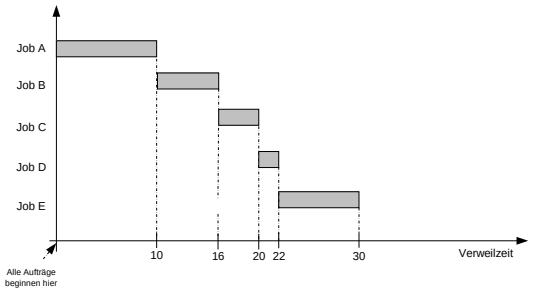
MAS: Betriebssysteme

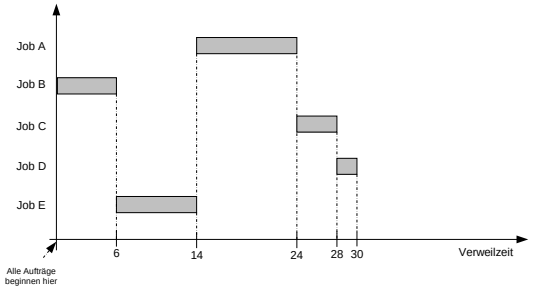
CPU-Scheduling - Grundkonzepte

T. Pospíšek

Gesamtüberblick

1. Einführung in Computersysteme
2. Entwicklung von Betriebssystemen
3. Architekturansätze
4. Interruptverarbeitung in Betriebssystemen
5. Prozesse und Threads
6. **CPU-Scheduling**
7. Synchronisation und Kommunikation
8. Speicherverwaltung
9. Geräte- und Dateiverwaltung
10. Betriebssystemvirtualisierung





Zielsetzung

- Der Studierende soll die wesentlichen CPU-Scheduling-Verfahren verstehen und erläutern können
- Insbesondere soll das Round-Robin-Verfahren mit und ohne Prioritätensteuerung skizziert und mit anderen Algorithmen verglichen werden können

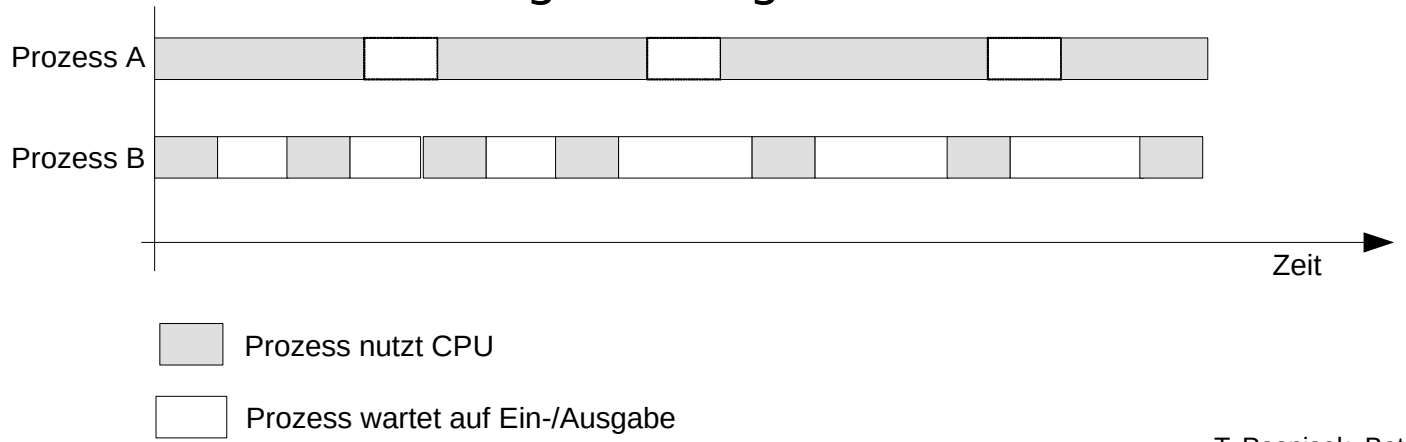
Scheduling = Ablaufplanung

Überblick

1. **Grundlagen**
2. Scheduling-Verfahren
3. Vergleich von Scheduling-Verfahren

Aspekte der CPU-Zuteilung

- CPU-Vergabe in Abhängigkeit der Scheduling-Einheit:
  - Prozess-basiert: CPU wird nur Prozessen zugeteilt
  - Thread-basiert: CPU wird Threads zugeteilt
  - Fiber: kooperatives Multitasking – Wechsel muss vom Programmierer explizit herbeigeführt werden
- CPU-Vergabe in Abhängigkeit des Prozesstyps:
  - Prozess A ist CPU-lastig (rechenintensiv)
  - Prozess B ist Ein-/Ausgabe-lastig

