Betriebssysteme

Scheduling und Kommunikation

Von

Prof. Dr. Franz-Karl Schmatzer

Literatur Verzeichnis

- Mandl, Peter; Grundkurs Betriebssysteme; 5.Aufl. 2020; Springer Verlag
- Baun, Christian, Betriebssysteme kompakt, 2.Aufl., Springer 2020
- Tanenbaum, Andrew; Moderne Betriebssysteme; 3.Aufl. 2009; Pearson Studium
- W. Stallings; Operating Systems; 9.ed; Pearson 2018
- M.Russinovich, D.A.Solomon,A.Iomescu; Windows Internal Part 1; 7 Auflage, Microsoft Press 2017

Gliederung

- Scheduling Ziele
- Scheduling Typen
- Scheduling Strategien
- Multi-level-Scheduling
- Deadlocks
- Kommunikation

Scheduling

- Scheduling:
  - Allgemein: Vorgehensweise bei der Zuteilung von Betriebsmitteln
  - Konkret: Zuteilung des Prozessors auf Prozesse
- Dispatcher:
  - Durchführung des Prozesswechsel
  - Sicherung des Zustandes des laufenden Prozesses
  - Aktivierung des nächsten Prozesses

Scheduling Ziele I

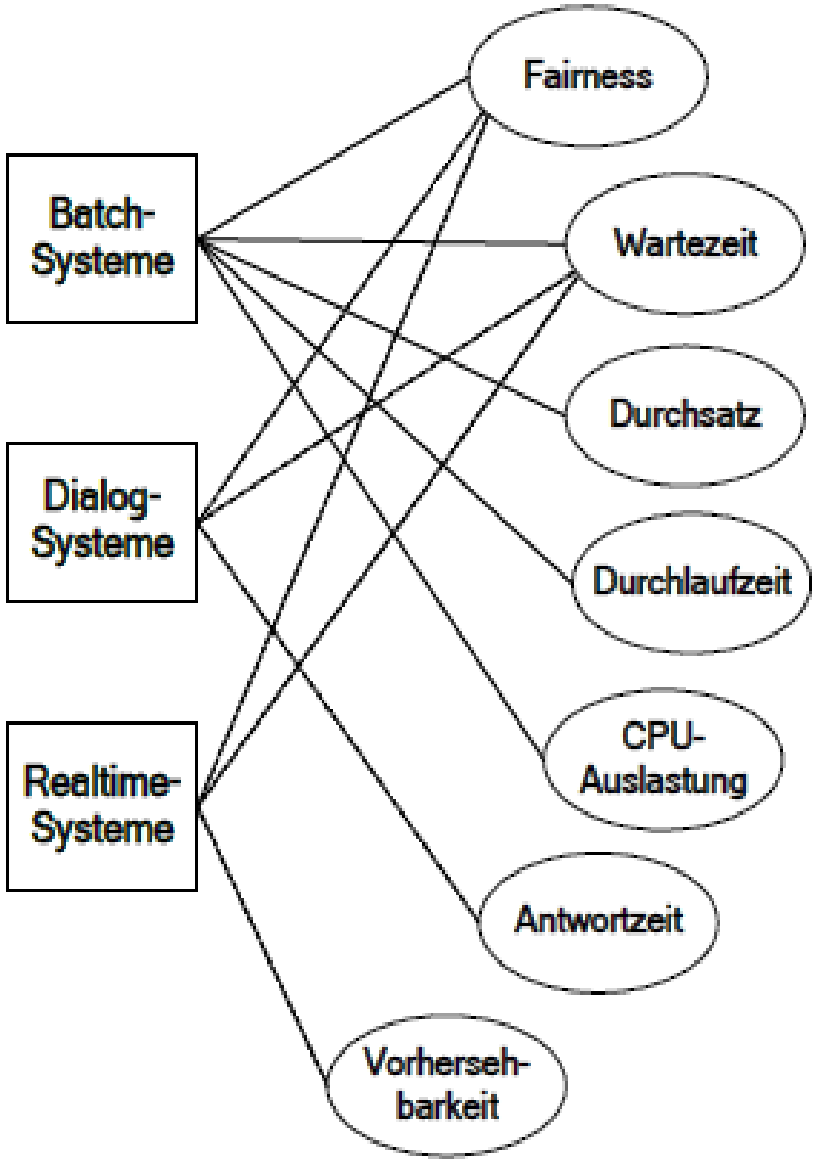
- Auslastung der CPU
  - Ziel ist die 100%ige Auslastung der CPU,
  - normal 40% - 90%
- Durchsatz (throughput)
  - Anzahl der Jobs pro Zeiteinheit sollte maximal sein
- Faire Behandlung (fairness)
  - Jeder Benutzer sollte im Mittel den gleichen CPU-Zeitanteil erhalten
  - Es wird zwischen weak fair und strong fair unterschieden.

