Angewandte Computer- und Biowissenschaften

HOCHSCHULE MITTWEIDA
University of Applied Sciences

Rechner- und Betriebssysteme

Vorlesung Betriebssysteme

Vorlesung 05 – Betriebsmittelverwaltung

Prof. Dr. Tobias Czauderna

Folien basieren auf Folien von Prof. Dr. Uwe Schneider

Vorlesung

1. Grundlagen
2. Architektur
3. Prozesse
4. Koordinierung paralleler Prozesse
5. **Betriebsmittelverwaltung**
6. Verklemmungen
7. Speicherverwaltung
8. Ein-/Ausgabe
9. Dateiverwaltung/-systeme
10. Schutz und Sicherheit

Betriebsmittelverwaltung

- Klassifikation von Betriebsmitteln (Ressourcen)
- Verwaltung von Betriebsmitteln
- Scheduling (mit Beispielen für verschiedene Verfahren)
- Zusammenfassung

Klassifikation von Betriebsmitteln

- Zugehörigkeit zu einer System-Schicht
  - **Hardware**: Drucker, Hauptspeicher, …
  - **Software**: Programmcode, Dateien, Nachrichten, …
- Wiederverwendbarkeit
  - **Wiederverwendbar**: von verschiedenen Prozessen nacheinander benutzbar (Prozessor, Dateien, …)
  - **Nicht wiederverwendbar**: verbrauchbar (Signale, Nachrichten, …)

Klassifikation von Betriebsmitteln

- Entziehbarkeit/Verdrängbarkeit
  - **Entziehbar**: falls die „Werte“ („Inhalt“) des Betriebsmittels mit vertretbarem Aufwand gerettet und wiederhergestellt werden können (Prozessor, Hauptspeicher-Bereiche, ...)
  - **Nicht entziehbar**: falls Rettung/Wiederherstellung der „Werte“ unmöglich oder zu aufwändig (verbrauchbare Betriebsmittel, eventuell große Dateien, Festplatten, ...)

Klassifikation von Betriebsmitteln

- Exklusivität der Nutzung
  - **Exklusiv nutzbar**: zu einem Zeitpunkt nur durch maximal einen Prozess benutzbar (Prozessor, Drucker, ...)
  - **Mehrfach nutzbar**: zu einem Zeitpunkt durch mehrere Prozesse parallel benutzbar (Hauptspeicher, Festplatte, manche Dateien, ...)

Verwaltung von Betriebsmitteln

- Lokale Betriebsmittel
  - Gehören ständig nur jeweils einem Prozess
  - Für Aspekte des gesamten Systems unwichtig
  - Keine Verwaltung im Betriebssystem nötig
- Globale Betriebsmittel
  - Stehen allen Prozessen jederzeit zur parallelen Nutzung zur Verfügung
  - Für den Zustand des gesamten Systems wichtig
  - Aber trotzdem keine Verwaltung im Betriebssystem nötig

Verwaltung von Betriebsmitteln

- Exklusiv nutzbare Betriebsmittel
  - Können jederzeit nur durch maximal einen Prozess (sinnvoll) genutzt werden
  - Verwaltung im Betriebssystem zwingend notwendig

- Notwendige Schritte
  1. Betriebsmittel vom Betriebssystem anfordern
  2. Betriebsmittel vorübergehend (lokal) nutzen
  3. Betriebsmittel an das Betriebssystem zurückgeben

Verwaltung von Betriebsmitteln

- Beispiele für geeignete Mittel (z.T. bereits bekannt)
  - Binäre Semaphore
  - Allgemeine Semaphore
  - Andere Koordinierungsmittel (z.B. Dienstleistungsprozesse)

- *Hinweis*: einige Betriebsmittel werden im Betriebssystem durch spezielle Komponenten verwaltet, z.B. Hauptspeicher (siehe Speicherverwaltung), Dateien (siehe Dateiverwaltung)

Scheduling

- Scheduling = Ablaufplanung
  - Entscheidung des Betriebssystems bzgl. Zuteilung von Betriebsmitteln an Prozesse
  - Die Auswahl eines geeigneten Prozesses erfolgt i. Allg. anhand einer Zuteilungsstrategie durch den **Scheduler**
  - Ist zentrale Aufgabe von Betriebssystemen
  - Kann auf mehreren Ebenen bzw. für verschiedene Betriebsmittel erfolgen
  - Dient der Optimierung des Systemverhaltens anhand einer Zielfunktion

Scheduling

- Scheduler
  - Spezielle Komponente des Betriebssystem-Kerns zur Ablaufplanung
- Dispatcher
  - Spezielle Komponente zur Realisierung des Prozess-Wechsels

Scheduling

- Wann erfolgt die Einplanung?
  - „Echte Planung“ der Zuteilungsreihenfolge
    - Praktisch nur möglich, wenn der Ressourcenbedarf der Prozesse vorab bekannt ist, z.B. im Stapelbetrieb für Jobs oder für periodische Prozesse
    - **Statische Betriebsmittel-Verwaltung**: Prozess erhält alle benötigten Betriebsmittel bei seiner Erzeugung vom Betriebssystem zugeteilt und gibt sie bei seiner Beendigung wieder zurück

Scheduling

- In allen anderen Fällen: operatives („ad hoc“) Scheduling
  - **Dynamische Betriebsmittel-Zuteilung** (abhängig von aktueller Lastsituation): Prozess fordert Betriebsmittel erst bei Bedarf an, benutzt es vorübergehend exklusiv und gibt es i. Allg. danach wieder zurück (*Achtung*: Verklemmungsgefahr)

