Betriebssysteme 1

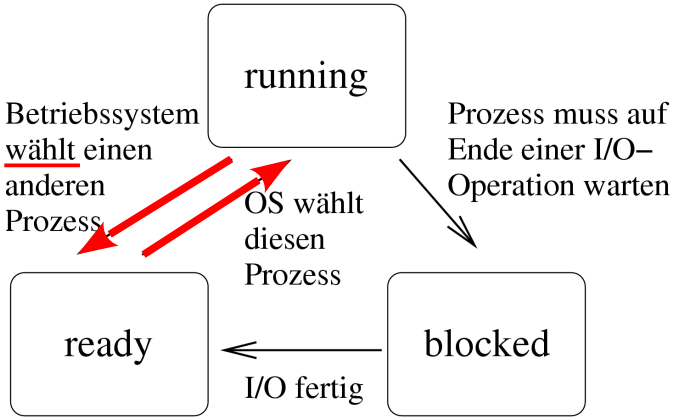
SS 2020

Prof. Dr.-Ing. Hans-Georg Eßer
Fachhochschule Südwestfalen

Foliensatz B: v2.0, 2020/04/16
- Scheduler

Scheduling: Prozess auswählen

Zustandsübergänge

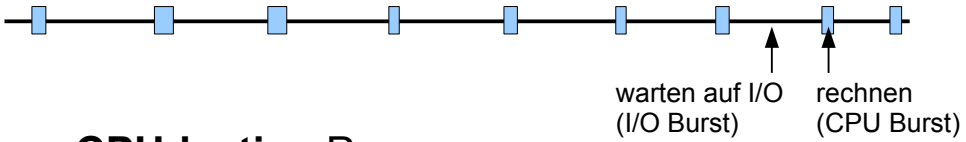


Wann wird Scheduler aktiv?

- Neuer Prozess entsteht (`fork`)
- Aktiver Prozess blockiert wegen I/O-Zugriff
- Blockierter Prozess wird bereit
- Aktiver Prozess endet (`exit`)
- Prozess rechnet schon zu lange
- Interrupt tritt auf

Begriffe

- **Kooperatives** Scheduling vs. **Präemptives** (**unterbrechendes**) Scheduling
- **I/O-Bursts**, **CPU-Bursts**
- **I/O-lastige** vs. …



… **CPU-lastige** Prozesse

Häufige Prozesswechsel?

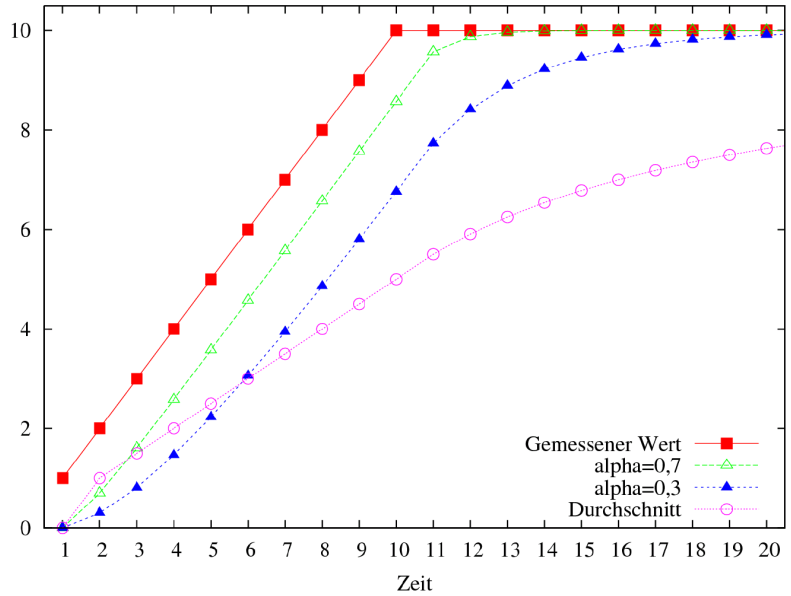
Faktoren

- **Zeit für Kontext-Switch:** Scheduler benötigt Zeit, um Prozesszustand zu sichern
  → verlorene Rechenzeit
- **Wartezeit der Prozesse:** Häufigere Wechsel erzeugen stärkeren Eindruck von Gleichzeitigkeit

Einfache Verfahren

- FCFS
- SJF und SRT → Burst-Dauer-Prognose?
  - Mittelwert
  - Exponentieller Durchschnitt
    $S_{n+1} = \alpha T_n + (1-\alpha) S_n$
    α: Gewicht zwischen 0 und 1

