Operating Systems
Interrupts

Prof. Dr. Oliver Hahm

Frankfurt University of Applied Sciences
Faculty 2: Computer Science and Engineering
oliver.hahm@fb2.fra-uas.de
https://teaching.dahahm.de

November 22, 2022

Some Analogies

- You are expecting a visitor – But your door does not have a Bell
- You are moderating a discussion – But a participant does not stop talking
- I am giving an online presentation – But you do not see my slides

Prof. Dr. Oliver Hahm – Operating Systems – Interrupts – WS 22/23

2/27

Interrupts are essential

- An Interrupt interrupts normal program execution for a certain purpose
- In practice no computer system can work without Interrupts
- **Without interrupts, preemptive multitasking is impossible**
  - Preemptive multitasking means: The operating system can remove the CPU from a process before its execution is complete

Interrupts vs. Polling
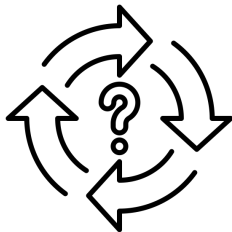
Created by Path Lord
from Noun Project

- **Polling:** In order to detect the occurrence of an event a process has continuously to send requests
- **Interrupt:** On occurrence of an event an interrupt is generated

Interrupts from an Application Perspective

> "Interrupts are an unpleasant fact of life; although they cannot
> be avoided, they should be hidden away, deep in the bowels of
> the operating system, so that as little of the operating system
> as possible knows about them."
> (Andrew S. Tanenbaum)

Agenda

- Interrupts, Exceptions, Traps

- Interrupt Handling

Agenda

■ Interrupts, Exceptions, Traps

■ Interrupt Handling

Terminology

Caution!

The terms *interrupt*, *exception*, or *trap* are used and defined slightly differently in the literature or documentation of the hardware manufacturers.

Interrupts and Exceptions

- In any computer system unpredictable events may (and will) occur at any time and must be handled
- Events that must be handled immediately are called interrupts
- Interrupts can be categorized into:
  - Hardware or External Interrupts
    - An I/O device provides feedback to a process
    - $\Rightarrow$ Asynchronous interrupt
  - Exceptions
    - Faults
      Error situation (error caused by an arithmetic operation)
      Division by zero, floating point error, address errors, . . .
    - Trap or Software Interrupt
      Triggered by a process $\Rightarrow$ Synchronous interrupt
      Examples are the exception 0x80 to switch from user mode to kernel mode and the single-stepping mode during program test (debugging, trace)

Interrupt Example

- X and Y are processes, which communicate via a network
  - Both processes are executed on different computers
  - On sending a message to the other process a reply is expected within a certain period of time
  - If the other process does not reply within that time the message must be sent again ($\rightarrow$ timeout)
    - **Reason**: The sender assumes that the message got lost
- Two ways exist to implement the described procedure:
 - Blocking
  - Non-blocking

Blocking Implementation

Blocking implementation

Sender
Process X

Receiver
Process Y

- Process X gets blocked until the message is acknowledged or the timeout expires
- If the acknowledgement arrives, sender process X may continue
    - Otherwise, process X must send the message again
- **Disadvantage**: Long idle times for process X arise

Message

blocked

Ack

Prof. Dr. Oliver Hahm – Operating Systems – Interrupts – WS 22/23

11/28
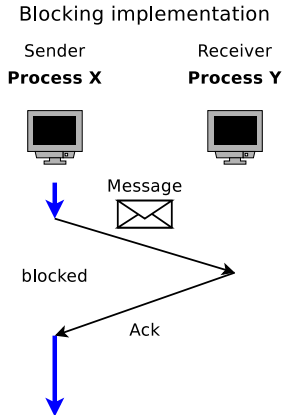
Non-blocking Implementation

- After process X sent the message, it continues to operate normally
  - If a timeout expires because of a missing acknowledgment, the operating system suspends the process
- The → context of the process is saved and a procedure for interrupt handling is called
  - In the example, the procedure would send the message again
 - If the execution of the procedure has finished, the process becomes reactivated

Non-blocking implementation

Sender
Process X

Receiver
Process Y

Message

Ack

Timeout

Message
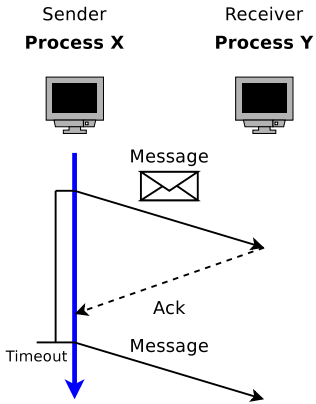
Subprograms without return value are typically called **procedures**. Subprograms with return value are called **functions** or **methods**.