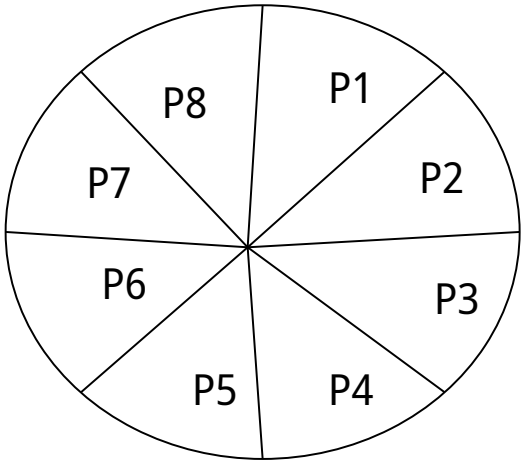


Der Prozessmanager

- **Scheduler**:
  - Komponente, welche die Planung der Betriebsmittelzuteilung (Scheduling = Ablaufplanung) übernimmt
- **Dispatcher**:
  - Komponente, die für den Prozesswechsel zuständig ist (Dispatching = Arbeitsverteilung, Disposition)
- Scheduler und Dispatcher sind logische Teile des Prozessmanagers

Das Zeitscheibenverfahren

- **Zeitscheibenverfahren** (Timesharing)
  - neu-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, Betriebssystem und Funktion/Konfig (Server, Desktop)
  - Zeitscheibe je CPU
- Beispiel: Ein Quantum von 10 ms bedeutet 100 Kontextwechsel pro Sekunde (Overhead vernachlässigt)

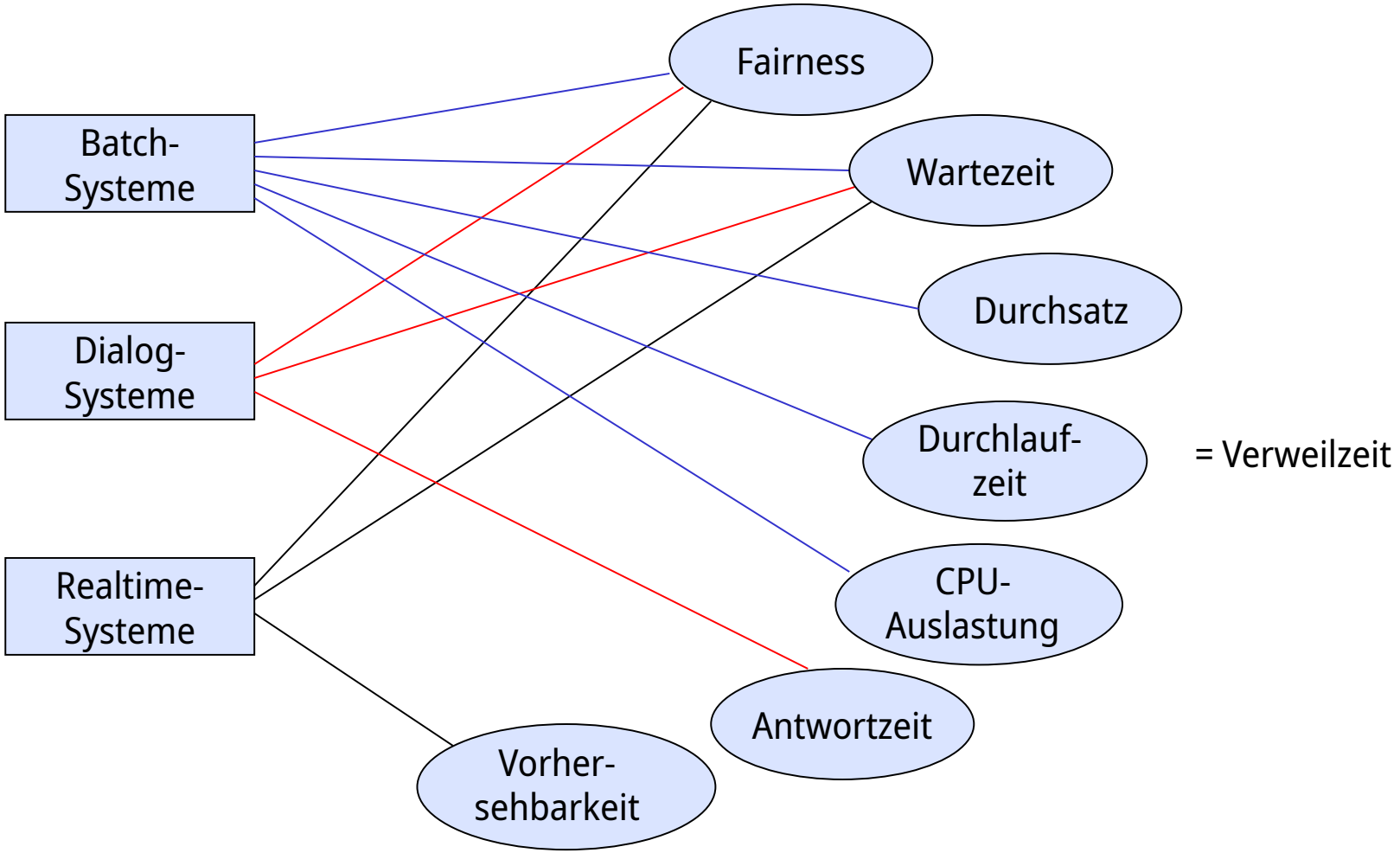


Zeitscheibe

Scheduling-Kriterien und -Ziele

- Für die Zuteilung von Zeitscheiben (Time slices) gibt es verschiedene Scheduling-Ziele:
  - **Fairness**:
    - Jeder Prozess erhält eine garantierte Mindestzuteilung
  - **Effizienz**:
    - Möglichst volle Auslastung der CPUs
  - **Antwortzeit**:
    - Soll minimiert werden
  - **Verweilzeit (Durchlaufzeit)**:
    - Wartezeit von Prozessen soll möglichst klein sein
  - **Durchsatz**:
    - Maximierung der Aufträge, die ein Betriebssystem in einem Zeitintervall durchführt
  - ...