Scheduling

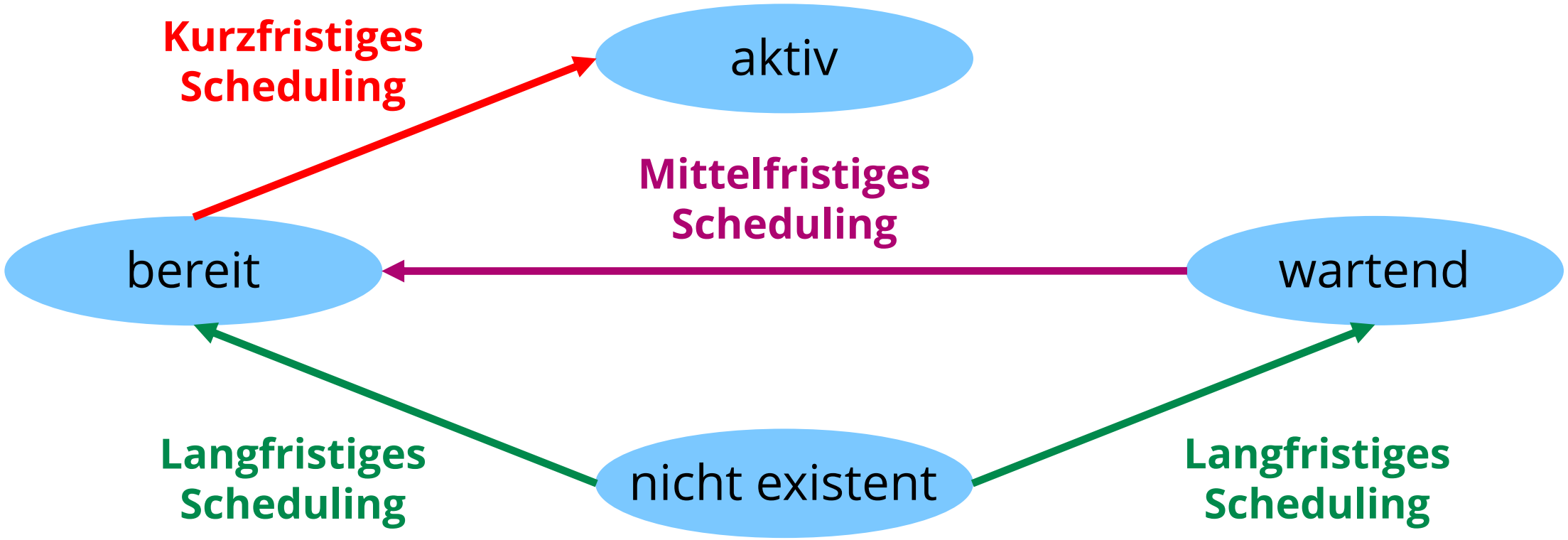
- „Planungsphasen“
  - **Langfristiges Scheduling** (long term scheduling)
    - Entscheidung über Annahme eines Auftrags am System
    - Nur einmalig zu Beginn eines Prozesses (z.B. begrenzte Zahl von Benutzerzugängen an Servern, Job Control bei Stapelbetrieb)
  - **Mittelfristiges Scheduling** (medium term scheduling)
    - (z.B.) Entscheidung über Wieder-Einlagerung von Prozessen (Swapping, Speicherverwaltung)
    - Tritt hin und wieder auf

Scheduling

- **Kurzfristiges Scheduling** (short time scheduling)
  - Entscheidung über vorübergehende CPU-Zuteilung an „den nächsten“ Prozess/Thread (= Aktivierung)
  - Tritt sehr oft im Leben eines Prozesses/Threads auf

Scheduling

- Zusammenhang zum Zustandsmodell



Scheduling

- Mögliche Ziele (Kriterien)
  - Fairness gegenüber allen Prozessen/Threads
  - Bevorzugung dringender Prozesse
  - Effizienz, maximale Auslastung der Ressourcen, z.B. CPU
  - Maximaler Durchsatz (= Aufträge pro Zeit)
  - Minimale Antwortzeit (insbesondere im Dialogbetrieb)
  - Minimale Reaktionszeit (insbesondere im Echtzeitbetrieb)
  - Minimale Wartezeit
  - Minimale Ausführungszeit
- Was ist optimal? -> Ziele widersprechen sich zum Teil

Scheduling

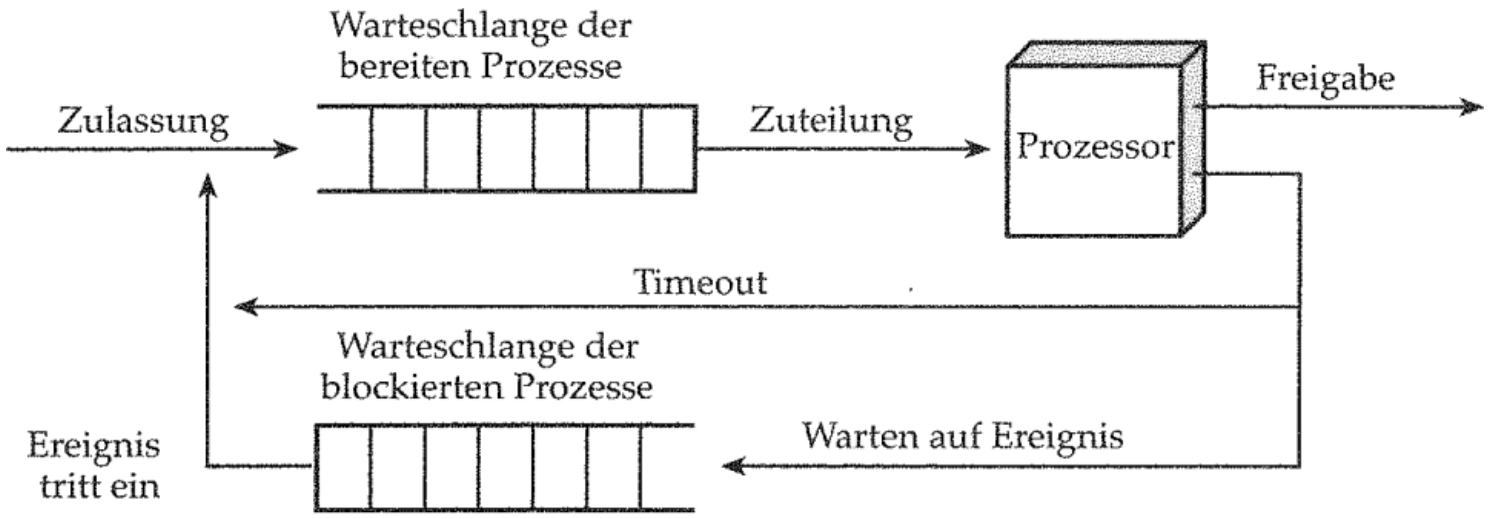
- „Mechanismus“ und „Strategie“
  - Trennung zwischen beiden Begriffen ist sinnvoll, weil es für die verschiedenen Benutzerforderungen keinen einheitlichen idealen Scheduler für jede Situation gibt
  - Mechanismen zum Scheduling
    - (z.B.) Art und Weise des Umordnens eines Prozesses aus einer Zustandsliste (Warteliste) in eine andere (siehe Prozess-Zustände)
    - Sind i. Allg. im Betriebssystem-Kern festgelegt

Scheduling

- Strategien zum Scheduling
  - Konkrete Algorithmen/Verfahren zur Auswahl eines Prozesses aus einer Liste („Welcher Prozess ist der Nächste?“)
  - Können aus dem Betriebssystem ausgegliedert sein (und sind damit ggf. direkt vom Benutzer änderbar)