Scheduling Ziele II

- Wartezeit (waiting time)
  - Wartezeit in der Bereit-Liste minimieren (einziger Scheduling-parameter)
- Ausführungszeit (turnaround time)
  - Die Zeitspanne vom Jobbeginn bis zum Jobende sollte minimal sein. Sie enthält alle Zeiten in Warteschlangen, der Ausführung (Bedienzeit) und der Ein- und Ausgabe
- Antwortzeit (response time)
  - Die Zeit zwischen einer Eingabe und der Übergabe der Antwortdaten an die Ausgabegeräte sollte minimal werden (interaktive Systeme!)
- Vorhersehbarkeit.
  - Wann erhält ein Prozess die CPU.

Scheduling Typen

- Je nach Anwendungstyp gibt es verschiedene Scheduling Ziele.
  - Batch
  - Dialog
  - Realtime
- Für Realtime System sind beispielsweise Vorhersehbarkeit besonders wichtig.



Aufgaben Scheduling Verfahren

- Erläutern Sie die Begriffe zu den folgenden zwei Scheduling-Verfahren und geben Sie dazu auch Beispiele an:
  - Non preemptive Scheduling
  - Preemptive Scheduling
    - zeitgesteuert
    - ereignisgesteuert

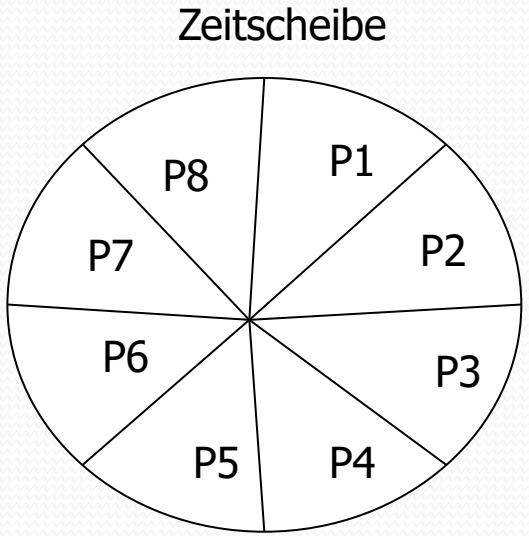
Scheduling-Verfahren

- non-preemptive, auch „run-to-completion“-Verfahren:
  - Ein Prozess darf nicht unterbrochen werden, bis er seine Aufgaben vollständig erledigt hat.
  - MS-DOS und auch die ersten Windows-Systeme und noch Mac OS 9!
- preemptive Scheduling Verfahren
  - Rechenbereite Prozesse können suspendiert werden.
  - Basis der Zeitscheibenverfahrens.
  - Die meisten heutigen Betriebssysteme (Windows, Linux)

Das Zeitscheibenverfahren

- **Zeitscheibenverfahren** (Timesharing)
  - CPU-Zuteilung durch Betriebssystem, wenn
    - laufender Prozess auf ein Ereignis wartet
    - laufender Prozess eine bestimmte Zeit den Prozessor nutzte
- Zeitscheibe = Quantum: 10 ms, 100 ms, ...
  - abhängig von CPU und Betriebssystem
  - Zeitscheibe je CPU

- Beispiel: Ein Quantum von 10 ms bedeutet 100 Kontextwechsel pro Sekunde (Overhead vernachlässigt)



@Mandl

Clock-Intervall und Quantum

- Clock-Intervall
  - ca. 10 ms bei x86 Singleprozessoren
  - ca. 15 ms bei x86 Multiprozessoren
  - siehe *clockres-Tool* zum Feststellen des Clock-Intervalls
- Quantum
  - Quantumszähler je Thread
    - Auf 6 bei Workstations (Windows 2000, XP, …) eingestellt
    - Auf 36 bei Windows-Servern eingestellt
    - Quantumszähler wird je Clock-Interrupt um 3 reduziert

  → Quantumsdauer = 2 (Workstation) bzw. 12 (Server) Clock-Intervalle
    - 2 * 15 ms = 30 ms bei x86-Workstations
    - 12 * 15 ms = 180 ms bei Servern

Scheduling-Strategien

- Im Fall des preemptive Schedulings gibt es mehrere Strategien.
- Die eingesetzten Strategien sind dabei abhängig von der Art der Systeme. Man unterscheidet:
  - Batch-Systeme
  - Dialog-Systeme
  - Realtime-Systeme
- In den aktuellen Betriebssystemen wird aber meistens eine Kombination von Strategien unterstützt.

Scheduling-Strategien

Batch-Systeme

- Typische Verfahren bei Batch-Systemen:
  - First Come First Serve (FCFS)
  - Shortest Job First (SJF), Shortest Process First (*SPF*) bzw. Shortest Process Next(*SPN*)
  - Shortest Remaining Time Next (*SRTN*)
- Diese Verfahren sind vom Prinzip her eigentlich non-preemptive. Sie werden aber heute als preemptive Verfahren implementiert.

