4 Übungsaufgabe: Unterbrechungen

In der Vorlesung wurde die Behandlung von Unterbrechungen auf realen CPUs besprochen.
In dieser Aufgabe soll die Unterbrechungsbehandlung durch den Einsatz von Signalhandlern nachgebildet werden und ein zweistufiges Behandlungskonzept *(FLIH/SLIH)* implementiert werden.

4.1 Unterbrechungsbehandlung

Die Unterbrechungsbehandlung in *Luxe* soll mit einem zweistufigen Konzept bestehend aus *First-* und *Second-Level Interrupt Handler* implementiert werden:

1. Der *First-Level Interrupt Handler (FLIH)* wird bei jeder auftretenden Unterbrechung ausgeführt und hat die Aufgabe, eine nachgelagerte Unterbrechungsbehandlung (*Second-Level Interrupt Handler (SLIH)*) in eine Warteschlange einzuhängen und die Unterbrechung zu quittieren. Nach der Rückkehr aus dem *FLIH* sollen alle in der Warteschlange vorhandenen *SLIH* ausgeführt werden.

2. Der *Second-Level Interrupt Handler (SLIH)* erledigt die eigentliche Arbeit der Unterbrechungsbehandlung und kann durch den *FLIH* unterbrochen werden.

Verschiedene Arten der Unterbrechungsbehandlung sind denkbar, welche unterschiedliche Anforderungen an die Synchronisation und Anzahl der *SLIH*-Warteschlange(n) stellen. In dieser Aufgabe sollen die folgenden drei Möglichkeiten implementiert werden:

4.1.1 Behandlung von FLIH/SLIH durch eine vorgegebene CPU

Sowohl der *FLIH* wie auch der *SLIH* sollen auf der gleichen (virtuellen) CPU ausgeführt werden. Hierbei wird die betreffende CPU zur Konfigurationszeit des Systems festgelegt.

4.1.2 Behandlung von FLIH/SLIH durch eine zufällige CPU

Durch die Zustellung von Unterbrechungen in Form von Signalen ist es möglich, diese zufällig vom Betriebssystem an eine (virtuelle) CPU zuzustellen, die dieses Signal nicht maskiert. Auf dieser CPU soll sowohl der *FLIH* wie auch der dazugehörige *SLIH* ausgeführt werden.

4.1.3 Behandlung des FLIH durch eine zufällige und des SLIH durch eine vorgegebene CPU

Die *FLIH* sollen von einer zufällig vom Betriebssystem ausgewählten CPU verarbeitet werden, die *SLIH* jedoch von einem zur Konfigurationszeit bestimmten Prozessor. Hierbei muss die *FLIH* CPU einen Interprozessorinterrupt and die *SLIH* CPU senden, um die Abarbeitung der *SLIH*-Warteschlange anzustossen.

4.2 Unterbrechungsquellen

In dieser Aufgabe soll die Behandlung und (teilweise) Auslösung von Unterbrechungen verschiedener Quellen implementiert werden. Jeder Quelle soll zur Konfigurationszeit (durch *kconfig*) eine Signalnummer zugewiesen werden.

4.2.1 Interprozessorinterrupts

Interprozessorinterrupts werden genutzt, um von einer CPU aus eine andere zu unterbrechen. Damit kann beispielsweise auf neue Elemente in der *SLIH*-Warteschlange hingewiesen werden. Diese Unterbrechungen werden gezielt an eine andere CPU gesendet, wofür sich unter Linux der Systemaufruf `tgkill` eignet.

4.2.2 Zeitgeber

Um in den folgenden Aufgaben präemptives Multitasking zu ermöglichen, soll ein Zeitgeber implementiert werden, welcher (konfigurierbar) alle 100 Millisekunden eine Unterbrechung erzeugt. Dies soll durch einen eigenen Thread geschehen, welcher selbst alle Signale blockiert und nicht fest an eine reale CPU gebunden ist.
Für diese Aufgabe soll in der *SLIH*-Funktion nur die aktuelle CPU-Nummer und „Timerinterrupt" auf `stdout` ausgegeben werden.

4.2.3 Andere (externe) Quellen

Für externe Unterbrechungen soll ein Signal definiert werden, welches der Fadengruppe von außen zugestellt werden kann. Im Augenblick soll die *SLIH*-Unterbrechungsbehandlung nur die aktuelle CPU-Nummer und das Signal auf `stdout` ausgeben.

Aufgaben:

→ Implementierung von FLIH/SLIH (mit Warteschlange) auf Basis von Signalen. Hierbei sind auch die Anforderungen an die Synchronisation der Warteschlange(n) zu bestimmen und umzusetzen. Die Behandlungsmethode ist über *kconfig* konfigurierbar zu gestalten.

→ Implementierung der Unterbrechungsquellen und Behandlungsfunktionen für *Interprozessorinterrupts* und *Zeitgeber*, sowie nur der Behandlungsfunktion für *externe Unterbrechungsquellen*.

Hinweise:

- Zustellung von Signalen an einen einzelnen Faden kann unter Linux mit dem Systemaufruf `tgkill` erfolgen.
- Jeder Faden braucht eine eigene Signalmaske. Diese kann unter Linux mit dem Systemaufruf `rt_sigprocmask` gesetzt werden.

4.3 Abgabe: am 10.06.2010