Prof. Dr. Oliver Hahm – Operating Systems – Interrupts – WS 22/23

12/27

Hardware Interrupts

- Asynchronous
- An external device (e.g., a keyboard) signals an event (e.g., keypress)
- Internal devices (e.g., a clock) can signal an event (e.g., timer tick), too
- Notifications of another processor
- The operating system maintains a list of addresses of all interrupt service routines
    - This list is called interrupt vector (IV)
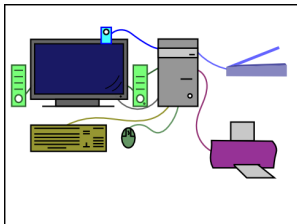
Exceptions

- Exceptions are caused by a running process
- When an exception occurs the CPU is interrupted and an handler in the kernel is activated
- A serious error has occurred, e.g.,:
    - Division by zero
    - Floating point operation without a FPU (floating point unit)
    - Invalid instruction
    - Segmentation fault (memory access violation)
    - Hardware failure
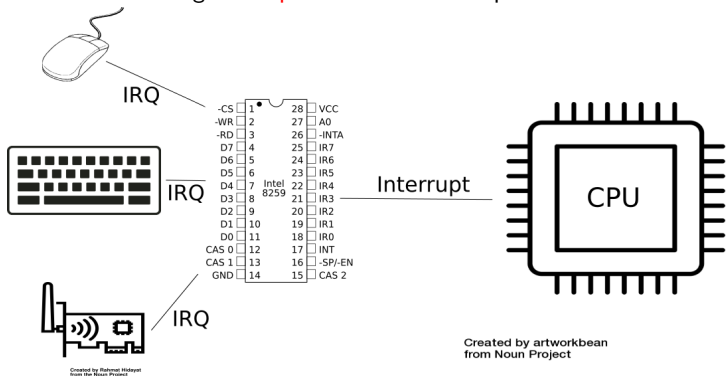- Exceptions can be used to generate an interrupt in software → trap
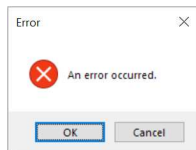
Traps

- Software instruction which causes an (intended) change of operation mode
- $\longrightarrow$ System calls
- processor synchronous

How is an Interrupt generated?

- An interrupt source generates an Interrupt Request (IRQ)
- IRQs are received by a Programmable Interrupt Controller (PIC) (for ARM CPUs Vectored Interrupt Controller (VIC))
- This controller determines the maximum number of different IRQs
    - Shared IRQs are possible
- It also manages and prioritizes the interrupts

Created by artworkbean
from Noun Project

Prof. Dr. Oliver Hahm – Operating Systems – Interrupts – WS 22/23

16/27

Interrupt Conflicts

- Two potential issues during interrupt handling:
  - During interrupt handling, further interrupts occur
  - Multiple interrupts occur at the same time
- Two possible solutions:
  - Sequential interrupt processing
  - Nested interrupt processing

Sequential Interrupt Processing

- Interrupts are strictly processed one after the other
- Interrupts are never interrupted
- **Advantage**: Simpler software design
- **Drawback**: Priorities and time-critical reactions are ignored
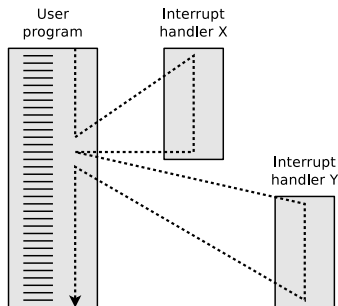
Source: William Stallings, Betriebssysteme, Pearson Studium, 2003