Scheduling-Ziele je nach Betriebssystemtyp



Überblick

1. Grundlagen
2. **Scheduling-Verfahren**
3. Vergleich von Scheduling-Verfahren

Grundlegende Scheduling-Verfahren

- **Non-Preemptive Scheduling**, auch: Run-To-Completion-Verfahren (nicht verdrängend)
  - Altes Verfahren, bei dem ein Prozess nicht unterbrochen wird, bis er fertig ist
  - Nicht geeignet für konkurrierende Benutzer im Dialogbetrieb und auch nicht für die Echtzeitverarbeitung
  - Beispiel: MS-DOS

- **Preemptive Scheduling** (verdrängend)
  - Auch Vorrangunterbrechung genannt
  - Rechenbereite Prozesse werden suspendiert
  - Geeignet für konkurrierende Benutzer
  - Zeitscheibentechnik erforderlich

Scheduling-Algorithmen für Batch-Prozesse

- First Come First Served (**FCFS**)
  - Nach der Reihenfolge des Eintreffens
- Shortest Job First (**SJF**)
  - Prozess mit kürzester Bedienzeit als erstes
- Shortest Remaining Time Next (**SRTN**)
  - Prozess mit der kürzesten verbleibenden Restrechenzeit im System
- Bei manchen Strategien ist das „Verhungern“ von Prozessen (**Starvation**) möglich
  - Derartige Prozesse erhalten die CPU nie
  - Bei SJF z.B. dann, wenn viele kurze Prozesse ins System kommen → Verhungern der langen Jobs

Scheduling-Algorithmen in interaktiven Prozessen (1)

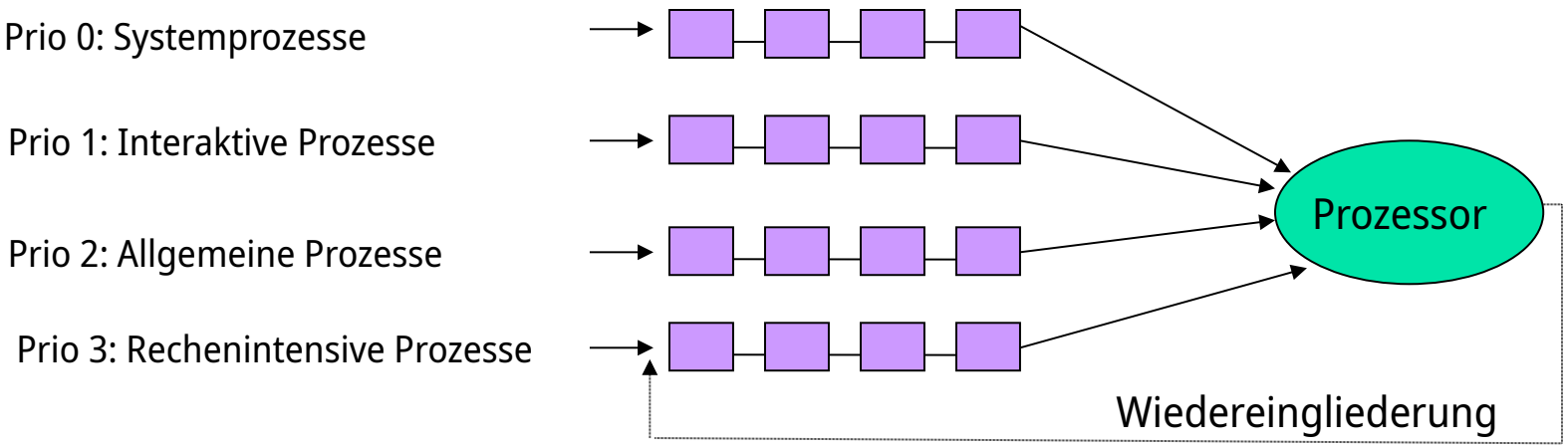
- Round-Robin-Scheduling (**RR**) = Rundlauf-Verfahren
  - FCFS in Verbindung mit Zeitscheibe
  - Leistung hängt von der Länge der Zeitscheibe ab
  - Wichtig: Kontext-Switch kostet auch Zeit
  - Verhältnis Arbeitszeit/Umschaltzeit darf nicht zu klein sein
    ➜ Durchsatz schlecht
  - Guter Wert für Zeitscheibe heute:
    ca. 10 bis 200 ms
- Priority Scheduling (**PS**)
  - Prozess mit höchster Priorität als nächstes
  - Dynamische (adaptive) und statische Prioritäten und Kombination möglich