Scheduling-Strategien

Dialog-Systeme

- Typische Verfahren bei Dialog-Systemen:
  - Round Robin (RR)
  - Priority Scheduling (PS)
  - Shortest Remaining Time First (SRTF)

  bzw. Shortest Remaining Process Time(SRPT)
  - Garantiertes Scheduling
  - Lottery Scheduling

Scheduling-Strategien

Realtime-Systeme

- Typische Verfahren bei Realtime-Systemen:
  - Minimal Deadline First
  - Polled Loop
  - Interrupt-gesteuert
- **Minimal Deadline First:** Auswahl des Prozesses mit der kleinsten nächsten Zeitschranke(deadline).
- **Polled Loop:** Alle Geräte (Ereignisquellen) werden zyklisch nach einem anstehenden Ereignis abgefragt, und dieses wird dann gleich bearbeitet.
- **Interrupt-gesteuerte** Systeme warten z.B. in einer Warteschleife auf Interrupts von Ereignisquellen und führen dann die geeignete Interrupt-Service-Routine (ISR) aus.

Scheduling-Strategien

Batch-Systeme

- **FCFS** bearbeitet die im System ankommenden Aufträge in der Reihenfolge ihres Eintreffens.
- ***SJF*** sucht sich dagegen immer den Job bzw. Prozess aus, von dem es die kürzeste Bedienzeit erwartet.
- ***SRTN*** ist eine verdrängende Variante von *SJF*. Er *wählt* als nächstes immer den Prozess mit der am kürzesten verbleibenden Restrechenzeit im System aus. Dieser Algorithmus bevorzugt wie SJF neu im System ankommende Prozesse mit kurzer Laufzeit.

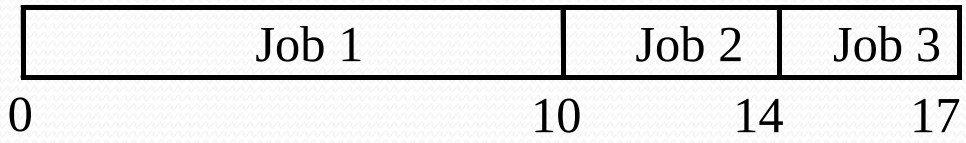
Scheduling-Strategien

Batch-Systeme

Job1=10
Job2=4
Job3=3

First Come First Serve (FCFS).

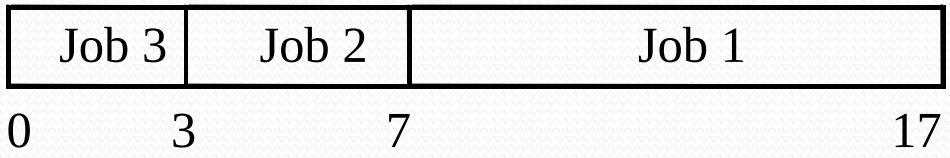
Einsortieren in der Ankunftsreihenfolge (FIFO-Warteschlange).



Ausführungszeit: Job1=10, Job2=14, Job3=17; Mittelwert=41/3=13.67.

Shortest Job First (SJF)

Job mit kürzester Bedienzeit zuerst (min. mittlere Wartezeit).



Ausführungszeit: Job1=17, Job2=7, Job3=3; Mittelwert=27/3=9.

Scheduling-Strategien

Batch-Systeme

Nachteil

- ***Verhungern***(engl. Starvation) von Prozessen ist möglich. Es kann also Prozesse geben, die nie eine CPU zugeteilt bekommen und daher nicht ausgeführt werden.
  - Beispiel SJF: Es kommen laufend kurze laufende Prozesse in die Queue. Damit wird der langlaufende Prozess nie fertig. Widerspricht der Fairness Forderung.
- **Bestimmung der Zeit**, die ein Prozess braucht, ist eigentlich nicht möglich. Daher sind diese Verfahren in der Praxis kaum umsetzbar.

