Proseminar: Konzepte von Betriebsystem-Komponenten (KVBK)

Schwerpunkt Linux

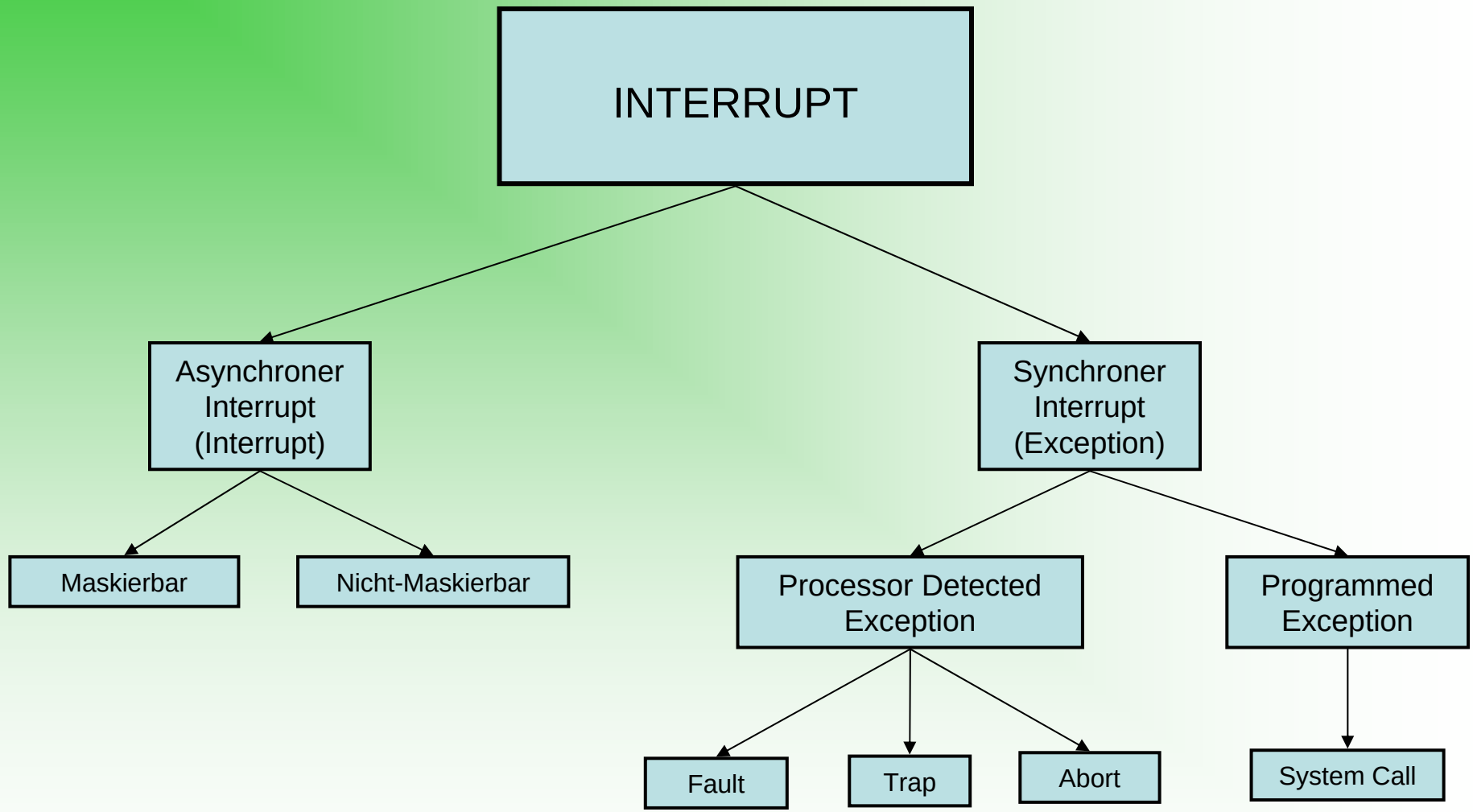
Interrupts, Softirqs, Tasklets, Bottom Halves