Scheduling-Algorithmen in interaktiven Prozessen (2)

- Shortest Remaining Time First (**SRTF**)
- Lottery Scheduling
  - Zufällige Vergabe von CPU-Zeit
- Fair-Share-Scheduling
  - Gleichmäßige CPU-Nutzung durch Systembenutzer wird angestrebt
- Meist wird eine Kombination mehrerer Verfahren verwendet
  - Beispiel: RR mit Prioritäten

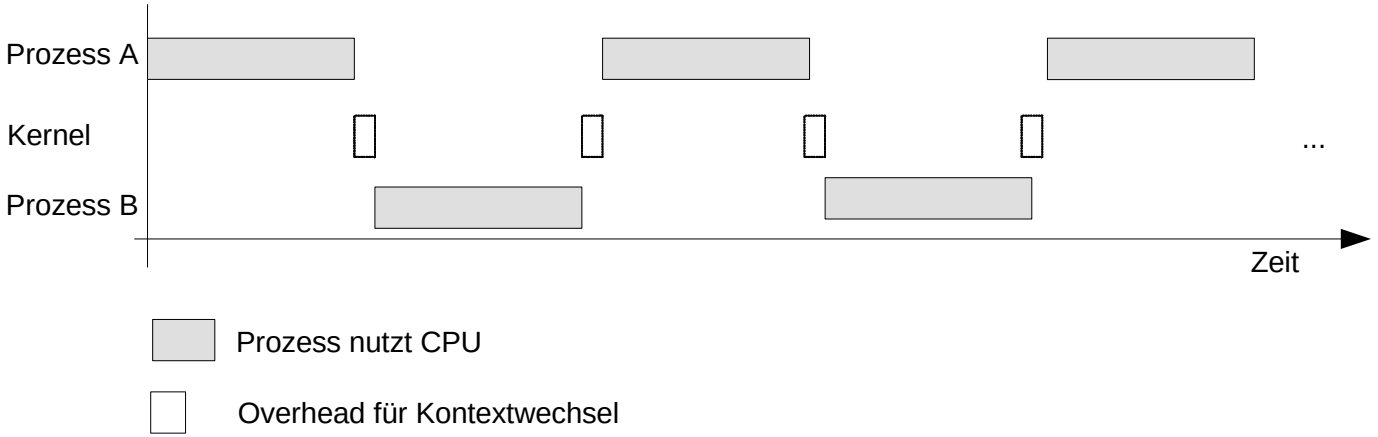
Multi-Level-Scheduling

- Multiple Warteschlangen: **Multi-Level Scheduling** mit mehreren Arten von Jobs unterschiedlicher Priorität
- **Multi-Level feedback Scheduling** bedeutet, dass ein Prozess auch in eine Warteschlange höherer (oder niedrigerer) Ordnung wechseln kann



Kombination: RR mit Prioritäten (1)

- **Frage 1:**
  - 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



Kombination: RR mit Prioritäten (2)

- **Frage 2:**
  - 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

Kombination: RR mit Prioritäten (3)

- Guter Ansatz in der Praxis: Dynamisches Verfahren
  - Initiale Festlegung des Quantums zum Prozessstartzeitpunkt
  - Dynamische Anpassung je nach Prozesstyp
  - Rechenintensive Prozesse → kürzeres Quantum
  - I/O-intensive Prozesse → längeres Quantum und/oder höhere Priorität
  - Prozess-Prioritäten werden statisch voreingestellt und unterliegen einer dynamischen Veränderung
  - In Kombination mit Multilevel-Feedback-Scheduling
- Voraussetzungen
  - Verwaltung der Prozessdaten *Quantum* und *Priorität* und eine zyklische Anpassung dieser erforderlich
  - Das Quantum der aktiven Prozesse wird *taktorientiert* herunter- gerechnet, Quantums-Neuberechnung bei vorgegebenen Ereignissen
  - Ebenso ist eine Neuberechnung der Prozess-Prioritäten bei definierten Ereignissen notwendig

Scheduling in Realtime-Systemen

- Schnelle und berechenbare Reaktionen bei anstehenden Events gefordert
- Man unterscheidet **hard real time** Systeme und **soft real time** Systeme
  - Erstere müssen schnell reagieren
  - Bei letzteren ist eine Verzögerung möglich
- Man unterscheidet **statische** und **dynamische** Scheduling-Algorithmen
  - Bei ersteren ist die Scheduling-Entscheidung bereits vor dem Start des Systems bekannt
  - Letztere treffen ihre Scheduling-Entscheidungen zur Laufzeit