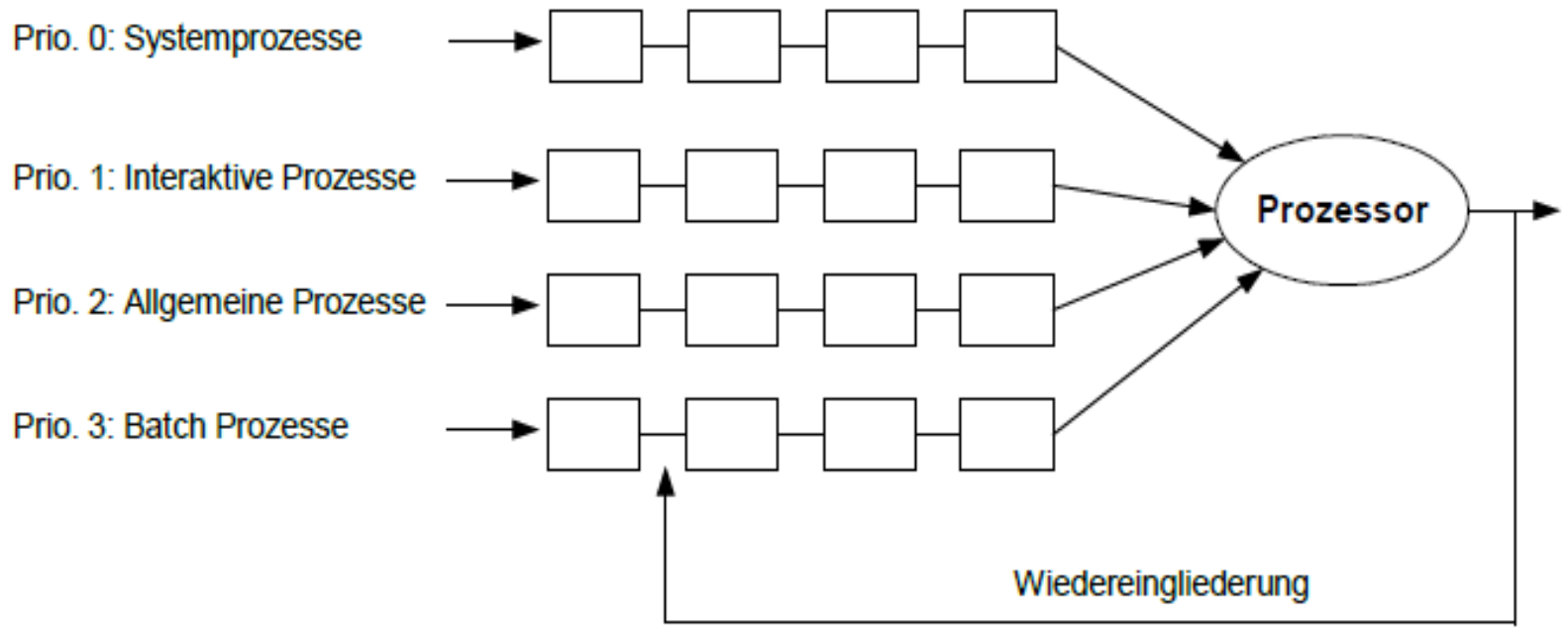
Aufgaben Scheduling Strategien

- Erläutern Sie folgenden Scheduling-Strategien. Gehen Sie auch auf die Vor- und Nachteile ein.
  - Round Robin (RR)
  - Priority Scheduling (PS)
  - Garantiertes Scheduling
  - Lottery Scheduling

Multi-level-Scheduling

Prinzipien

- Aufbau einer eigenen Warteschlange für jede Priorität oder Prozesstyp.
- Wenn die Prozesse die Warteschlange wechseln können, nennt man das Multi-Level-Feedback-Scheduling.



Multi-level-Scheduling

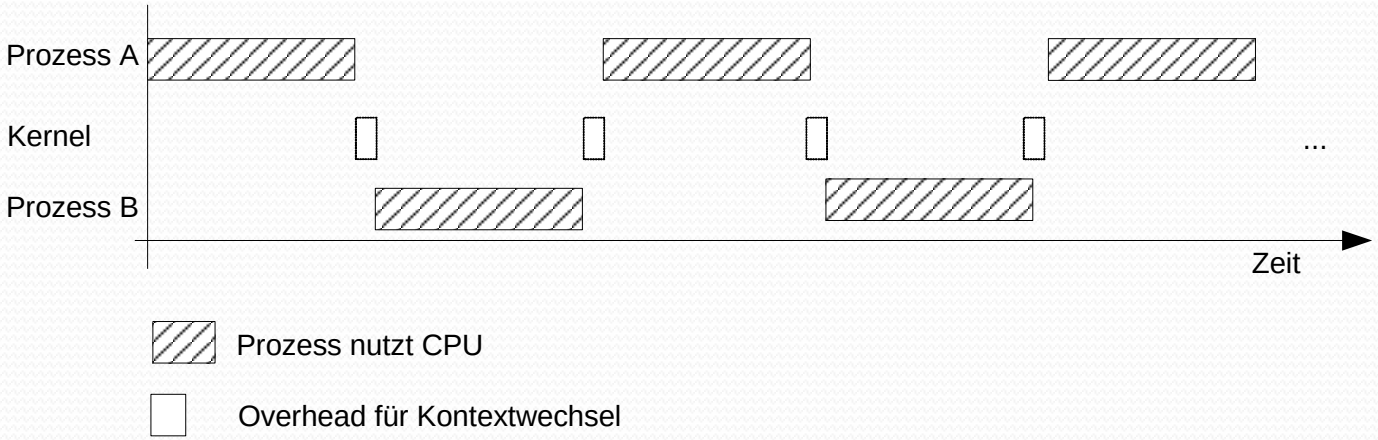
Round-Robin mit Prioritäten

- Round Robin Verfahren mit Prioritätensteuerung
- Folgende Aspekte sind zu beachten:
  - Wie wird die Priorität einzelner Prozesse bestimmt?
  - Wie lange soll die Zeitscheibe für einzelne Prozesse (das Quantum) sein?