Thomas Engelhardt 10.11.2003

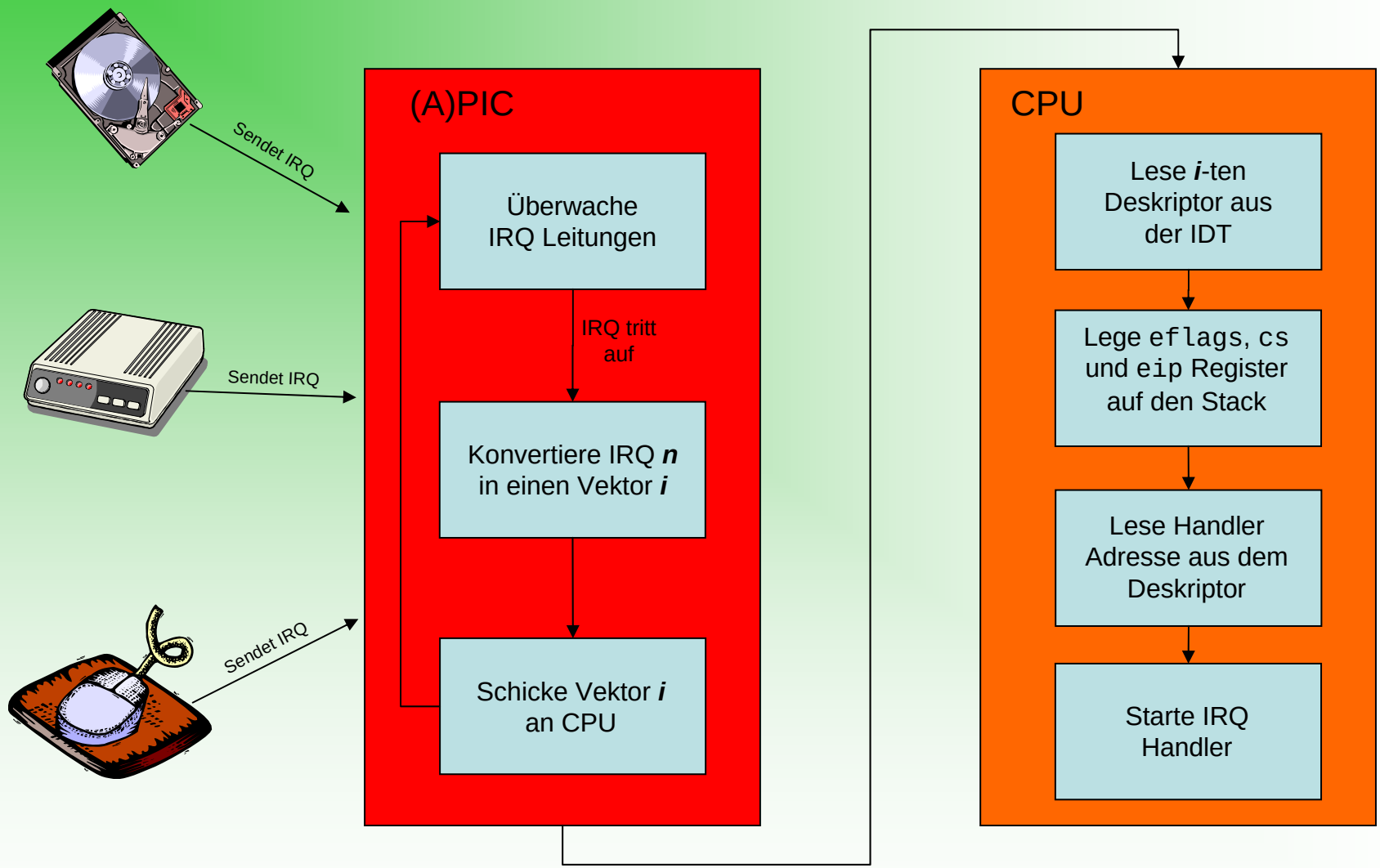
Übersicht:

- Klassifizierung
- Interrupts auf der Hardware Ebene
  - Vom Gerät zur CPU
- IDT: Schnittstelle zwischen Hardware und Software
- Behandlung der Interrupts auf Software Ebene
  - IRQs und Exceptions
  - Softirqs, Tasklets, Bottom Halves

Klassifizierung:



Vom Gerät zur CPU:



Interrupt Descriptor Table (IDT):

- Assoziiert jeden Vektor mit einem Interrupt- oder Exception Handler
- IDT enthält 256 Einträge (einen pro Vektor) á 8 Byte, sog. Deskriptoren
- Deskriptoren speichern Zeiger auf Handler
- Muss vor Aktivierung der Interrupts initialisiert werden
- 3 Typen von Deskriptoren:
  - *Task Gate:* Speichert Verweis auf einen Handler-Prozess, der den aktuell laufenden ersetzen muss.
  - *Interrupt Gate:* Speichert Adresse des Interrupt-Handlers. Deaktiviert Interrupts beim Aufruf durch Löschen des IF-Flags
  - *Trap Gate:* Speichert Adresse des Exception-Handlers. Lässt Zustand des IF-Flag unberührt.