Überblick

1. Grundlagen
2. Scheduling-Verfahren
3. **Vergleich von Scheduling-Verfahren**

Beispiel zu Scheduling-Algorithmen

- 5 Aufträge A – E treffen im System fast gleichzeitig ein
- Geschätzte Ausführungszeiten in Millisekunden: A=10, B=6, C=4, D=2, E=8
- Prioritäten: A=3, B=5, C=2, D=1, E=4 (5 ist höchste)
- Der Rechner verfügt über eine CPU (einen Rechnerkern)
- Gesucht: Durchschnittliche Verweilzeit im System bei
  - A) Priority Scheduling (nicht verdrängend)
  - B) FCFS (Reihenfolge-Annahme: A, B, D, C, E) (nicht verdrängend)
  - C) Shortest Job First (nicht verdrängend)
  - D) RR mit Prioritäten bei einem Quantum von 2 ms (Theorie!)
- Prozesswechselzeit wird vernachlässigt
- Aufträge werden nacheinander ausgeführt, außer bei D)

Beispiel zu Scheduling-Algorithmen: Priority Scheduling (1)

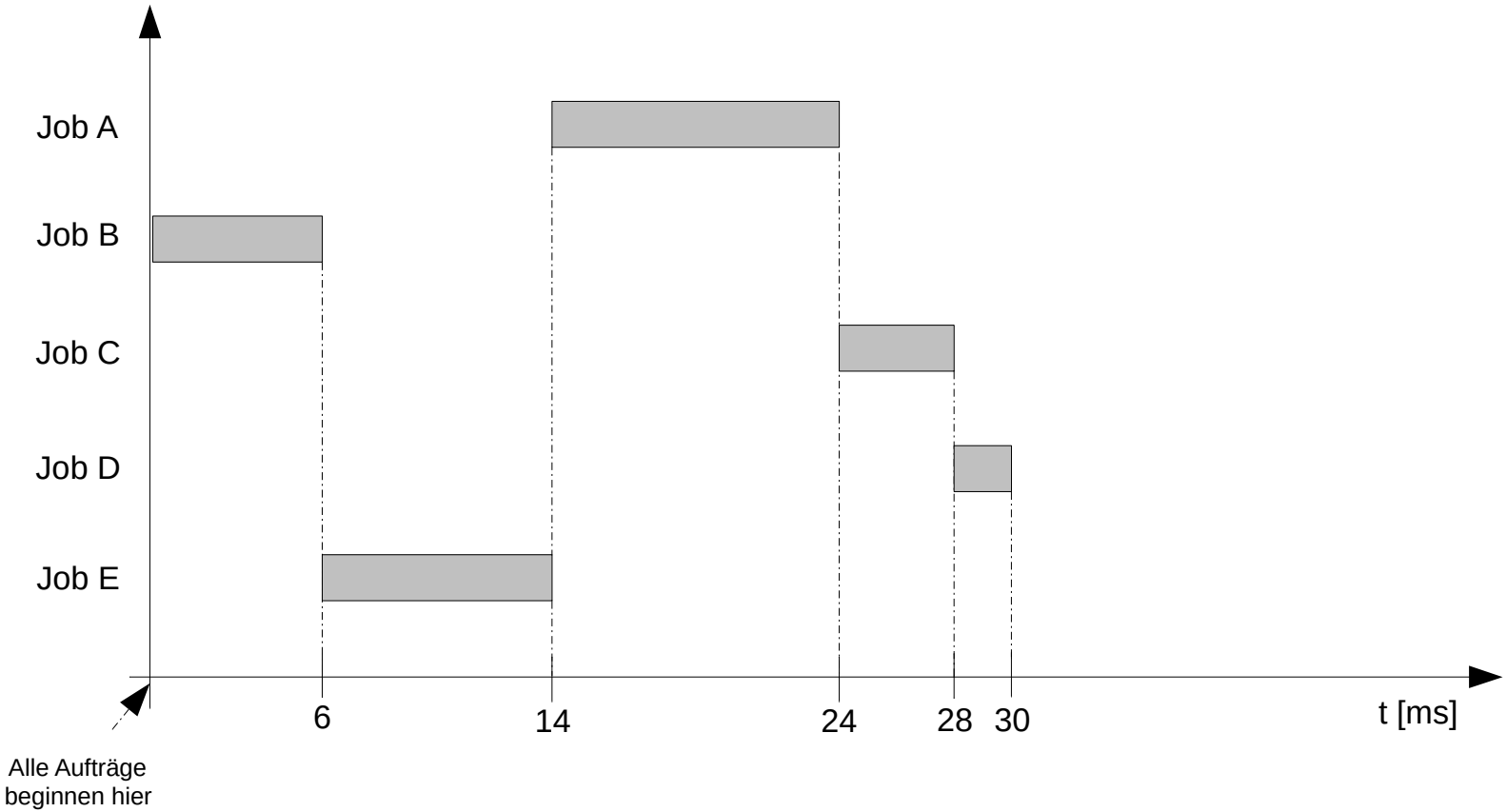
Job	A	B	C	D	E
Zeit	10	6	4	2	8
Priorität	3	5	2	1	4

- Ablauf mit Priority Scheduling: 5 ist die höchste Priorität!