Scheduling

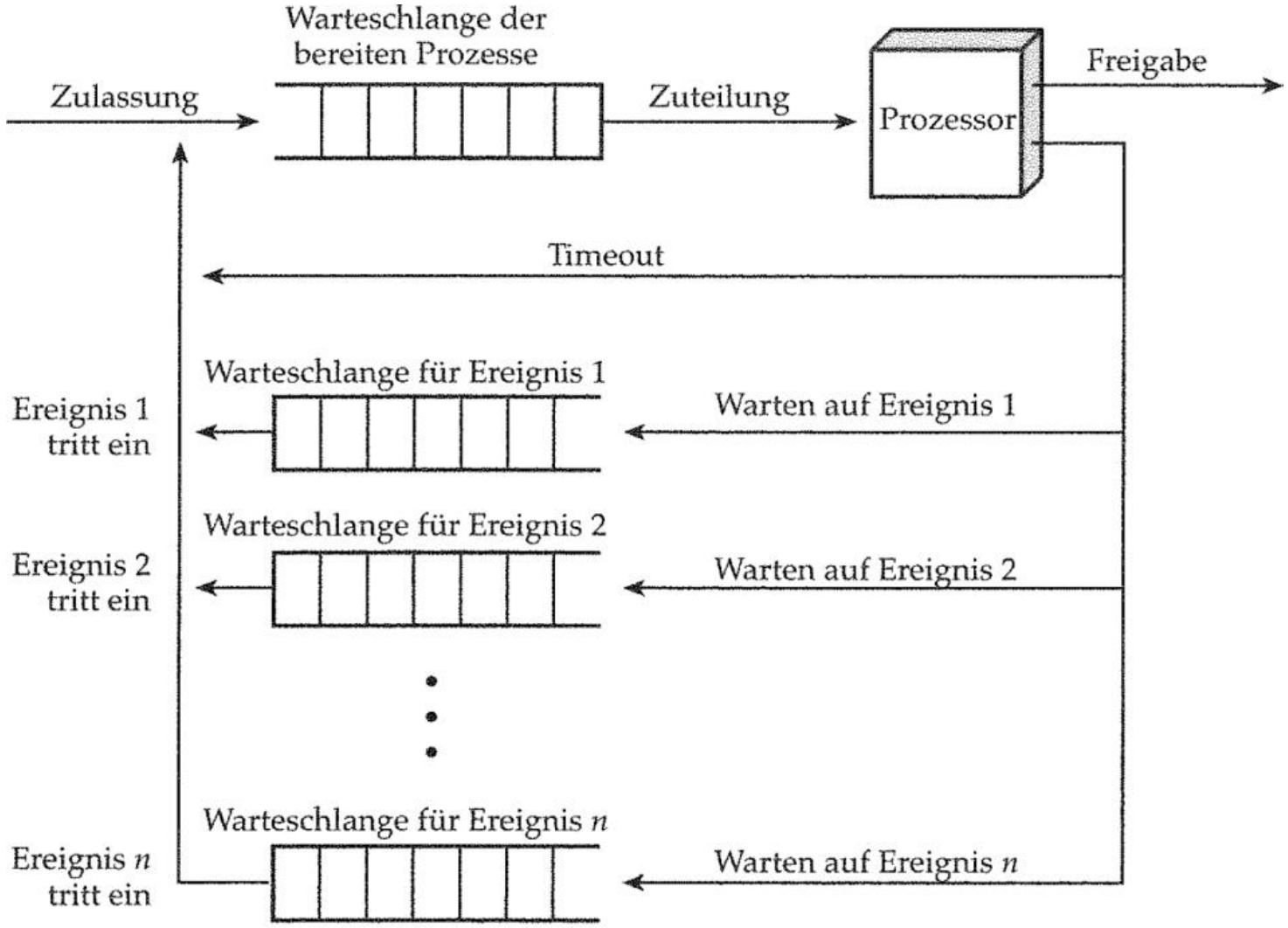
- „Bedienungsmodell“



(a) Eine Warteschlange mit blockierten Prozessen

Quelle: W. Stallings: Betriebssysteme – Prinzipien und Umsetzung. Pearson Studium, 2003

Scheduling



(b) Mehrere Warteschlangen mit blockierten Prozessen

Quelle: W. Stallings: Betriebssysteme – Prinzipien und Umsetzung. Pearson Studium, 2003

Scheduling

- Scheduling-Strategien (für Prozessor-Zuteilung)
  - Nicht präemptiv, „nicht verdrängend“, ohne Entzug (non-preemptive, run-to-completion)
    - Der Prozess, der das Betriebsmittel besitzt, entscheidet selbst, wann er es (freiwillig) abgibt

Scheduling

- Präemptiv, „verdrängend“, mit Entzug (preemptive)
  - Der aktive (Betriebsmittel-besitzende) Prozess kann an einer beliebigen Stelle vom Betriebssystem unterbrochen werden, er wird verdrängt, ihm wird das Betriebsmittel entzogen
  - Mögliche Auslöser für die Verdrängung
    - Zeitsteuerung (zyklisch, Zeitüberschreitung, festgelegter Zeitpunkt)
    - Ereignissteuerung (z.B. Interrupts, Signale, API-Funktionsaufrufe)

Scheduling

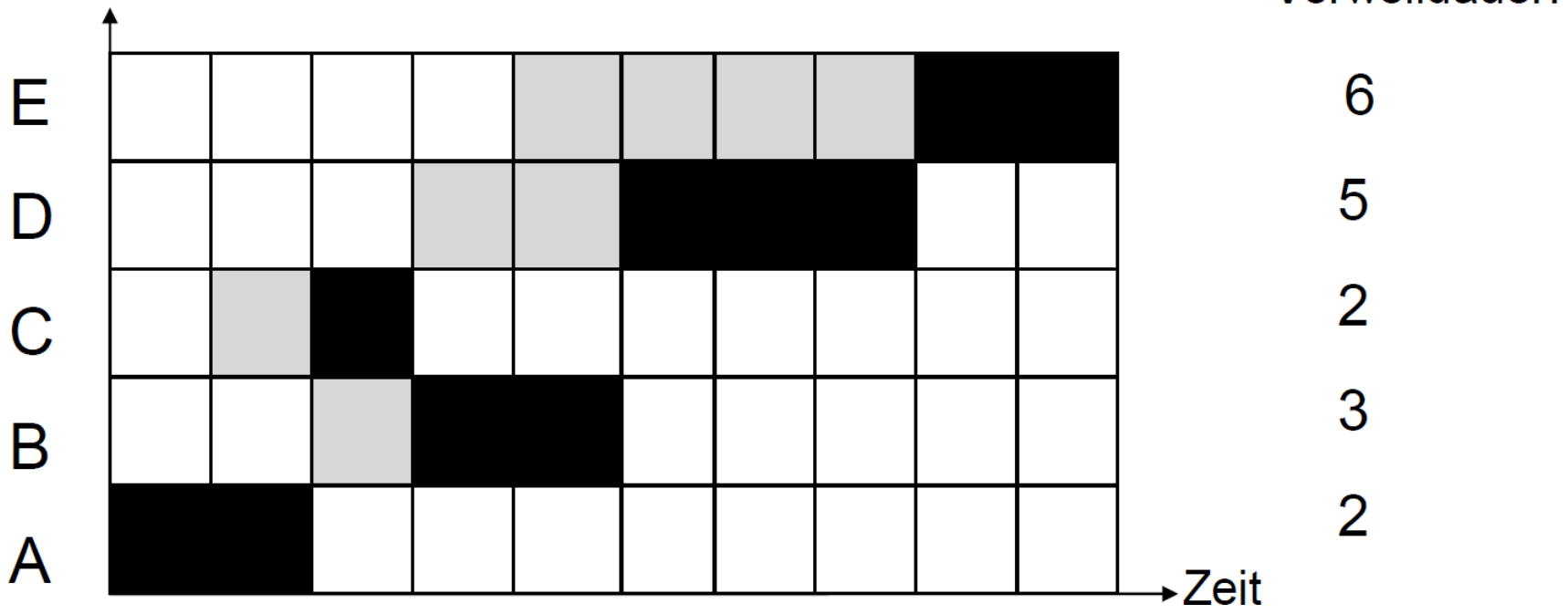
- First Come First Serve(ed) (FCFS)
  - Zuteilung in der Reihenfolge des Eintreffens der Prozesse
  - Kriterium: Ankunftszeit
    - „Wer zuerst kommt, wird zuerst bedient“ (FIFO-Warteschlange)
  - Nicht präemptiv
  - Voraussetzung: Scheduler kennt Ankunftszeit und (voraussichtliche) Bearbeitungsdauer jedes Prozesses
  - Geringer Overhead, aber Leistungsfähigkeit begrenzt
  - Ggf. Benachteiligung kurzer Prozesse, falls sie unmittelbar nach einem langen Prozess eintreffen
  - Anwendung z.B. bei Stapelbetrieb