Initialisierung der Interrupt Descriptor Table (IDT):

1. BIOS initialisiert IDT beim Starten des Rechners
   - Handler des BIOS können verwendet werden

2. Linux überschreibt IDT mit *ignore_int()* Handlern
   - Bei Aufruf: Fehlermeldung

3. IDT wird mit sinnvollen Handlern beschrieben
   - Vektoren der Exceptions sind vom Prozessordesign abhängig
   - Korrespondenz zwischen Vektoren und Geräten stellen Gerätetreiber her

Interrupt-Handling in Linux:

- Strategie des Interrupt Handling ist abhängig von der Art des Interrupts
  - I/O Interrupt
  - Timer Interrupt
  - Interprocessor Interrupt

I/O Interrupt Handling:

- Handler muss sich um Zustände der Register kümmern.
- I/O Handler startet eine *Interrupt Service Routine (ISR)*, die das Interrupt auslösende Gerät bedient.
  - *ISR* ist gerätespezifisch und muss vom Treiber bereitgestellt werden

- *IRQ Sharing:* Mehrere Geräte teilen sich eine IRQ Leitung.
  - Mit dem Interrupt verknüpfte ISRs werden sequenziell abgearbeitet

- *IRQ Dynamic Allocation:*
  - Zweck: Erlaube es mehreren IRQ Sharing unfähigen Geräten sich einen IRQ zu teilen.
  - IRQ Leitung, falls nicht belegt, wird dem Gerät auf Nachfrage zugewiesen und nach erfolgter Benutzung wieder freigegeben.

- Einteilung der Interrupts in Dringlichkeitsklassen:
  - *Kritisch:* Äußerst zeitkritische Aufgaben. Werden sofort in der ISR mit deaktivierten Interrupts bearbeitet.
  - *Nichtkritisch:* Werden von der ISR bearbeitet. Interrupts werden aber wieder aktiviert.
  - *Nichtkritisch verzögerbar:* Bearbeitung kann aufgeschoben werden. Verlagerung des Funktionsumfangs in *verzögerbare Funktionen (→ Softirqs, Tasklets, Bottom Halves)*

Timer Interrupts:

- Beliebiger Timer kann einen Interrupt auslösen
- Informiert den Kernel über ein verstrichenes Zeitintervall.
- Werden als I/O Interrupts behandelt

Interprocessor Interrupts:

- Insgesamt 5 Stück.
- Dienen zur Nachrichtenübermittlung zwischen den vorhanden CPUs.
- Senden und Handling der Interrupts ist einfach zu realisieren

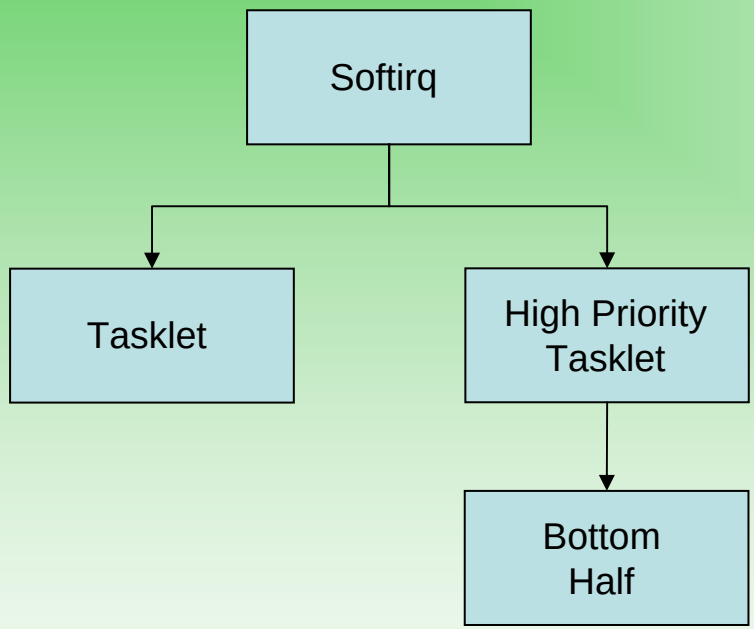
Exception Handling:

- Handler schickt ein UNIX-Signal an den Prozess, der die Exception ausgelöst hat
  - Prozess reagiert auf das Signal mit eigenem Signal-Handler
  - Existiert kein Signal-Handler, wird der Prozess terminiert

- Exception Handler prüft ob Exception im User Mode oder im Kernel Mode auftritt
  - Im User Mode: Normalfall
  - Im Kernel Mode:
    - Prüfe auf ein ungültiges Argument eines Systemaufrufs (System Call) und behandle Exception
    - Fehler im Kernel für Exception verantwortlich, führe die *die()* Funktion aus, um den Kernel zu terminieren.