Multi-level-Scheduling

Round-Robin mit Prioritäten

- Wie lange soll die Zeitscheibe (Quantum) für einzelne Prozesse sein?
  - Kurzes Quantum → hoher Overhead
  - Beispiel: Quantum 10 ms, Kontextwechsel: 1 ms → 10 % Overhead
  - Statische und/oder dynamische Festlegung möglich
  - 10 bis 200 ms heute üblich



@Mandl

Multi-level-Scheduling

Round-Robin mit Prioritäten

- Wie wird die Priorität für einzelne Prozesse eingestellt bzw. ermittelt?
  - Statische Festlegung zum Start des Prozesses
  - Zusätzlich adaptive (dynamisch) Festlegung möglich
  - → Man spricht dann auch von relativer Priorität

Round-Robin mit Prioritäten

Prioritätenwahl

- Prioritätenwahl:
  - statisch zu Beginn der Laufzeit und dann keine Änderung mehr
  - Dynamische adaptive Prioritätenwahl
  - Kombination der beiden Möglichkeiten, d.h. statische Festlegung zu Beginn und dann dynamisch adaptiv anpassen. (Heute bevorzugtes Verfahren)
- Probleme:
  - Gerechte Ressourcenzuteilung

Round-Robin mit Prioritäten

Quantum

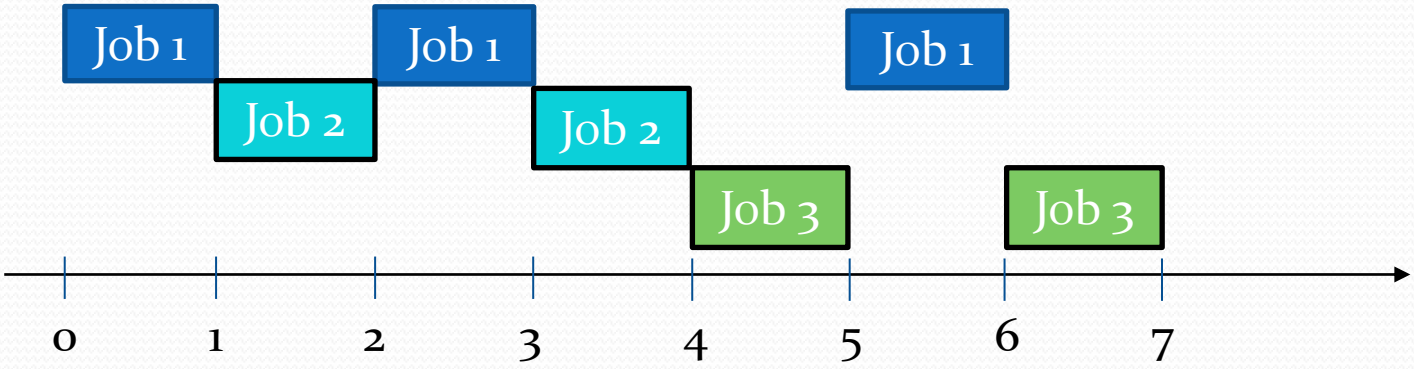
- In der Praxis stellt folgendes Verfahren einen recht guten Ansatz dar:
  - Das Quantum wird initial eingestellt
  - Anpassung dynamisch, so dass auch Prozesse mit niedriger Priorität die CPU ausreichend lange erhalten.
  - Ein-/Ausgabe-intensive Prozesse erhalten ein höheres Quantum, rechenintensive ein kürzeres. Damit können vernünftige Antwortzeiten von Dialogprozessen erreicht werden, und die rechenintensiven Prozesse können gut damit leben.
  - Die Prozess-Prioritäten werden statisch voreingestellt und unterliegen einer dynamischen Veränderung. Die aktuelle Priorität wird auch als relative Priorität bezeichnet.
  - Quantumgröße von 10ms bis 200ms

Vergleich von Scheduling-Verfahren

- Zum Vergleich der Scheduling-Verfahren kann man folgende Größen heranziehen:
  - **Durchlaufzeit** (= Verweilzeit): Gesamte Zeit, in der sich ein Prozess im System befindet (Servicezeiten + Wartezeiten).
  - **Wartezeit:** Zeit, die ein Prozess auf die Ausführung warten muss, also die Summe aller Zeiträume, in denen ein Prozess warten muss.
  - **Bedienzeit** (Servicezeit): Zeit, in der ein Prozess die CPU hält und arbeiten kann.
  - **Antwortzeit:** Zeit, in der ein Anwender auf die Bearbeitung seines Auftrags warten muss.
  - **Durchsatz:** Anzahl an Prozessen, die ein System in einer bestimmten Zeit bearbeiten kann.
  - **CPU-Auslastung:** Auslastung der CPU während der Bearbeitung von Prozessen in Prozent der Gesamtkapazität.

Scheduling-Strategien

wichtige Parameter



- Beispiel:
- Verweilzeit: Job1 = 6, Job2= 3, Job3 = 3
- Wartezeit: Job1= 3, Job2=1, Job3 =1
- Bedienzeit: Job1=3, job2=job3 = 2

Vergleich von Scheduling-Verfahren

Job	A	B	C	D	E
Ablaufzeit	10	6	4	2	8
Priorität	3	5	2	1	4

Gesucht sind jeweils die gesamte (V_{all}) und die durchschnittliche (V_{avg}) Verweilzeit im System bei Einsatz folgender Scheduling-Algorithmen:

1) Priority Scheduling (nicht verdrängend)
2) FCFS unter Berücksichtigung der Reihenfolge-Annahme: A, B, D, C, E (nicht verdrängend)
3) Shortest Job First (nicht verdrängend)
4) RR mit statischen (also sich nicht verändernden) Prioritäten bei einem Quantum von 2 Sekunden (verdrängend)