Scheduling

Beispiel

Prozess	A	B	C	D	E
Ankunftszeit	0	2	1	3	4
Bearbeit.-dauer	2	2	1	3	2

Ankunftsreihenfolge: A C B D E



Verweildauer:

6
5
2
3
2

Scheduling

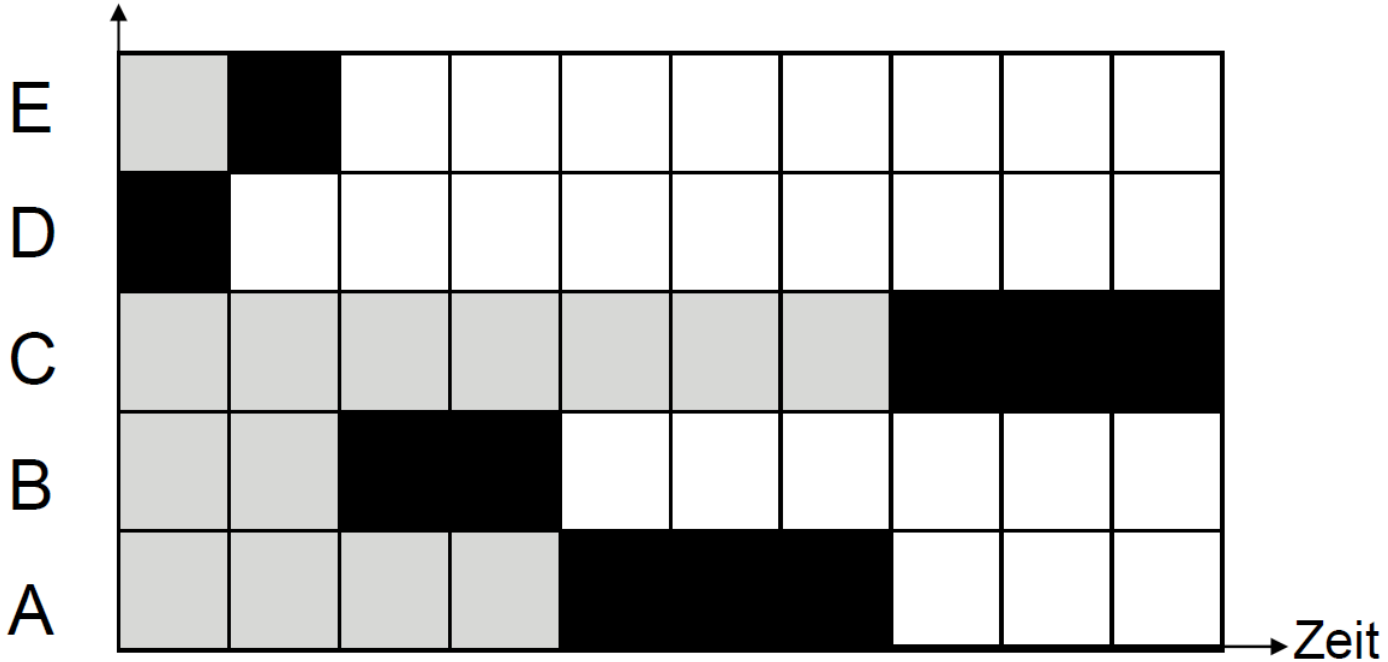
- Shortest Job Next (SJN), Shortest Job First (SJF)
  - Zuteilung an den Prozess mit der jeweils kürzesten (erwarteten) Bearbeitungsdauer
  - Nicht präemptiv
  - Voraussetzung: Scheduler kennt Ankunftszeit und (voraussichtliche) Bearbeitungsdauer jedes Prozesses
  - Alternative dazu: Bearbeitungsdauer schätzen, Erfahrungswerte
  - Mittlere Verweildauer -> Minimum (falls alle Prozesse zu Beginn vorliegen), Gegenteil: Largest/Longest Job Next/First
  - Falls es viele kurze Prozesse gibt: Gefahr des „Verhungerns“ von langen Prozessen (starvation)
  - Anwendung z.B. bei Stapelbetrieb, Transaktionssysteme

Scheduling

Beispiel

Prozess	A	B	C	D	E
Ankunftszeit	0	0	0	0	0
Bearbeit.-dauer	3	2	3	1	1

Kürzeste Bearbeitungsdauer: z.B. D E B A C



Verweildauer:

2

1

10

4

7

$tv_{mittl}=4,8$

Scheduling

- Highest Response Ratio Next (HRRN)
  - Zuteilung anhand des Quotienten $\frac{\text{Bearbeitungsdauer} + \text{Wartedauer}}{\text{Bearbeitungsdauer}}$
  - = „normalisierte Antwort-/Durchlaufzeit“
  - Nicht präemptiv
  - Voraussetzung: Scheduler kennt Ankunftszeit und (voraussichtliche) Wartezeit sowie Bearbeitungsdauer jedes Prozesses (z.B. durch Schätzungen auf Basis vorheriger Messungen)