Softirqs, Tasklets, Bottom Halves:

Hierarchie:

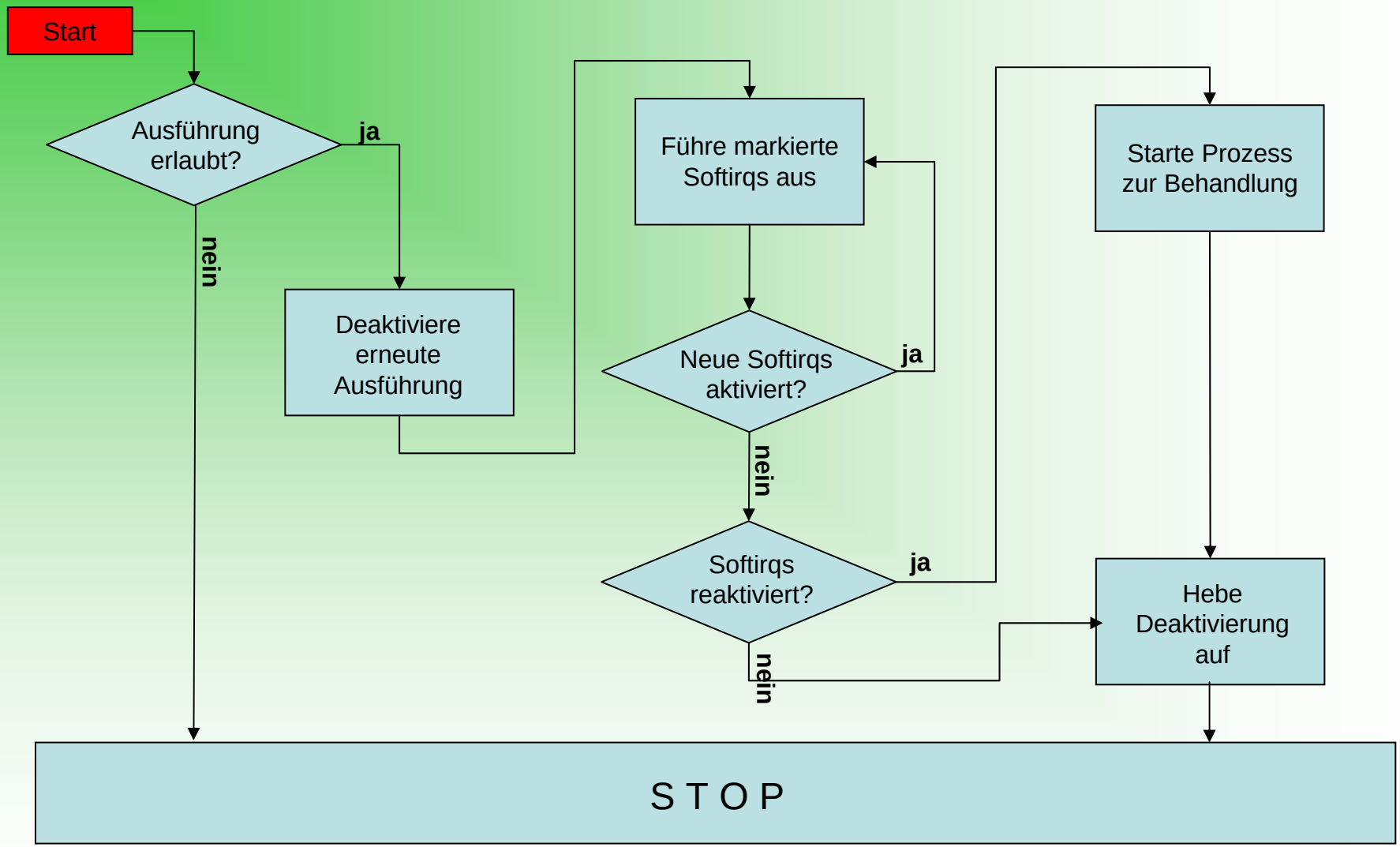


- *Softirq:* verzögerbare Funktion
  - Übernimmt zeitaufwendige Aufgaben einer Interrupt Service Routine
- *Tasklets*: Softirqs mit veränderten Eigenschaften
- *Bottom Halves:* Eingeschränkte High-Priority Tasklets

