Guía de creación de Kernel - TP ARQUI - ITBA

Lpinilla

October 31, 2018

Abstract

Este documento es una gua para ayudar en el desarrollo del kernel en el Trabajo Prctico de la materia Arquitectura de las computadoras. *Disclaimer:* Este documento no fue revisado por ningún miembro de la cátedra. Puede contener errores. Queda en su propia responsabilidad utilizar el contenido de este documento.

Consideraciones previas

Las versiones de Ubuntu mayores a 16.04 presentan problemas al querer manipular la IDT, se recomienda utilizar Docker **desde cero.**

PIC (Programmable Interface Controller)

Lo primero que hay que entender es que las computadoras tienen 2 PIC. El "MasterPIC" y el "SlavePIC". el slave esta "colgado" del master (cascading), por lo tanto se puede deshabilitar o ignorar lo que venga de l desde el master (IRQ02).
Cada PIC tiene 2 entradas, una para leer los datos y otra para conf igurarlos, el master esta en 0x21 y 0x20. El slave en 0xA1h y 0xA0 respectivamente.

IRQ

Cada PIC tiene 8 entradas, 1 bit por cada entrada $\longrightarrow$ 2bytes para configurarlo.
Las entradas del IRQ son:

IRQ	Descripcin
MasterPIC	
0	Timer-Tick
1	Keyboard
2	Cascade (used internally by the two PICs. never raised)
3	COM2 (if enabled)
4	COM1 (if enabled)
5	LPT2 (if enabled)
6	Floppy Disk
7	LPT1 / Unreliable "spurious" interrupt (usually)
SlavePIC	
8	RTC
9	Free for peripherals / legacy SCSI / NIC
10	Free for peripherals / SCSI / NIC
11	Free for peripherals / SCSI / NIC
12	PS2-Mouse
13	FPU / Coprocessor / Inter-processor
14	Primary ATA Hard Disk
15	Secondary ATA Hard Disk

Masking del PIC

Para cambiar la configuracin del PIC tenemos que hacer un masking de bits. Para deshabilitar algo, tenemos que ponerlo en 1, en caso contrario esta habilitado.

Supongamos que queremos solamente habilitar el teclado, para eso habra que solamente dejar en 0 el bit 1 del MasterPic. Para eso debemos encontrar el valor en hexa que deja al primer byte del pic en 1101.

Recordando que cada PIC tiene 2 bytes para configurarlo, tenemos que hallar el valor.

En este caso, queremos que este todo en 1 menos el anteltimo bit: 1111-1101. La primer parte corresponde al valor F mientras que la segunda al D. Por lo tanto, el valor a mandarle al MasterPIC es FD.

```
picMasterMask:
push rbp
mov rbp, rsp
mov ax, di
out 21h,al
pop rbp
retn
```

Anlogamente configuramos el slavePIC con el valor FF para no habilitar ninguna otra interrupcion.

Interrupciones de Hardware

Recordando que una interrupción es un evento externo que ocurre, veamos como podemos realizar driver de un perifrico especfico (el teclado).

Cadena de ejecución de interrupciones

Veamos como resuelve la pc cuando se interactúa con el teclado.

1. Se aprieta una tecla en el teclado.
2. El teclado activa una interrupción al PIC.
3. El PIC recibe la interrupción desde el IRQ01 y se fija si la deja pasar o no.
4. Si la deja pasar, le indica al procesador que tiene una interrupción.
5. El procesador le indica si esta listo o no para recibir interrupciones.
6. Si esta listo, le envía, el PIC le envía cual de sus interrupciones se activó.
7. El microprocesador con esa información va a buscar a la IDT el registro correspondiente a la interrupción IRQ01.

En la IDT, el PIC está mapeado directamente, osea que IRQ0 arranca en X0h, IRQ1 en X1h, ... donde X en principio es 0 Pero ac hay un problema ya que las primeras 32 entradas de la IDT son excepciones por lo tanto se pisarían las entradas, por eso, se "mueve" el inicio de las IRQ. En este caso sería simplemente que X valga 2.

Por lo tanto, la tabla de los IRQ arranca en 20 (32 en hexa).

Creación de Interrupciones

Para crear un driver tenemos que manejar las interrupciones del periférico (ej Teclado). Para eso, tenemos que hacer un par de cosas:

1. Crear la entrada en la IDT.
2. En la entrada de la IDT, asignar un puntero a función que va a ser la rutina de ejecución de la interrupción.
3. De ser necesario, llamar desde la rutina a una función en C.

Por ejemplo: Se crea la interrupción en la entrada 21h. $\longrightarrow$ la interrupción llama a la rutina de asm $\longrightarrow$ la rutina llama a una función de C que se encarga de interactuar con la lectura del teclado.

Hay que entender que la rutina apuntada por la IDT no es lo que finalmente será lo que lee del teclado, esta rutina llama a una función de C que se va a encargar de eso y de proveer más funcionalidades.

Entradas comunes

Entradas más utilizadas cuando se lee en asm.

Entradas	Dispositivo
20-21h	MasterPIC
A0-A1h	SlavePIC
60-64h	Keyboard