Job	B	E	A	C	D
Verweil-zeit	6	14	24	28	30

- Summe: 102
- Mittlere Verweilzeit = 102 / 5 = **20.4** ms

Beispiel zu Scheduling-Algorithmen: Priority Prioritäten (2)



Beispiel zu Scheduling-Algorithmen: FCFS (1)

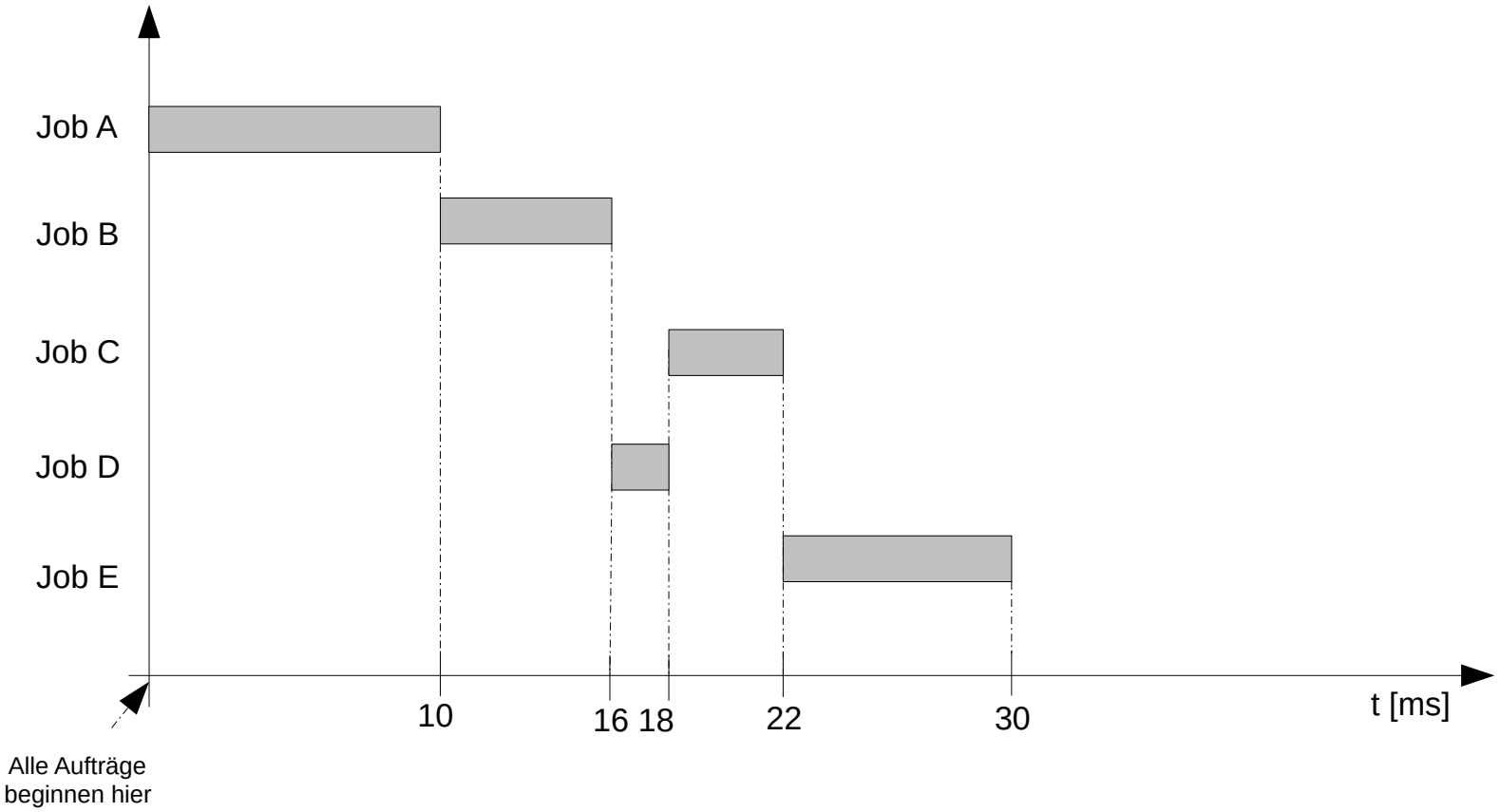
Job	A	B	C	D	E
Zeit	10	6	4	2	8
Priorität	3	5	2	1	4

- Ablauf mit FCFS: Reihenfolge-Annahme: A, B, D, C, E

Job	A	B	D	C	E
Verweil-zeit	10	16	18	22	30

- Summe: 96
- Mittlere Verweilzeit = 96 / 5 = **19.2** ms

Beispiel zu Scheduling-Algorithmen: FCFS (2)



Beispiel zu Scheduling-Algorithmen: Shortest Job First (1)