Scheduling

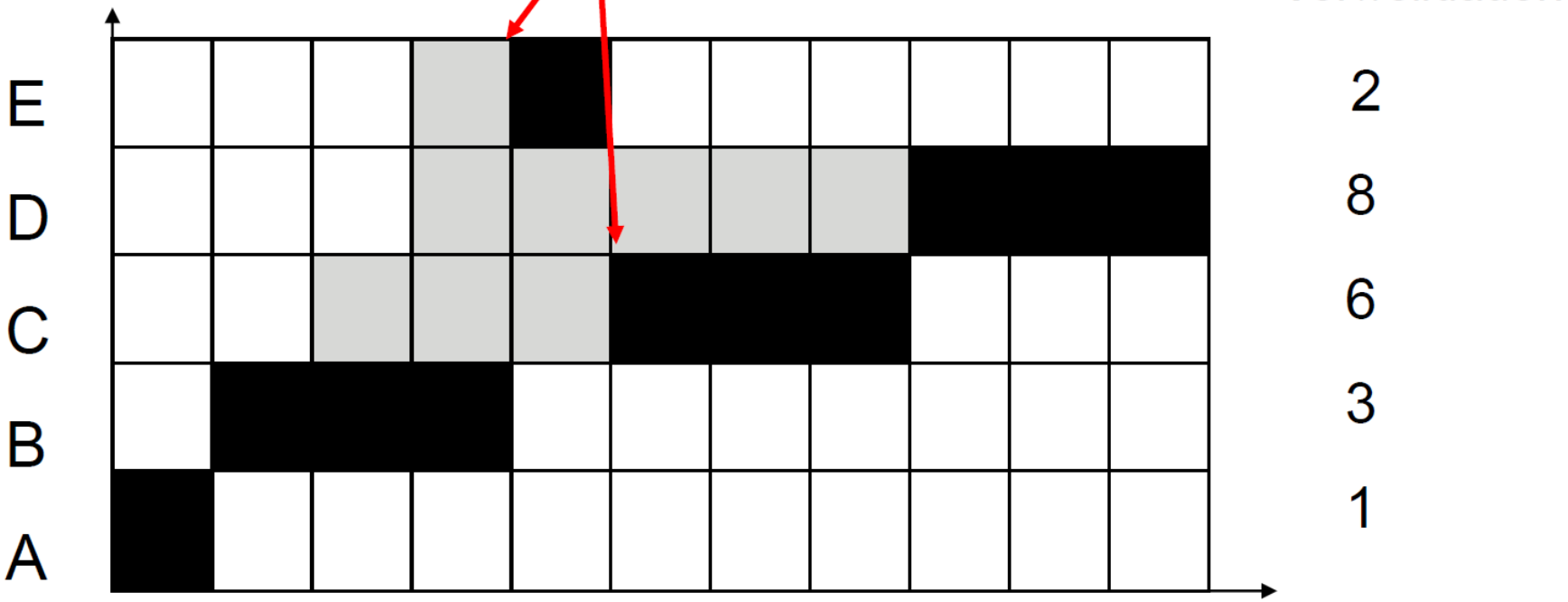
- Bevorzugung von Prozessen mit kurzer Bearbeitungsdauer, aber auch Begrenzung der Wartezeit von Prozessen mit langer Bearbeitungsdauer
- Gute Lastverteilung
- Anwendung z.B. bei Stapelbetrieb

Scheduling

Beispiel

Prozess	A	B	C	D	E
Ankunftszeit	0	1	2	3	3
Bearbeit.-dauer	1	3	3	3	1

Antwortquotient zum Zeitpunkt 4: E (2,0) C (1,7) D (1,3)
Antwortquotient zum Zeitpunkt 5: C (2,0) D (1,7)



Scheduling

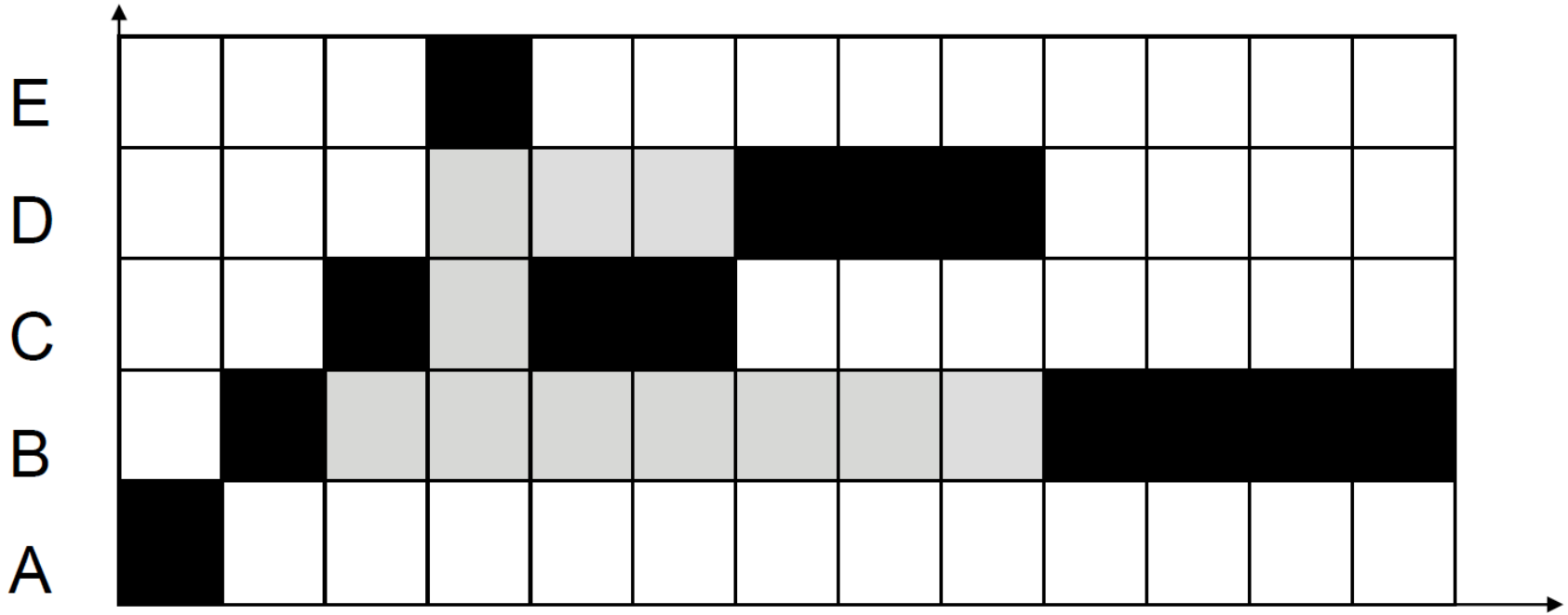
- Shortest Remaining (Processing) Time (SRT)
  - Zuteilung an den Prozess mit der voraussichtlich kürzesten noch verbleibenden Bearbeitungszeit (Restbearbeitungsdauer)
  - Präemptiv
  - Voraussetzung: Scheduler kennt (voraussichtliche) Bearbeitungsdauer jedes Prozesses (z.B. durch Schätzungen auf Basis vorheriger Messungen)
  - Bevorzugung von neuen Prozessen mit kurzer Bearbeitungszeit
  - „Verhungern“ von (langandauernden) Prozessen möglich
  - Gute Antwortzeit
  - Anwendung z.B. bei Stapelbetrieb