Softirqs, Tasklets, Bottom Halves; Allgemeines:

- Führen verzögerbare nichtkritische Aufgaben aus
  - Jeder Softirq (analog: Tasklet, Bottom Half) mit einer eigenen Funktion verknüpft
  - Verknüpfung mit der Funktion: *Initialisierung*
- Können auf der selben CPU nicht von einem anderen Softirq-Handler unterbrochen werden.
- Müssen für die Ausführung markiert werden, von
  - Interrupt Service Routinen
  - Softirqs (analog: Tasklets, Bottom Halves) selbst
- Ausführung startet mit bestimmten Ereignissen im System, z. B. nach Bearbeitung eines I/O Interrupt Handler.

Ausführung von Softirqs:



Softirqs:

- Anzahl maximal möglicher Softirqs auf 32 beschränkt.
  - Insgesamt nur vier in Verwendung.
- Verwaltung über eine statische Tabelle.
  - Platz in der Tabelle muss zur Compilerzeit bekannt sein.
- Parallele Ausführung zweier Softirqs *vom selben Typ* im Multiprozessorsystem erlaubt
  - Willkürliche Veränderung von gemeinsamen Daten parallel laufender Softirqs möglich → Programmierer muss Datensicherheit gewährleisten.
  - Sehr effizient.
- Kommen nicht in Gerätetreibern zum Einsatz

Tasklets:

- Zwei Tasklet-Typen:
  - High-Priority Tasklet: Ein Softirq mit höchstmöglicher Priorität
  - Tasklet: Von den verwendeten Softirqs, der mit niedrigster Priorität
- Verwaltung der Tasklet-Funktionen (bzw. Tasklets) über eine dynamisch wachsende Struktur
  - Anzahl der möglichen Tasklets quasi unbeschränkt
  - Platz in der Struktur nicht festgelegt → Zur Compilierzeit nicht bekannt.
  - Effizienz fällt mit wachsender Struktur.
- Nebenläufigkeit nur Tasklets *von unterschiedlichem Typ* gestattet → Gemeinsame Datenstrukturen werden nicht willkürlich verändert.
- Kommen in Gerätetreibern zum Einsatz

Bottom Halves:

- Bottom Half ist ein High Priority Tasklet → Bottom Halves werden immer zuerst ausgeführt
- Maximale Anzahl auf 32 beschränkt
- Verwaltung mit Hilfe einer statischen Tabelle:
  - Platz in der Tabelle muss zur Compilierzeit bekannt sein
- Parallele Ausführung nicht gestattet.
  - Geringere Effizienz im Multiprozessorsystem.
- Kommen nur in wenigen älteren Gerätetreibern und für Aufgaben des Kernels zum Einsatz.

Erweiterung der Bottom Halves

- Gestatte dem Kernel mehrere Funktionen mit einem Bottom Half auszuführen.
- Verknüpfung von mehreren Funktionen mit einem, dem Interrupt Handling zugewiesenen Bottom Half.
- Gruppe von Funktionen heißt *Task Queue:*
  - Zum Beispiel von I/O Gerätetreibern benutzt
  - Ausführung von mehreren Kernel-Funktionen pro Timer-Interrupt

Softirqs, Tasklets, Bottom Halves; Übersicht:

Verzögerbare Funktion	***Platzvergabe (in Tabelle / Liste)***	***Nebenläufigkeit***
Softirq	Statisch. Platz zur Compilierzeit bekannt	Grundsätzlich möglich. Auch Softirqs des selben Typs
Tasklet	Dynamisch. Platz erst zur Laufzeit bekannt	Nur Tasklets unterschiedlichen Typs.
Bottom Half	Statisch. Platz zur Compilierzeit bekannt	Nicht möglich.

Wozu der Aufwand?

Hardwareseitig:

- Zeiteffizienz:
  - Vermeiden von Pollingverfahren
- Möglichkeit die CPU über Fehlerzustände zu informieren
- Nachrichtenaustausch zwischen CPUs
- ***Softwareseitig:***
- Zeiteffizienz durch:
  - Verschachtelung von Interrupt / Exception-Handlern
  - Verlagerung zu vieler verzögerbarer Funktionen in einen Prozess (*ksoftirqd_CPUn; n=logische Nummer der CPU*)