Vergleich von Scheduling-Verfahren

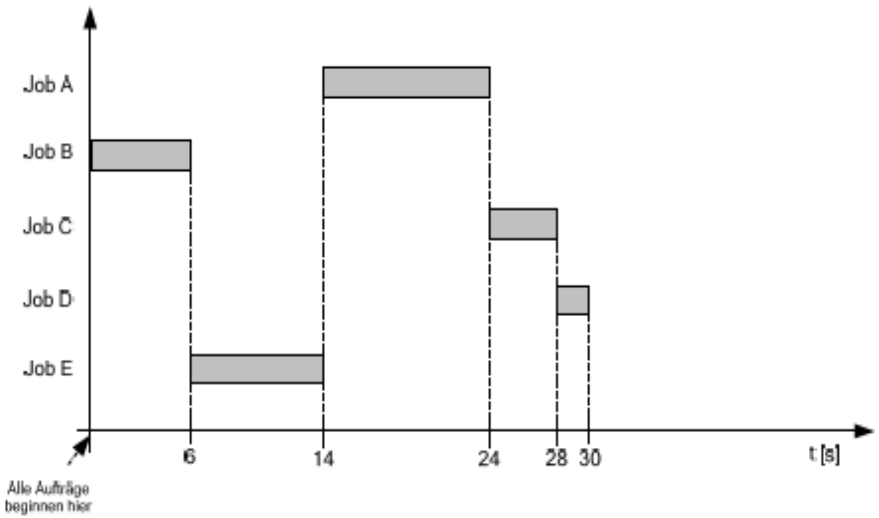
- **Priority Scheduling:** Die folgende Tabelle zeigt die Reihenfolge der Auftragsausführung bei Priority Scheduling sowie die jeweiligen Verweilzeiten der einzelnen Jobs:

Job	B	E	A	C	D
Verweilzeit	6	14	24	28	30

Die Summe über alle Verweilzeiten ist $V_{all} = 6 + 14 + 24 + 28 + 30 = 102$ s.

Die durchschnittliche Verweilzeit ist $V_{avg} = V_{all} / 5 = 20{,}4$ s.

Die Abbildung 5-5 verdeutlicht den Ablauf nochmals.



Vergleich von Scheduling-Verfahren

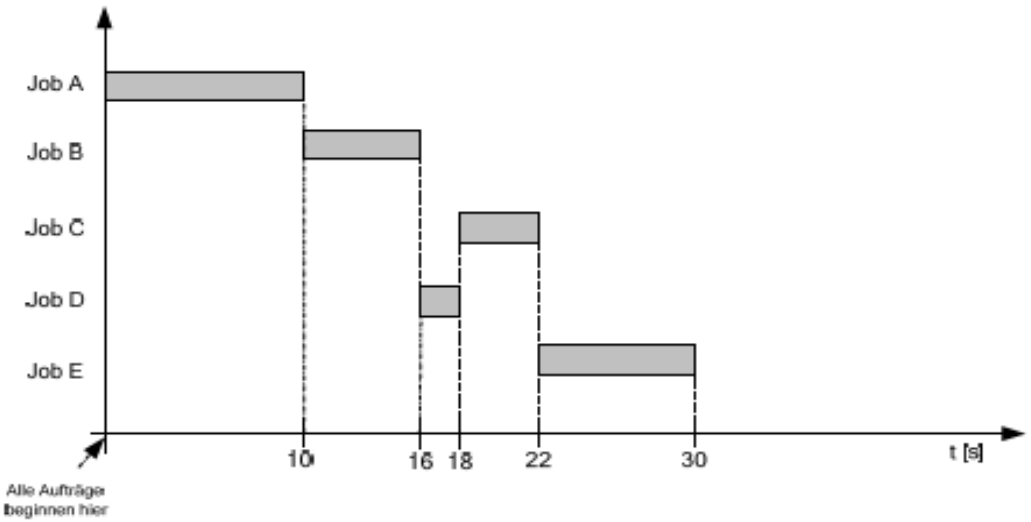
- **FCFS-Scheduling:** Die folgende Tabelle zeigt die Reihenfolge der Auftragsausführung bei FCFS-Scheduling sowie die jeweiligen Verweilzeiten der einzelnen Jobs.

Job	A	B	D	C	E
Verweilzeit	10	16	18	22	30

Die Summe über alle Verweilzeiten $V_{all} = 10 + 16 + 18 + 22 + 30 = 96$ s.

Die durchschnittliche Verweilzeit ist $V_{avg} = V_{all} / 5 = 19{,}2$ s.

FCFS ist also in diesem Szenario besser als reines Prioritäten-Scheduling. Die Abbildung 5-6 zeigt den Ablauf bei FCFS-Scheduling.