Burst-Dauer-Prognose

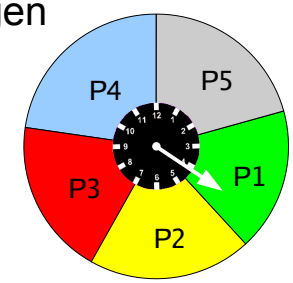


SJF vs. SRT

- Vorsicht (auch für Prüfung): SJF und SRT werden oft verwechselt.
- „SJF < SRT“
  - (SJF ist i.d.R. schlechter als SRT, weil es keine Unterbrechungen kennt)
  - dieser Vergleich gilt auch lexikalisch (Eselsbrücke)

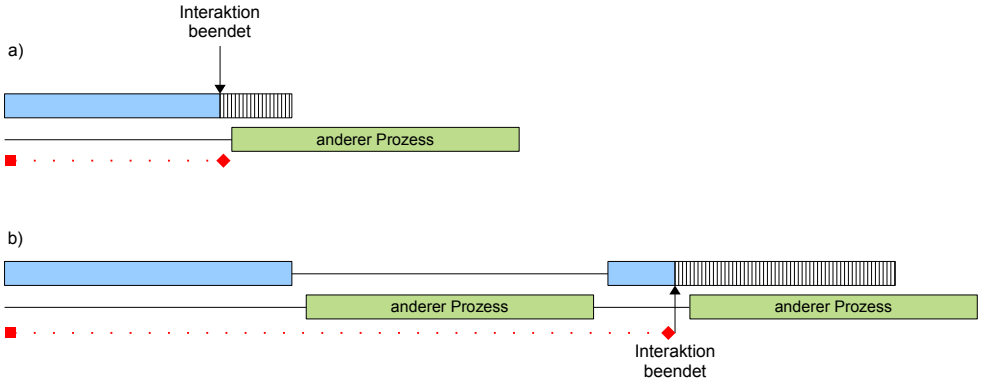
Round Robin / Time Slicing (1)

- Wie FCFS – aber mit Unterbrechungen
- Alle bereiten Prozesse in einer Warteschlange
- Jedem Thread eine Zeitscheibe (quantum, time slice) zuordnen
- Ist Prozess bei Ablauf der Zeitscheibe noch aktiv, dann:
  - Prozess verdrängen (→ Zustand „bereit“)
  - Prozess ans Ende der Warteschlange hängen
- Wie groß soll das Quantum sein?



Round Robin / Quantum

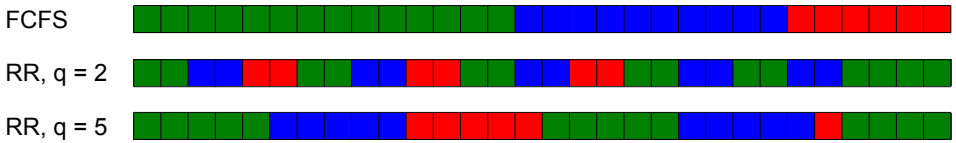
Oft: Quantum *q* etwas größer als typische Zeit, die das Bearbeiten einer Interaktion benötigt



Round-Robin-Beispiel

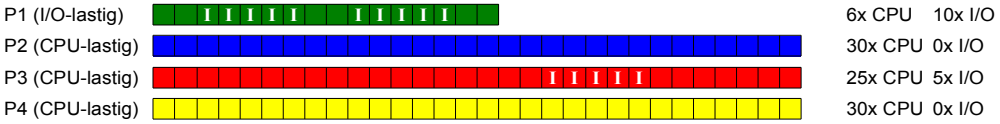
Szenario: Drei Prozesse

- FCFS (einfache Warteschlange, keine Unterbrechung)
- Round Robin mit Quantum 2
- Round Robin mit Quantum 5

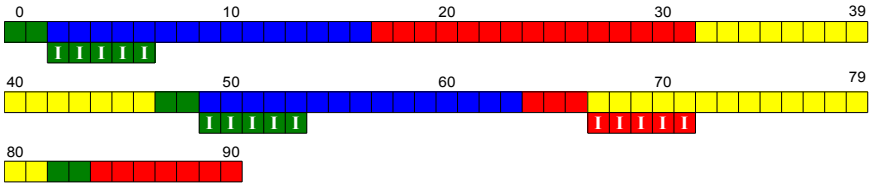


Round Robin: I/O- vs. CPU-lastig

Idealer Verlauf (wenn jeder Prozess exklusiv läuft)



Ausführreihenfolge mit Round Robin, Zeitquantum 15:



Prozess	CPU-Zeit	I/O-Zeit	Summe	Laufzeit	Wartezeit *)
P1	6	10	16	84	68
P2	30	0	30	64	34
P3	25	5	30	91	61
P4	30	0	30	82	52