Nested Interrupt Processing

- Priorities are specified for the interrupts
- Interrupt handers can be interrupted, when an interrupt with higher priority occurs
- Interrupts with lower priority are delayed until all interrupts with a higher priority are handled
- **Drawback**: Interrupts with a low priority may be significantly delayed
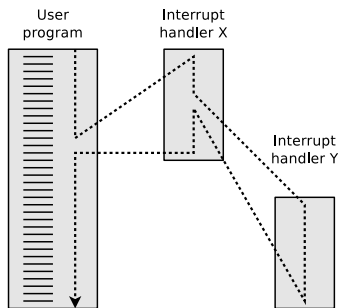
Source: William Stallings, Betriebssysteme, Pearson Studium, 2003

The real-time operating systems QNX Neutrino and Windows CE 5.0 both support nested interrupts

http://www.qnx.com/developers/docs/660/topic/com.qnx.doc.neutrino.sys_arch/topic/kernel_Nested_interrupts.html
http://msdn.microsoft.com/de-de/library/ms892539.aspx

Agenda

■ Interrupt Handling

Interrupt Handling

An interrupt request (IRQ) indicates an event and the operating system provides an **event handler**, the so called interrupt service routine (ISR)

Source: Eduard Glatz, Betriebssysteme
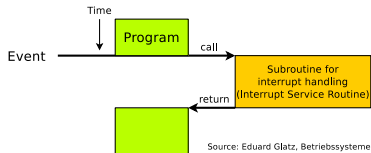
- The CPU is interrupted and for interrupt handling, the *ISR* of the kernel is called
  - The value of the *program counter* register is set to the address of the ISR which is executed next
  - The operating system stores the process context and restores the process context after the execution of the ISR has finished
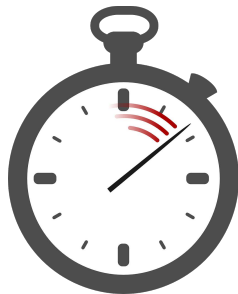
Interrupt Handling Procedure

- The common pattern is:
    - Save the state
    - Prepare ISR execution
    - Select the correct ISR
    - ISR execution
    - Potentially restore the state

Prof. Dr. Oliver Hahm – Operating Systems – Interrupts – WS 22/23

22/27
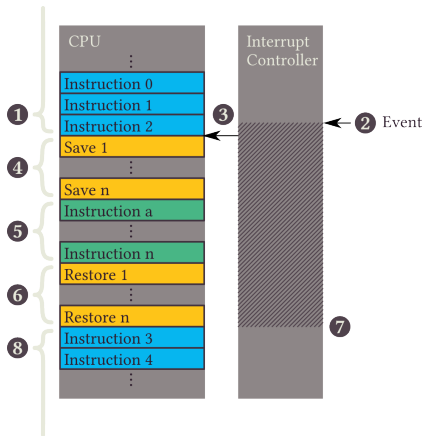
ISR Properties

- Short runtime
- No blocking operations
- Low memory footprint
- thread-safe

Delayed Interrupt Handling

- Only the most necessary is done in the ISR itself
- All more expensive operations are postponed (outside the interrupt context)
- On MS Windows this is called *Deferred Procedure Call (DPC)*

Delayed Interrupt Handling

- Only the most necessary is done in the ISR itself
- All more expensive operations are postponed (outside the interrupt context)
- On MS Windows this is called *Deferred Procedure Call (DPC)*

Example: character transmission

- Assume that characters are received on a certain line
- The application program proesses entire lines
- The ISR must save the characters
- The application program should be notified only when the first linebreak has been detected

Interrupt desired?

- Most interrupts can be deactivated (masked)
- Critical interrupts (e.g., exceptions) are not maskable ($\rightarrow$ *non-maskable interrupt (NMI)*)
- Masking an interrupt may be necessary to allow for concurrency

Problems of Interrupt Handling

- Concurrency
- Time criticality
- Hard to debug
- Nesting is possible (priorities)
- Compiler optimizations ($\rightarrow$ *volatile*)
- Non-reentrant functions

You should now be able to answer the following
questions:

- Why does any computer system need
  interrupts?
- What is the difference between hardware
  interrupts, exceptions, and traps?
- How are interrupts handled?