Real Time and Embedded Systems

by

Dr. Lesley Shannon

Email: lshannon@ensc.sfu.ca

Course Website: http://www.ensc.sfu.ca/~lshannon/courses/ensc351

Simon Fraser University

Slide Set: 7
Date: November 10, 2011

Slide Set Overview

- Interrupts
  - Let’s recall your assembly interrupt “past”
  - Now let’s think about interrupts in our Linux context

Recalling your previous experience with Interrupts

Hardware Interrupts

- What do you recall from your assembly/microprocessor days?
  - PIC – Programmable Interrupt Controller
  - IRQ – Interrupt ReQuest
  - IACK – Interrupt ACKnowledge

Hardware Interrupts

- What do you recall from your assembly/microprocessor days?
  - ISR – Interrupt Service Routine
  - IVT – Interrupt Vector Table
  - NMI – Non-Maskable Interrupts

Hardware Interrupts

- What do you recall from your assembly/microprocessor days?
  - Anything else?

Hardware Interrupts – Basic Hardware setup

Hardware Interrupts – Daisy Chaining

Hardware Interrupts – The Sequence

Types of Interrupts – Level - Sensitive

Types of Interrupts – Edge - Sensitive

Types of Interrupts – Edge - Sensitive

Things to worry about

- Interrupt Priority
- Interrupt Latency
- Responding to the correct interrupt
- Debouncing Hardware
  - Done for you?
  - Done by you?

Things to worry about

- ISR duration – how long does it take to clear the source of the interrupt?
- Anything Else?

Interrupts in Linux

- The interrupt runs at a priority higher than any software priority
- We use our “favourite” structure to communicate indicate that an interrupt has happened to the “normal” threads
  - sigevent
- Variables used in ISRs and threads need to be declared as ***volatile*** so that they are not cached
- Stack space for ISRs is very limited

Interrupts in Linux- Control Flow

Interrupts in Linux- Control Flow

Interrupts- Things to Remember

- Protect variables used by both the ISR and normal threads (remember we have SMT/SMP/CMP machines now)
- You can’t call printf
- in*/out* functions may be helpful
- May need to make variables volatile
- No everyone can add interrupts in all systems
  - May require I/O privity depending on the system (e.g. QNX**)

Interrupts- Things to Remember

- Interrupts
  - Runs at higher than thread level priority
  - Could crash the system if you do it wrong
  - Hard to debug
  - Unless interrupts are disabled (locked), will run when it is triggered

- May require root access aka “I/O privity”
  - In other words, only users writing/running apps from the root account (with setuid() to root) can do this

Questions?

- What is the necessary sequence of steps to guarantee the proper operation of interrupts in assembly?

- What's the priority level of ISRs?

Questions?

- Why do interrupt latency and duration matter?

- What's the difference between level and edge triggered interrupts