Scheduling

Beispiel

Prozess	A	B	C	D	E
Ankunftszeit	0	1	2	3	3
Bearbeit.-dauer	1	5	3	3	1

Unterbrechungen: zum Zeitpunkt 2 (B) und 3 (C)



Scheduling

- Round Robin (RR)
  - Zeitscheibenverfahren, „rundenbasiert“, Time Slice Verfahren
  - Zuteilung der CPU für ein bestimmtes Zeitquantum (Zeitscheibe, Δt) ohne Bevorzugung reihum an jeden (laufbereiten) Prozess
  - Präemptiv: Unterbrechung des aktiven Prozesses (spätestens) nach Ablauf von Δt durch Timer-Interrupt
    - Unterbrochener Prozess kommt dadurch wieder in den Zustand „bereit“ und aus der „bereit-Liste“ wird der nächste Prozess aktiviert

Scheduling

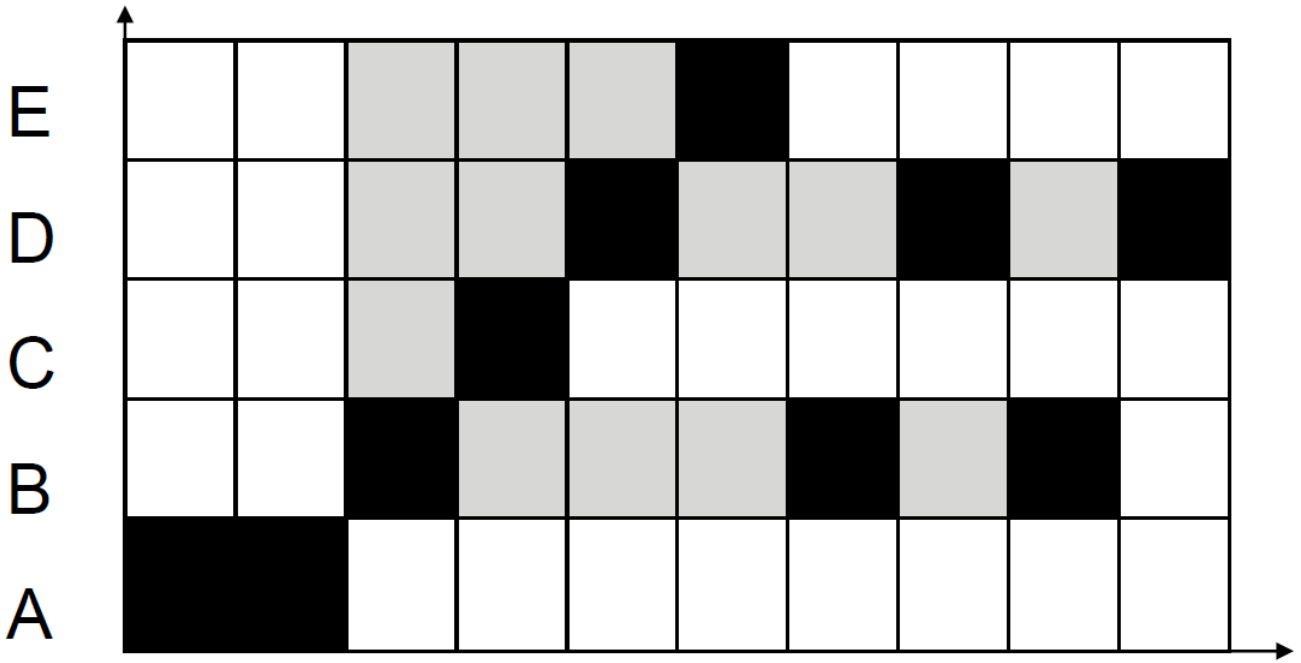
- Geeignete Größe für Δt wichtig (übliche Werte etwa 10...150 ms)
  - Zu kleines Δt: großer Overhead wegen Prozessumschaltung
  - Zu großes Δt: Round Robin wird zu First Come First Serve(d)
- Falls ein Prozess sein Quantum nicht vollständig verbraucht, teilt Round Robin dem nächsten laufbereiten Prozess sofort ein neues Quantum zu
- Anwendung z.B. im Dialogbetrieb, auch Kopplung mit anderen Verfahren (z.B. Prioritäten)

Scheduling

Beispiel

Prozess	A	B	C	D	E
Ankunftszeit	0	2	2	2	2
Bearbeit.-dauer	2	3	1	3	1

Annahme: Zeitscheibe Δt=1



Verweildauer:

4

8

2

7

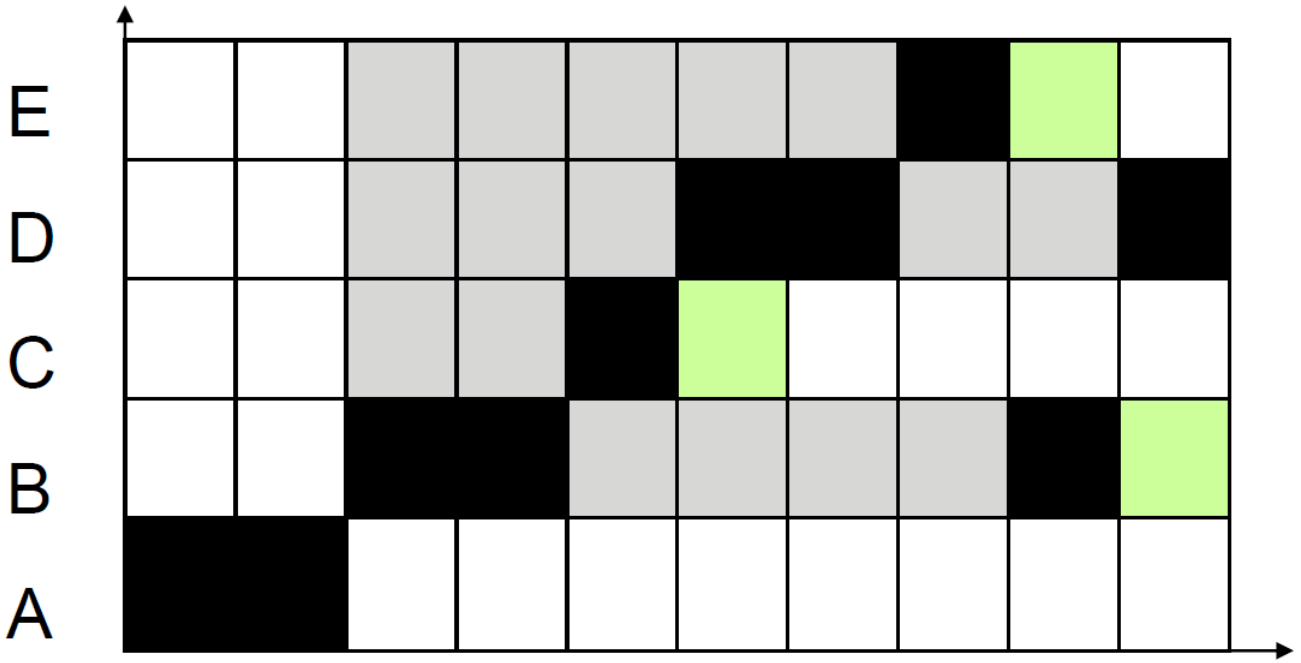
2

Scheduling

Beispiel

Prozess	A	B	C	D	E
Ankunftszeit	0	2	2	2	2
Bearbeit.-dauer	2	3	1	3	1

Annahme: Zeitscheibe Δt=2



Verweildauer:

6

8

3

7

2

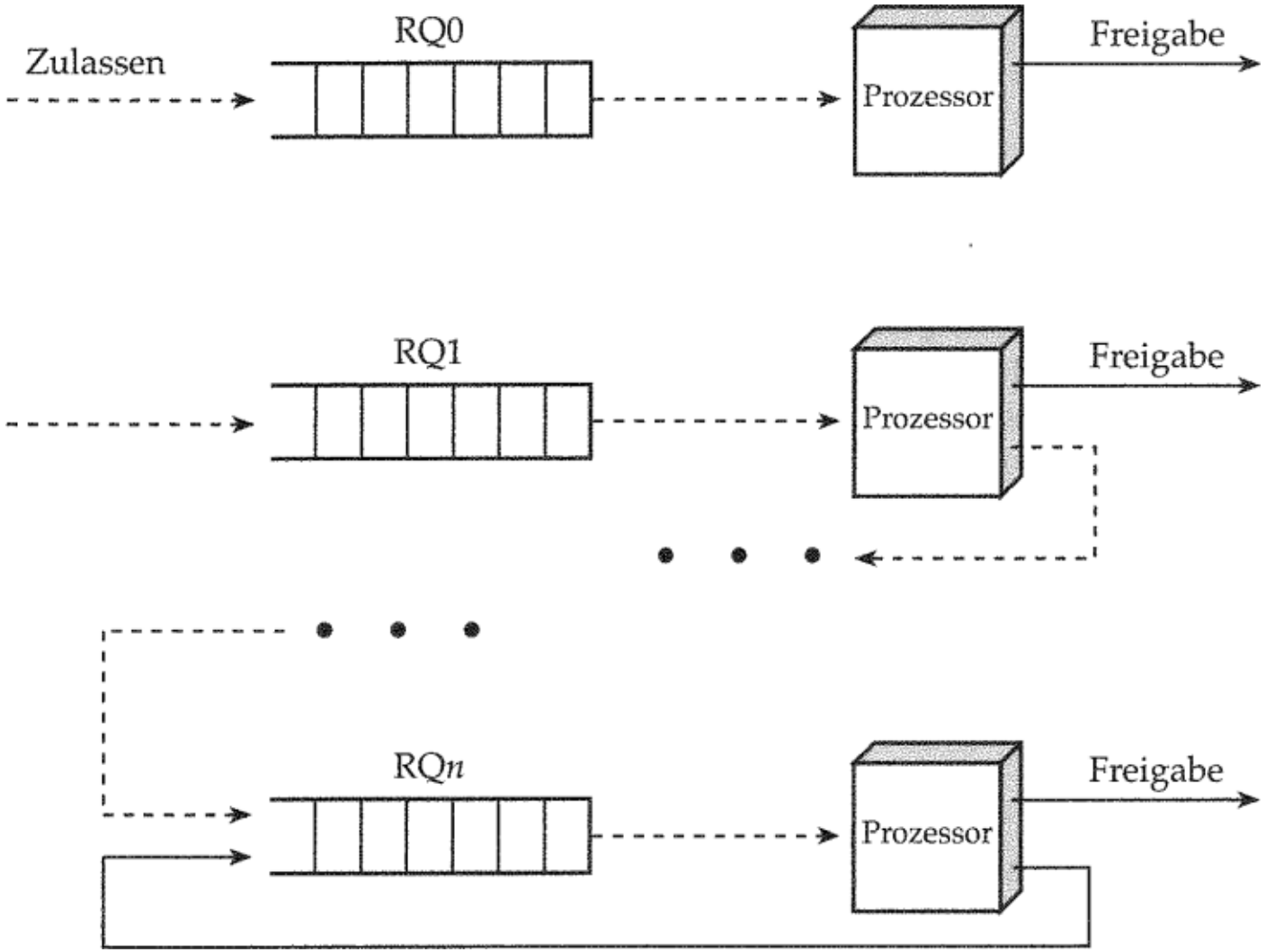
Scheduling

- Feedback Scheduling
  - Problem bei SJF/SJN, HRRN und SRT: eventuell fehlende Kenntnisse über Zeiten (Bearbeitungszeit, Wartezeiten, ...)
  - Ausweg: bisher benutzte Bearbeitungszeit verwenden und präemptiv vorgehen (Zeitunterbrechung nach bestimmten Zeitintervall/ Quantum)

Scheduling

- Mehrere priorisierte Listen für laufbereite Prozesse
- Nach jeder Ausführung rutscht ein Prozess, der noch weiter die CPU benötigt, um eine Stufe nach unten („nächst schlechtere Liste“)
- Innerhalb jeder Liste: FCFS (letzte Liste: Round Robin)
- Bevorzugung neuerer, kürzerer Prozesse gegenüber älteren, langen
- Ggf. Erhöhung des Zeitquantums in den Listen, je „schlechter“ die Liste, desto höher das Quantum

Scheduling



Quelle: W. Stallings: Betriebssysteme – Prinzipien und Umsetzung. Pearson Studium, 2003