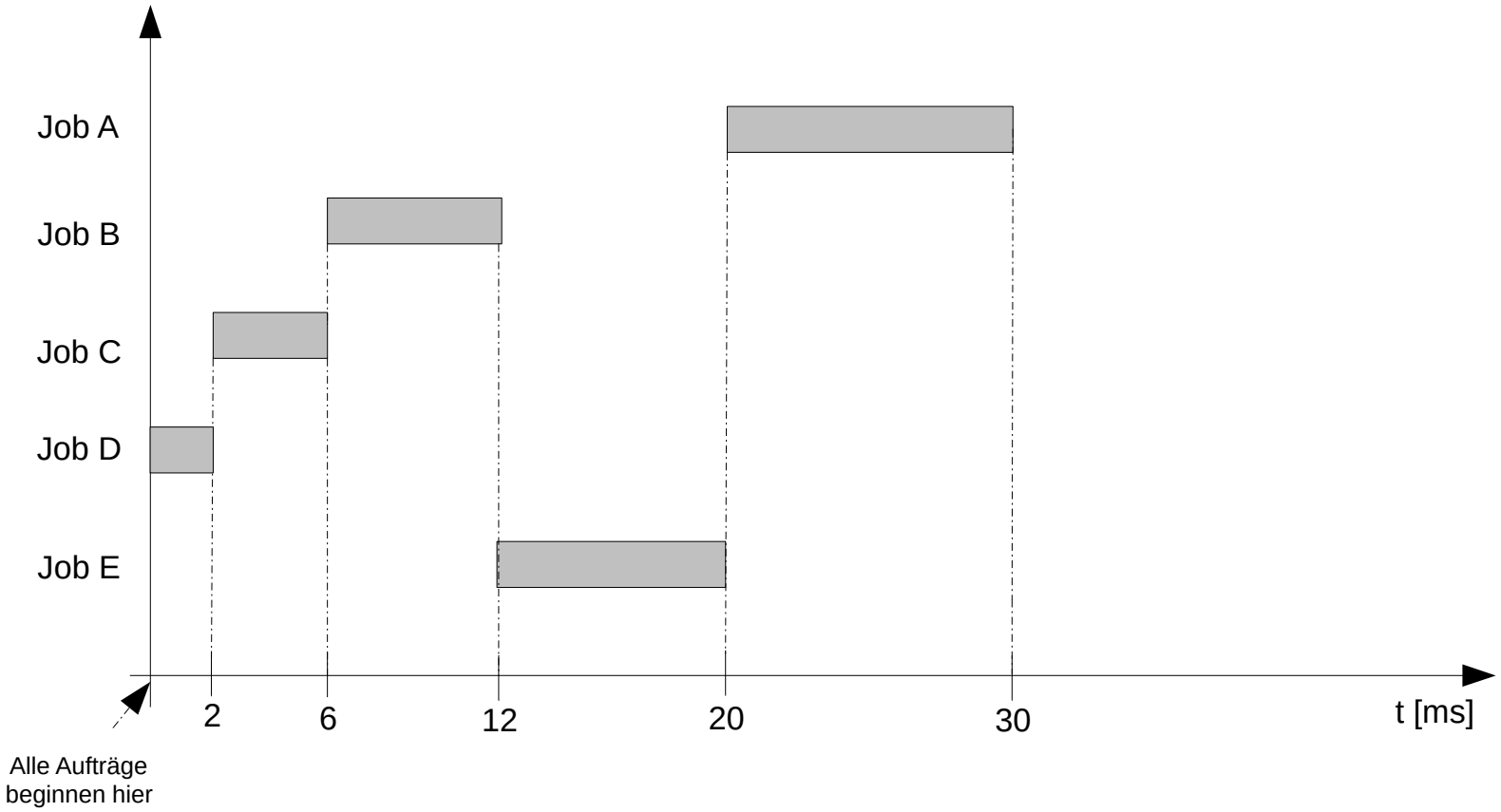
Job	A	B	C	D	E
Zeit	10	6	4	2	8
Priorität	3	5	2	1	4

- Ablauf mit Shortest Job First

Job	D	C	B	E	A
Verweil-zeit	2	6	12	20	30

- Summe: 70
- Mittlere Verweilzeit = 70 / 5 = **14.0** ms

Beispiel zu Scheduling-Algorithmen: Shortest Job First (2)



Beispiel zu Scheduling-Algorithmen: RR mit Prioritäten (1)