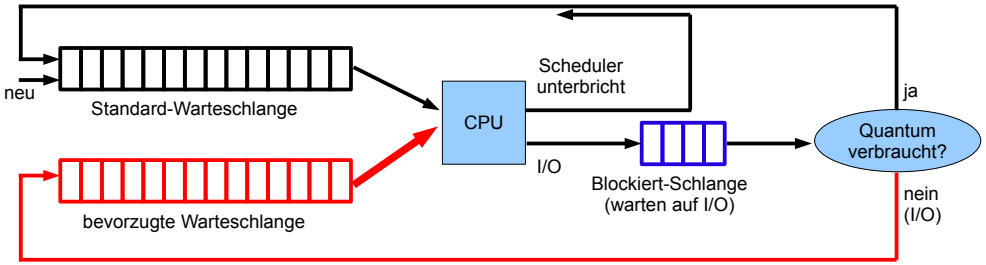
*) im Zustand *bereit*, nicht *blockiert*!

Virtual Round Robin

Beobachtung: RR unfair zu I/O-lastigen Prozessen

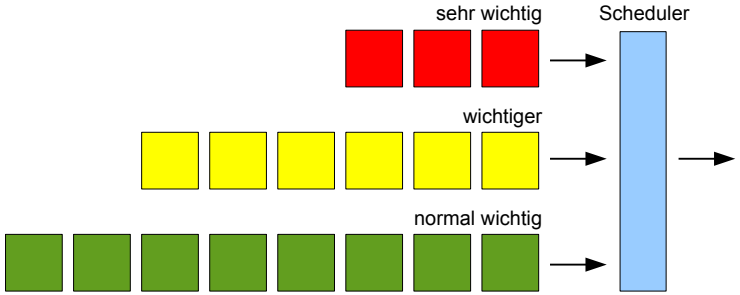
– CPU-lastige nutzen ganzes Quantum,
– I/O-lastige nur einen Bruchteil

Lösungsvorschlag: VRR

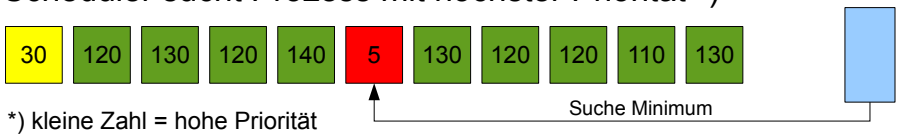


Prioritäten-Scheduler

a) Mehrere Warteschlangen für Prioritätsklassen



b) Scheduler sucht Prozess mit höchster Priorität *)



*) kleine Zahl = hohe Priorität

Prioritäten → Probleme

Prozesse können „verhungern“ → Aging

Prioritätsinversion:

- Prozess hoher Priorität ist blockiert (benötigt ein Betriebsmittel)
- Prozess niedriger Priorität besitzt dieses Betriebsmittel, wird aber vom Scheduler nicht aufgerufen (weil es höher-prioritäre Prozesse gibt)
- Beide Prozesse kommen nie dran, weil immer Prozesse mittlerer Priorität laufen
- Ausweg: Aging

Prioritäten: Aging

Aging:

- Priorität eines Prozesses, der bereit ist und auf die CPU wartet, wird regelmäßig erhöht
- Priorität des aktiven Prozesses und aller nicht-bereiten (blockierten) Prozesse bleibt gleich
- Ergebnis: Lange wartender Prozess erreicht irgendwann ausreichend hohe Priorität, um aktiv zu werden

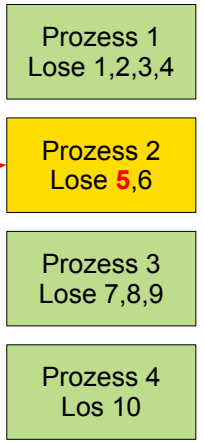
Prioritäten: variables Quantum

Verschiedene Quantenlängen

- Mehrere Prioritätsklassen:
  1. Priorität = 1 Quantum, 2. Priorität = 2 Quanten,
  3. Priorität = 4 Quanten, 4. Priorität = 8 Quanten
- Prozesse mit hoher Priorität erhalten kleines Quantum.
- Geben sie die CPU vor Ablauf des Quantums zurück, behalten sie hohe Priorität
- Verbrauchen sie Quantum, verdoppelt Scheduler die Quantenlänge und stuft die Priorität runter – solange, bis Prozess sein Quantum nicht mehr aufbraucht

Lotterie-Scheduler

Scheduler zieht
Los Nr. 5



Lotterie-Scheduler: Gruppenbildung

- Gruppenbildung und Los-Austausch:
  - Zusammenarbeit Client / Server
  - Client stellt Anfrage an Server, gibt ihm seine Lose und blockiert
  - Nach Bearbeitung gibt Server die Lose an den Client zurück und weckt ihn auf
  - Keine Clients vorhanden?
    → Server erhält keine Lose, rechnet nie