde se aplaza a bottom halve. Esto se ejecuta en el futuro, con todas las interrupciones habilitadas.

- **REGISTRAR MANEJADOR DE INTERRUPCIONES**

Cada dispositivo tiene un controlador asociado y, si ese dispositivo utiliza interrupciones, entonces ese controlador debe registrar un manejador de interrupciones.

int request_irq(unsigned int irq, irqreturn_t (*handler)(int, void *, struct pt_regs *), unsigned long irqflags, const char *devname, void *dev_id)

- **LIBERAR MANEJADOR DE INTERRUPCIONES**

Cuando un controlador se descargue, necesitará liberar su manejador de interrupciones y potencialmente deshabilitar la línea de interrupción.
void free_irq(unsigned int irq, void *dev)

- **IMPLEMENTAR MANEJADORES DE INTERRUPCIONES**

Depende de la arquitectura, del procesador, del tipo de controlador de interrupción utilizado y del diseño de la arquitectura y la máquina.

Un dispositivo emite una interrupción enviando una señal eléctrica a través de su bus al controlador de interrupciones. Si la línea de interrupción está habilitada, el controlador de interrupción envía la interrupción al procesador. A menos que las interrupciones estén desactivadas en el procesador, éste deja inmediatamente de hacer lo que está haciendo, desactiva el sistema de interrupciones y salta a un punto predefinido de la memoria y ejecuta el código que allí se encuentra. Este punto predefinido lo establece el kernel y es el punto de entrada para los gestores de las interrupciones.

Para cada línea de interrupción, el procesador salta a una ubicación única en la memoria y ejecuta el código ubicado allí. De esta manera, el kernel conoce el número IRQ de la interrupción entrante. El punto de entrada inicial simplemente guarda este valor y almacena los valores del registro actual en la pila; entonces el núcleo llama a do_IRQ().

/proc/interrupciones

Procfs es un sistema de archivos virtual que existe sólo en la memoria del kernel y normalmente se monta en /proc. La lectura o escritura de archivos en procfs invoca funciones del kernel que simulan la lectura o escritura de un archivo real.

BOTTOM HALVES

Su trabajo es realizar cualquier trabajo relacionado con la interrupción que no sea realizado por el manejador de la interrupción. En un mundo ideal, este es casi todo el trabajo porque quieres que el manejador de interrupciones realice el menor trabajo posible.

Sin embargo, el manejador de la interrupción debe realizar parte del trabajo. Por ejemplo, el manejador de la interrupción casi seguramente necesita saber que ha recibido la interrupción al hardware.

Casi todo lo demás es susceptible de ser realizado en la mitad inferior (b ottom half). Desgraciadamente, no existen reglas estrictas sobre qué trabajo realizar en cada lugar - la decisión se deja enteramente en manos del autor del controlador del dispositivo. Aunque ninguna disposición es ilegal, una disposición puede ser ciertamente subóptima.

Los manejadores de interrupción se ejecutan de forma asíncrona, con al menos la línea de interrupción actual deshabilitada. Es importante minimizar su duración. Aunque no siempre está claro cómo dividir el trabajo entre la mitad superior e inferior, un par de consejos útiles ayudan:

- Si el trabajo es sensible al tiempo, realizar en el manejador de interrupciones.

- Si el trabajo está relacionado con el hardware, realizar en el manejador de interrupciones.

- Si el trabajo necesita asegurar que otra interrupción no lo interrumpa, realizar en el manejador de interrupciones.

- Para todo lo demás, considera realizar el trabajo en la mitad inferior.

- **POR QUÉ BOTTOM HALVES?**

Se quiere limitar la cantidad de trabajo que realiza en un manejador de interrupción. Minimizar el tiempo que se pasa con las interrupciones deshabilitadas es importante para la respuesta y el rendimiento del sistema. La solución es aplazar parte del trabajo hasta más tarde.

El objetivo de una mitad inferior no es hacer el trabajo en un punto específico en el futuro, sino simplemente aplazar el trabajo hasta cualquier punto en el futuro cuando el sistema está menos ocupado y las interrupciones están de nuevo habilitadas. A menudo, las mitades inferiores se ejecutan inmediatamente después de que regrese la interrupción. La clave es que se ejecutan con todas las interrupciones habilitadas.

- **TASKLETS**

Los tasklets son un mecanismo bottom-half construido sobre los softirqs. No tienen nada que ver con las tareas.

Los softirqs son necesarios sólo para usos de alta frecuencia y altamente roscados. Los tasklets, en cambio, tienen un uso mucho mayor. Los tasklets funcionan bien para la gran mayoría de los casos y son muy fáciles de usar.