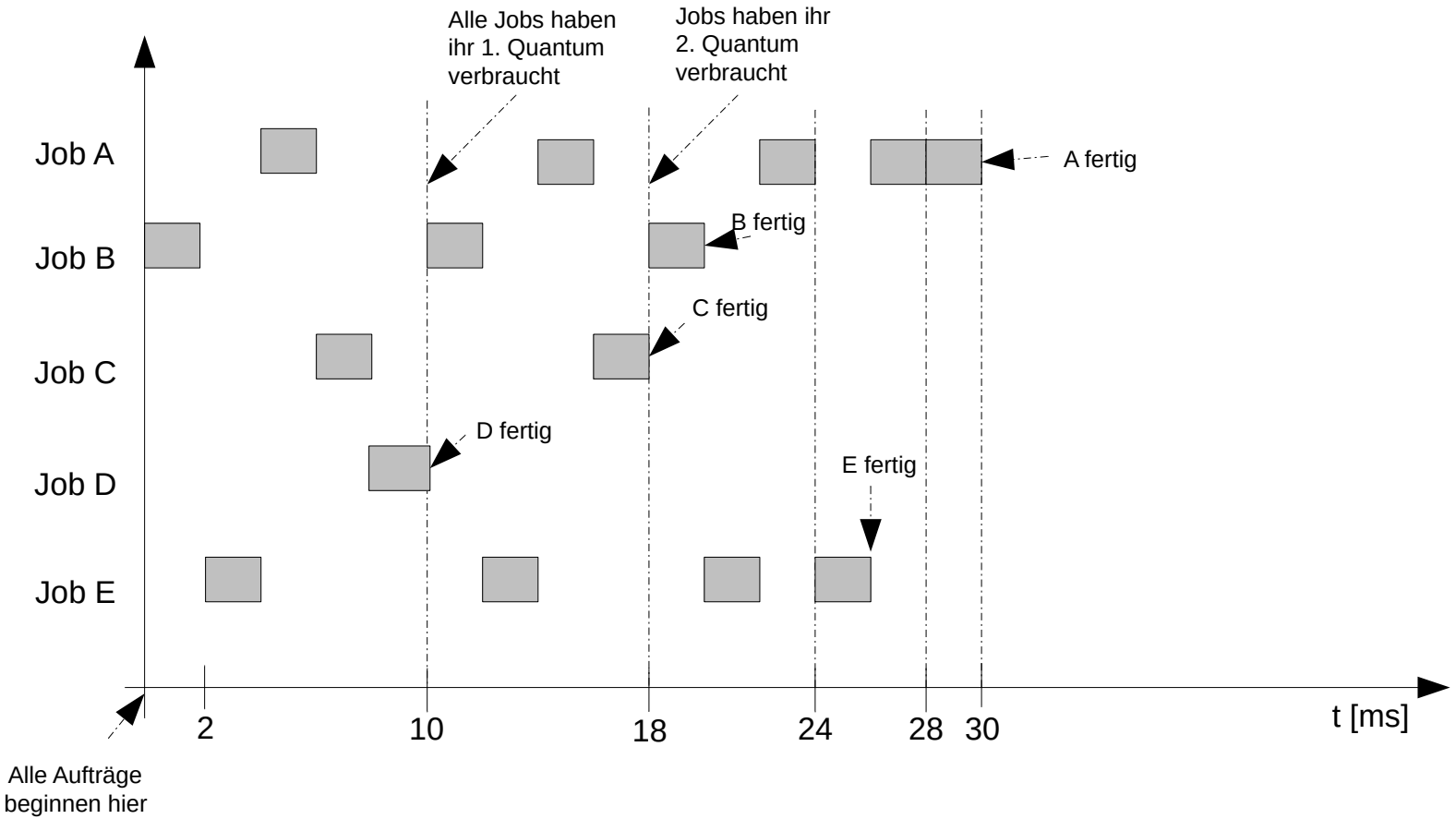
Job	A	B	C	D	E
Zeit	10	6	4	2	8
Priorität	3	5	2	1	4

- Ablauf bei RR mit Prioritäten und einem Quantum von 2 ms

Job	D	C	B	E	A
Verweil-zeit	10	18	20	26	30

- Summe: 104
- Mittlere Verweilzeit = 104 / 5 = **20,8** ms

Beispiel zu Scheduling-Algorithmen: RR mit Prioritäten (2)



Beispiel zu Scheduling-Algorithmen: Resumé

- Shortest-Job-First ist am besten, wenn man nur die Verweilzeit (Durchlaufzeit) betrachtet
- Beweis: a, b, c, d, e sind die Ausführungszeiten der Jobs A, B, C, D, E:

  $V_{all} = a + (a+b) + (a+b+c) + (a+b+c+d) + (a+b+c+d+e) = 5a + 4b + 3c + 2d + e$

  Die durchschnittliche Verweilzeit ergibt sich dann aus

  $V_{avg} = (5a + 4b + 3c + 2d + e) / 5$

  → Job A trägt am meisten zu V_{all} und V_{avg} bei und sollte daher am kürzesten sein
- SJF ist aber unrealisierbar, da die benötigte Prozessorzeit initial nicht bekannt ist

Übung zu Scheduling-Algorithmen

- optional – siehe Folien `optional/05-3_CPU_Scheduling_Grundkonzepte_Uebung.odp`

Zusammenfassung, Wichtige Aspekte

- Scheduling-Kriterien:
  - Durchsatz, Fairness, CPU-Auslastung, ...
- Scheduling-Algorithmen:
  - FCFS, RR, SJF, Prioritäten,...
- SJF ist der optimale Algorithmus (Theorie)
- Jedes Betriebssystem hat eigene Strategie, meist wird aber eine Variante mit RR, Prioritäten und Multi-Level-Queues implementiert
  - Prioritäten und Quanten sind entscheidende Parameter
- CPU-Scheduling ist meist ein komplexes Verfahren, aber es gibt auch einfache Ansätze → Linux CFS (kommt noch)

Gesamtüberblick

- ✓ Einführung in Computersysteme
- ✓ Entwicklung von Betriebssystemen
- ✓ Architekturansätze
- ✓ Interruptverarbeitung in Betriebssystemen
- ✓ Prozesse und Threads
- ✓ CPU-Scheduling (es folgen Fallbeispiele)

7. Synchronisation und Kommunikation
8. Speicherverwaltung
9. Geräte- und Dateiverwaltung
10. Betriebssystemvirtualisierung