Vergleich von Scheduling-Verfahren

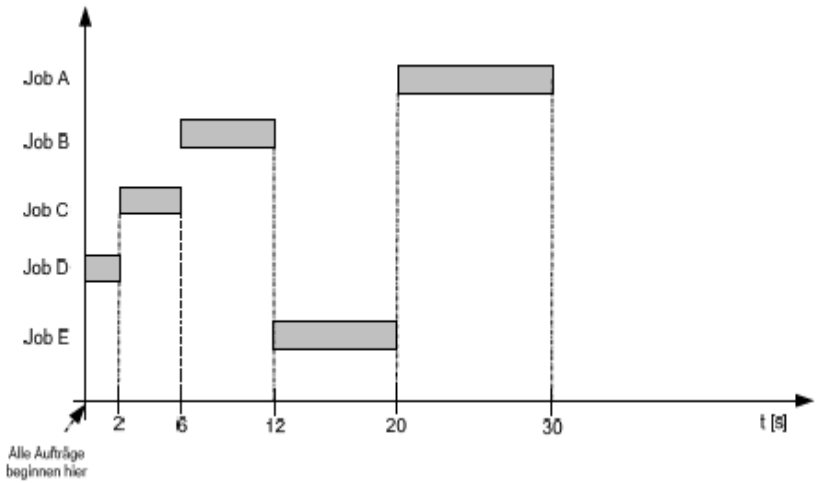
- **SJF-Scheduling**: Die folgende Tabelle zeigt die Reihenfolge der Auftragsausführung.

Job	D	C	B	E	A
Verweilzeit	2	6	12	20	30

Die Summe über alle Verweilzeiten $V_{all} = 2 + 6 + 12 + 20 + 30 = 70$ s.

Die durchschnittliche Verweilzeit ist $V_{avg} = V_{all} / 5 = 14{,}0$ s.

SJF ist, wie bereits erläutert, die beste aller Lösungen was die Verweilzeit anbelangt.



Vergleich von Scheduling-Verfahren

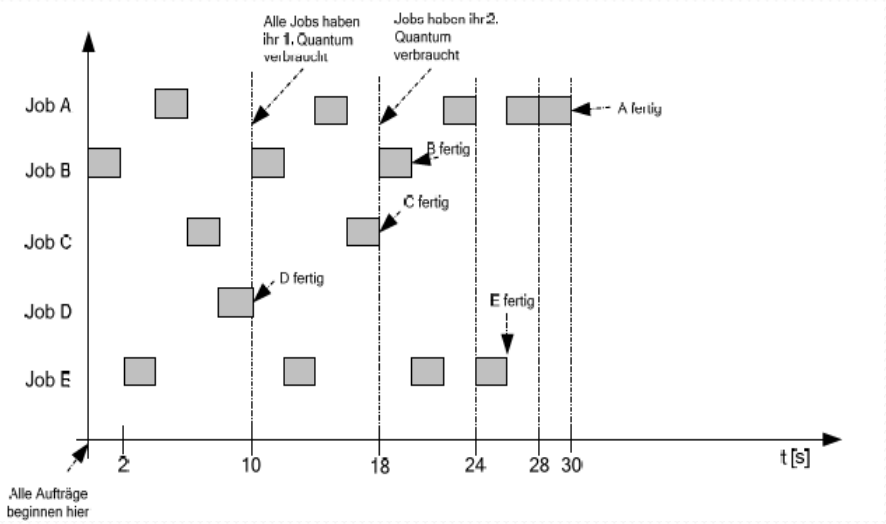
- **RR-Scheduling mit Prioritäten.** Die folgende Tabelle zeigt die Reihenfolge der Auftragsausführung bei RR-Scheduling unter Berücksichtigung von Prioritäten und einer Zeitscheibe von 2 s, wobei der Overhead für den Prozesswechsel vernachlässigt wird.

Job	A	B	C	D	E
Verweilzeit	30	20	18	10	26

Die Summe über alle Verweilzeiten $V_{all} = 30 + 20 + 18 + 10 + 36 = 104$ s.

Die durchschnittliche Verweilzeit ist $V_{avg} = V_{all} / 5 = 20{,}8$ s.

RR-Scheduling mit Prioritäten ist also die schlechteste aller Varianten, dafür aber auch die gerechteste.



Aufgabe Job-Scheduling

Jobname	A	B	C	D	E
Zeit t	8	6	10	4	2
Priorität P	3	5	2	1	4

Gesucht sind jeweils die gesamte (V_{all}) und die durchschnittliche (V_{avg}) Verweilzeit im System bei Einsatz folgender Scheduling-Algorithmen:

1) Priority Scheduling (nicht verdrängend)
2) FCFS unter Berücksichtigung der Reihenfolge-Annahme: A, B, D, C, E (nicht verdrängend)
3) Shortest Job First (nicht verdrängend)
4) RR mit statischen (also sich nicht verändernden) Prioritäten bei einem Quantum von 2 Sekunden (verdrängend)

