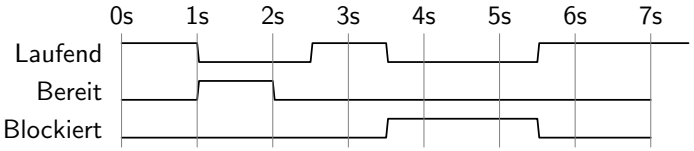
Grundlagen der Betriebssysteme
Blatt 05
Gruppe 055

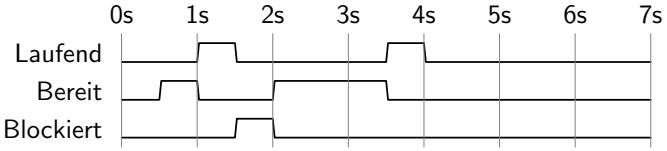
Marco Deuscher
Ibrahem Hasan

Mai 2019

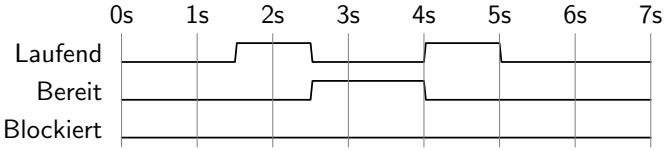
1 Round-Robin und Highest Priority Scheduling



(a) Effizientes RR Proc. A



(b) Effizientes RR Proc. B



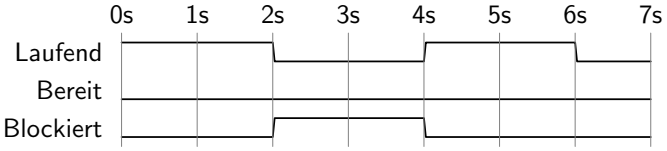
(c) Effizientes RR Proc. C

Abbildung 1: Effizientes RR

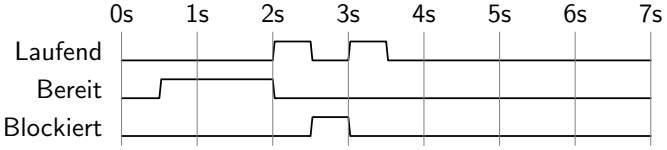
2 Threads und Prozesse

(a) Mehrere Threads innerhalb eines Prozesses teilen sich

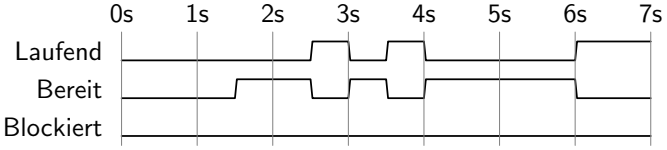
- Text und Data Section werden geteilt, d.h. alle Threads eines Prozesses nutzen den gleichen Code im Speicher und es ist theoretisch Zugriff auf die gleichen Daten möglich



(a) HPF Proc. A



(b) HPF Proc. B



(c) HPF Proc. C

Abbildung 2: HPF

- Threads eines Prozesses nutzen die gleiche Schutzumgebung. Threads eines Prozesses arbeiten alle im gleichen virtuellen Address Space
- nutzen IO-Ressourcen gemeinsam (bpsw. Dateien, Netzwerkverbindungen, etc)

(b) Ressourcen die jeder Thread exklusive hat sind

- eigene virtuelle CPU, d.h. eigenen Registersatz
- eigenen Stack
- 3. antwort einfügen

(c) Ein Nachteil von User-level Threads ist, dass der Kernel nicht weiß wie viele Threads in einem Prozess vorhanden sind. Der Prozess als Ganzes bekommt eine bestimmte Zeit vom Scheduler, unabhängig davon unter wie vielen Threads diese Zeit geteilt werden muss.
Ein weiterer Nachteil ist, dass User-level Threads non-blocking System calls benötigen. Ansonsten wird bei einem blocking Syscall durch einen Thread der gesamte Prozess geblockt, obwohl es noch Threads geben könnte welche *ready* sind.
Der Nachteil, dass non-blocking System calls benötigt werden, ist der Hauptgrund warum heutzutage vor allem eine Kombination aus Kernel- und User-level Threads verwendet werden.