Umsetzung von Z-Operations-Schemata in Implementationen

Jan Peleska, Jan Bredereke

Vorlesung SCS 3, 20.5.2003

Scheduling-Struktur für sicherheitsrelevante Systeme

A. Prioritäten-gesteuertes, präemptives Scheduling

B. Zyklisches sequentielles Scheduling

- nicht präemptiv

A. Prioritäten-gesteuertes, präemptives Scheduling

+ einfache Anwendungsprogrammierung:
  + keine Zeitüberwachung (Dauer der CPU-Nutzung) erforderlich
  + sequentielles Programmiermodell:
    Warten auf benötigte Inputs ist erlaubt
+ Prioritäten gemäß Sicherheitsrelevanz einstellbar
+ schnelles Interrupt-Handling für sicherheitsrelevante Unterbrechungen
  → *Low-Latency*-Kernel erforderlich:

  Auch BS-Operationen können „schnell“ unterbrochen werden, um eine kritische Applikation zu starten.

  Kernel-Preemption: Kernel-Aufgaben sind unterbrechbar.

− schwer verifizierbar

B. Zyklisches sequentielles Scheduling

+ garantierte Zykluszeiten
+ *keine* Interrupts, sondern Interface-Polling
  + DMA
  + Dual-Ported RAM (auf I/O-Karte)
    (braucht weniger BS-Unterstützung)
− CPU-Zeit für vergebliches Polling

+ leichte Verifizierbarkeit:
  Reaktion auf kritischen Input erfolgt spätestens nach einem Zyklus

– State-Machine-Programmiermodell:
  statt Warten auf Inputs:

  1. Zustand merken
  2. CPU freigeben

– Zeitüberwachung muß in der Applikation erfolgen
  (deshalb oft Hardware-Watchdog notwendig)

Von einer Z-Spezifikation zur Programmstruktur

1. Z-Schemata definieren i.a. *partielle* Operationen.
   Sie sind nur auf einer *Teilmenge* des Zustandsraums ausführbar. Diese ist durch die Precondition des Schemas definiert.

2. Ziel: Eine totale Operation aus den einzelnen Schemata herstellen durch Schemakomposition

   - Schema-Konjunktion: $S_{tot} = S_{part1} \wedge S_{part2} \wedge \ldots \wedge S_{partn}$
   - Schema-Disjunktion: $S_{tot} = S_{part1} \vee S_{part2} \vee \ldots \vee S_{partn}$

3. Umsetzung von S_{tot}:

```
while (1) {
    S1();
    S2();
    ...
    Sn();
}

void Si() {
    if ("pre-condition erfüllt") {
        ...
    }
}
```