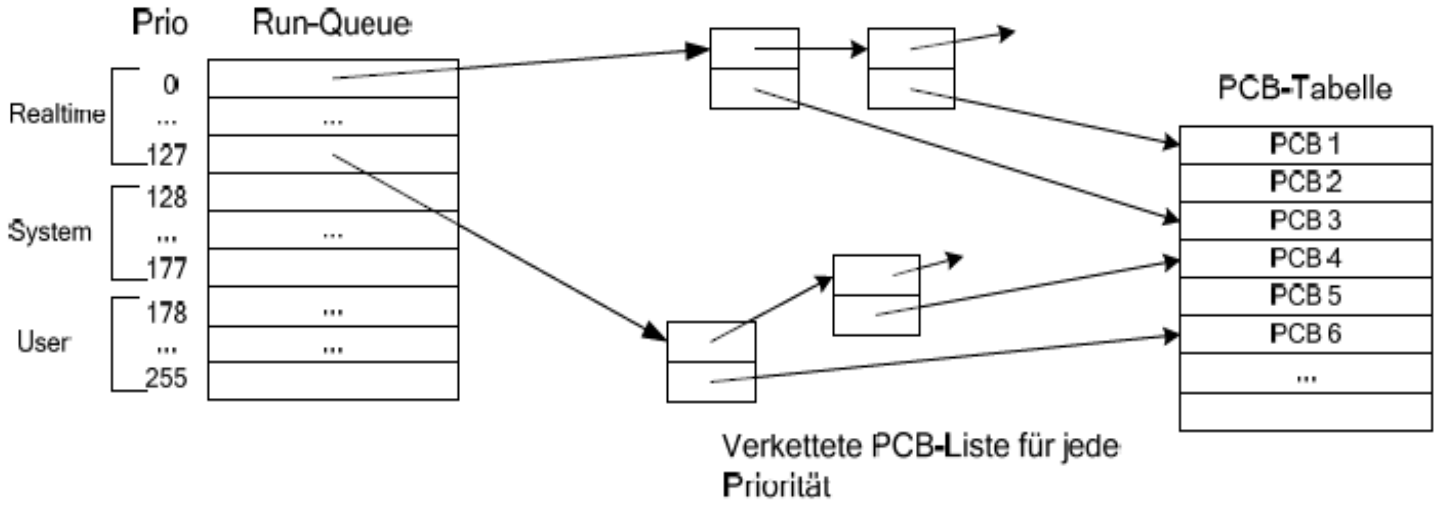
Scheduling-Strategien unter Unix

- Unterstützte Scheduling-Strategie:
  - **Preemptives** Verfahren
  - **Prozess als Scheduling-Einheit** in traditionellem Unix
  - **RR mit Prioritäten** und **Multi-Level-Feedback-Scheduling**
    - Eine Queue je Priorität
    - Round Robin innerhalb der gleichen Priorität
    - Vorderster Prozess aus aktuell höchster Priority-Queue wird als nächstes ausgewählt
    - Nach Ablauf der Zeitscheibe wird der Prozess ans Ende seiner aktuellen Queue oder ans Ende einer anderen Queue gehängt

Scheduling-Verfahren

Unix

- Ursprüngliche Unix Systeme nehmen RR mit Prioritäten-Steuerung
  - Jede Priorität hat eine eigene Queue (Run-Queue)
  - Verfahren ist verdrängend
  - Prozesse sind die Einheiten für den Scheduler
  - Prioritäten von 0 bis 255 (0 höchste Priorität)
  - Initiale Priorität ändert sich mit der Zeit dynamisch



Aufgabe Scheduling in realen Systemen

- Linux
  - O(1) Scheduler
  - Completely Fair Scheduler
- Windows
- Mac OS X

Erläutern Sie die Scheduling-Strategien und beantworten Sie folgende Fragen

- Welche Algorithmen verwenden die Scheduler?
- Wie werden die Prioritäten gesetzt?
- Aufbau der Run-Queue?

Ressourcenverwaltung

- Betriebsmittelverwalter
  - führt eine Freiliste für die freien Betriebsmittel-Einheiten
  - teilt anfordernden Prozessen die Betriebsmittel zu
  - nimmt von einem Prozess nicht mehr benötigte Betriebsmittel entgegen
  - fordert die restlichen bei Prozessbeendigung oder -abbruch zurück
  - versetzt er sie wieder in einen definierten Ausgangszustand

Aufgaben

- Erläutern Sie
  - Was versteht man unter einem Deadlock? Erläutern Sie dabei den Deadlock auch anhand eines Beispiels.
  - Was sind die Voraussetzungen, damit ein Deadlock eintreten kann.
  - Welche Strategien verwendet man zur Vermeidung von Deadlocks?
  - Welche Strategien verwendet man zur Deadlockbeseitigung?

Kommunikation von Prozessen und Threads

IPC (Interprocess Communication)

Kommunikationsformen:

- verbindungsorientiert versus verbindungslos
- speicherbasiert versus nachrichtenbasiert
- synchron versus asynchron
- halbduplex versus vollduplex
- Varianten der Empfängeradressierung

Wichtige Mechanismen

- Gemeinsame Nutzung von Datenbereiche
  - innerhalb eines Prozesses
  - über Shared Memory (prozessübergreifend)
  - über externe Speicherobjekte wie Dateien
- Kommunikation
  - Pipes und FIFOs
  - Message Queues
  - Sockets

Aufgaben Kommunikation

- Verbindungsorientiert versus verbindungslose Kommunikation
- Synchrone versus asynchrone Kommunikation
- Kommunikationskanal: Halbduplex- versus Vollduplex-Betrieb
- Varianten der Empfängeradressierung

Erklären Sie die Prinzipien!