Übung zu Betriebssysteme

Aufgabe 2: Unterbrechungen

Wintersemester 2020/21

Bernhard Heinloth & Christian Eichler

Lehrstuhl für Informatik 4
Friedrich-Alexander-Universität Erlangen-Nürnberg

Lehrstuhl für Verteilte Systeme
und Betriebssysteme

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

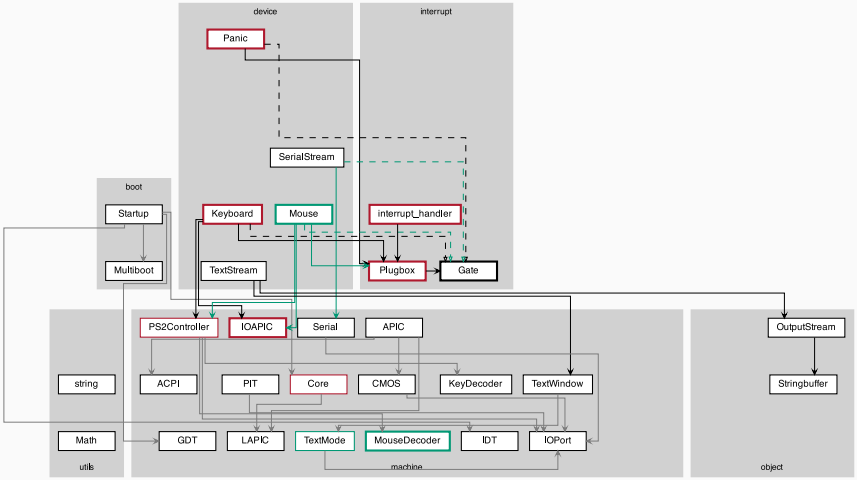
Lernziele

- Behandlung asynchroner Ereignisse
- Problematik und Schutz kritischer Abschnitte

Aufgabe

- Konfiguration externer Geräte über I/O APIC
- Treiber für **Tastatur** und (*optional*) **Maus**
- Wechselseitiger Ausschluss (**Ticket-**/**Spinlock**)
- *Optional:* **GDB Stub**

Umsetzung in StuBS


device
interrupt
Panic
SerialStream
boot
Startup
Keyboard
Mouse
interrupt_handler
Multiboot
TextStream
Plugbox
Gate
PS2Controller
IOAPIC
Serial
APIC
OutputStream
string
ACPI
PIT
Core
CMOS
KeyDecoder
TextWindow
Stringbuffer
Math
GDT
LAPIC
TextMode
MouseDecoder
IDT
IOPort
utils
machine
object

Programmierung des APIC Systems

Wo ist welches Gerät angeschlossen?

- Das kann evtl. unterschiedlich sein von Rechner zu Rechner!
- Steht in der Systemkonfiguration, heutzutage i.d.R. **ACPI** (Advanced Configuration and Power Interface)
- `APIC` stellt die relevanten Teile diese Informationen bereit

  `APIC::getIOAPICSlot` liefert für jedes Gerät den Index in die Redirection Table (siehe `enum Device` in `machine/apic.h`)

  `APIC::getIOAPICID` liefert die ID des I/O-APICs

Adressierung der APIC Nachrichten in StuBS

- Zusammenspiel mehrerer Faktoren
  - `Destination Mode`, `Destination Field` und `Delivery Mode` im I/O-APIC
  - Prozessor Priorität in den Local APICs der einzelnen CPUs
- **Ziel:** Gleichverteilung der Interrupts auf alle CPUs
  - Priorität der Prozessoren im Local APIC fest auf `0` einstellen
  - Im I/O-APIC `Lowest Priority` als Delivery Mode verwenden
  - Verwendung des `Logical Destination Mode`; bis zu 8 CPUs adressierbar
  - `Destination Field`: Bitmaske mit gesetztem Bit pro aktivierter CPU

Redirection Table Einträge in StuBS

Destination Field: Zieladresse des IRQs
bei `Dest. Mode == Physical` APIC ID der Ziel-CPU
bei `Dest. Mode == Logical` Gruppe von Ziel-CPUs

reserviert

Interrupt-Mask: Interrupt aktiv (`0`) oder inaktiv (`1`)
Trigger Mode: Flanken-(`0`) oder Pegelsteuerung (`1`)
Remote IRR: Art der erhaltenen Bestätigung
Interrupt Polarity: Active High (`0`) bzw. Active Low (`1`)
Delivery Status: Interrupt Nachricht noch unterwegs?
Destination Mode: Physical (`0`) oder Logical (`1`) Mode

Delivery Mode: Modus der Nachrichtenzustellung, z.B.
`0` **Fixed** – Signal allen Zielprozessoren zustellen
`1` **Lowest Priority** – CPU mit niedrigster Priorität

Interrupt Vektor: Nummer in der Vektortabelle (32 – 255)

KVM nutzt ggf. Vector Hashing

- **Problem:** Tastatur-Interrupts in KVM nur auf Core `1`
- **Lösung:** Datei `/etc/modprobe.d/kvm_options.conf` editieren, Eintrag **`options kvm vector_hashing=N`** hinzufügen und System neu starten.

(weitere Details siehe FAQ auf der Webseite)

Zusammenfassendes Beispiel: Keyboard Interrupt in StuBS

Vorbereitung

- **I/O APIC** initialisieren
  - **I/O APIC ID** setzen
  - Einträge in **Redirection Table** initialisieren (deaktivieren)
- **Keyboard** konfigurieren
  - Anmelden bei der **Plugbox**
  - Tastaturslot herausfinden und den entsprechenden Eintrag in der **Redirection Table** konfigurieren und aktivieren
  - Tastaturbuffer leeren
- **Interruptbehandlung** erstellen
  - Einsprungsroutinen `interrupt_entry` mit Aufruf zu `interrupt_handler` schreiben [wird in der Vorgabe bereits erledigt]
  - Eintragen in die **Interrupt Deskriptor Tabelle** (**IDT**) und diese in das Register `idtr` laden [ebenfalls erledigt]
  - Ereignisbehandlung in `interrupt_handler` mittels **Plugbox**
- Interrupts mit `Core::Interrupt::enable()` aktivieren

Ablauf

1. **Tastendruck** – Tastaturprozessor (in der Tastatur) meldet dies seriell an den **PS/2-Controller**
2. **PS/2-Controller** aktiviert Interruptleitung zu **I/O APIC**
   2.1 Anhand der **Redirection Table** wird Aktion gewählt
   2.2 Nachricht auf **APIC-Bus** mit Interruptnr. `33` und Ziel-CPU
3. entsprechender **LAPIC** empfängt Nachricht vom **APIC-Bus** und unterbricht CPU
4. **CPU** führt Unterbrechungsbehandlung aus
   4.1 Mittels Register `idtr` wird der entsprechende Eintrag in der **Interrupt Deskriptor Tabelle** ausgewählt und in die Einsprungsroutine gesprungen
   4.2 Einsprungsroutine `interrupt_entry_33` sichert Register und ruft `interrupt_handler` mit Parameter `vector = 33` auf
   4.3 `interrupt_handler` behandelt mittels **Plugbox** den Interrupt
5. **LAPIC** quittiert die Behandlung