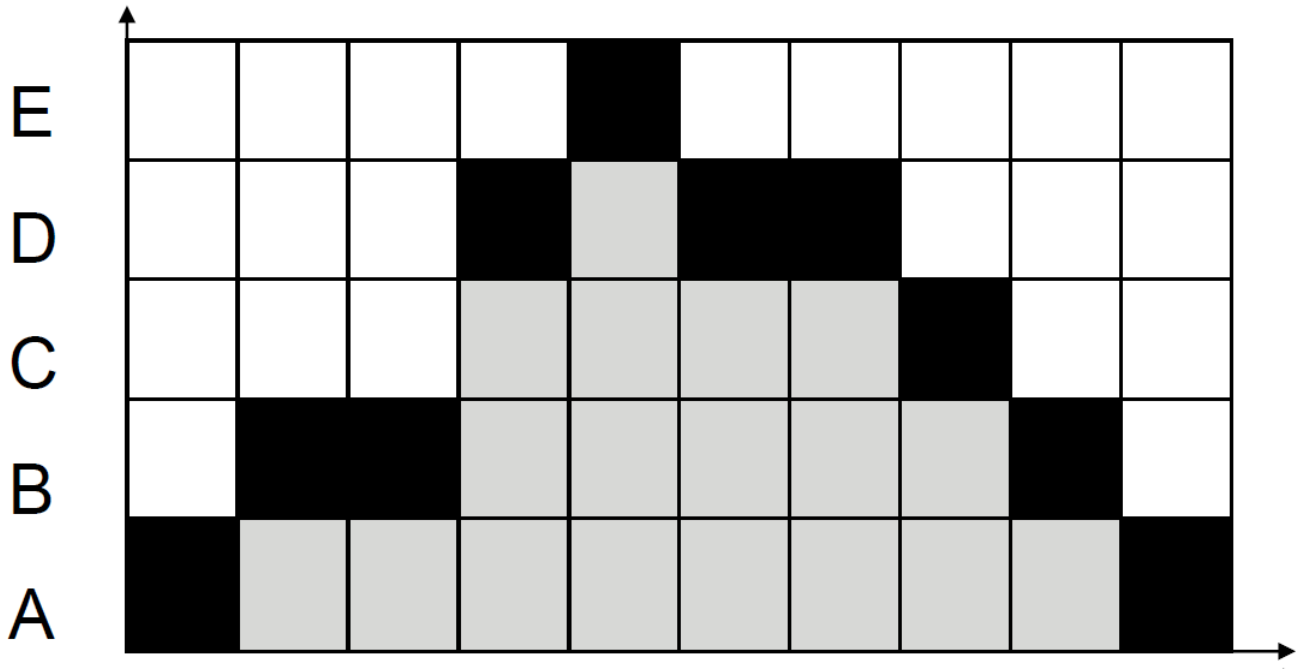
Scheduling

- Prioritäten
  - Zuordnung einer (festen oder dynamischen) Priorität an jeden Prozess
  - Zuteilung der CPU an den bereiten Prozess mit der aktuell höchsten Priorität
  - Präemptiv
  - „Verhungern“ von niedrig priorisierten Prozessen möglich
    - Ggf. dynamische Anpassung (Erhöhung) der Priorität mit dem Alter
  - Anwendung z.B. im Dialogbetrieb, auch bei Echtzeit

Scheduling

Beispiel

Prozess	A	B	C	D	E
Priorität	1=klein	2	3	4	5=hoch
Ankunftszeit	0	1	3	3	4
Bearbeit.-dauer	2	3	1	3	1



Verweildauer:

1

4

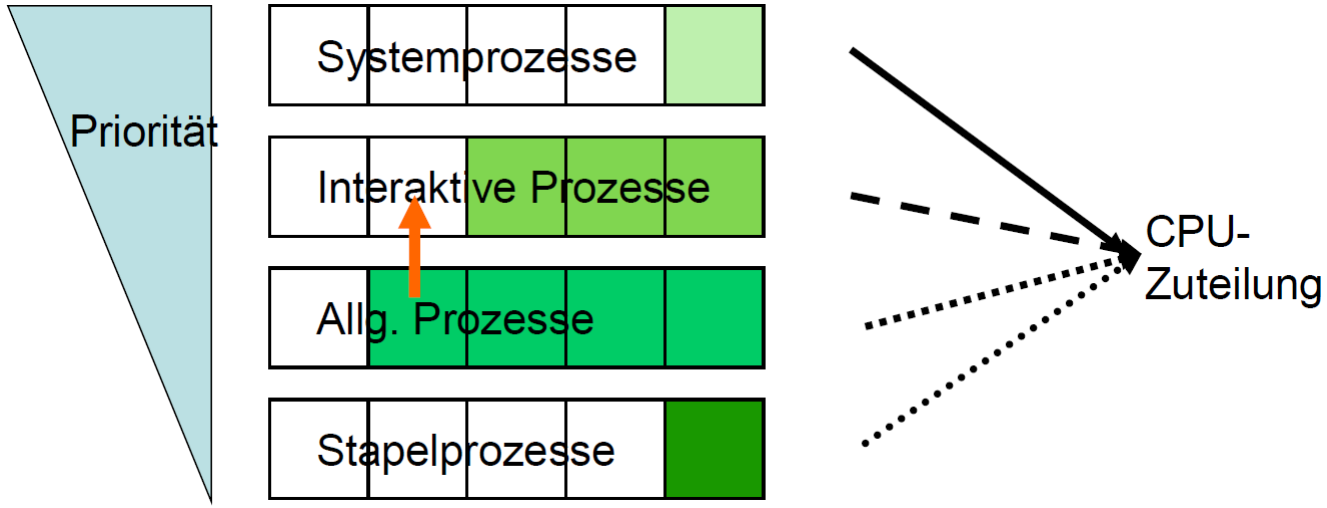
5

8

10

Scheduling

- Problem: welche Entscheidung bei gleicher Priorität?
  - (z.B.) Round Robin für gleich priorisierte Prozesse
  - Prioritätsklassen, „Multilevel Scheduling“
  - Beispiel



  - Multilevel Feedback Scheduling (MLFQ): Prozess kann ggf. in eine andere Warteliste wechseln (z.B. nach oben)

Scheduling

- Multilevel Feedback Scheduling plus weitere „Verfeinerungen“
  - Zeitquantum initial eingestellt, aber dynamisch (vom Betriebssystem) änderbar
  - Prioritäten statisch voreingestellt, aber dynamisch änderbar (vom Betriebssystem bzw. durch Programmierer)
  - Verwaltung von Quantum (Zeit) und Priorität nötig
- Praktische Lösungen (Beispiele)
  - Ein-/Ausgabe-intensive Prozesse erhalten höheres Quantum
  - Prioritätsbonus an bestimmte Prozesse (priority boost)
  - Modifizierte Lösungen z.B. in Windows und UNIX/Linux
- *Hinweis*: Die Veränderung der Prozesspriorität durch das Betriebssystem ist z.B. für Echtzeitlösungen meist nicht tolerierbar

Scheduling

- Mögliches Problem bei Prioritätenscheduling
  - Prioritätenumkehr (priority inversion)
  - Hoch priorisierter Prozess wird eventuell durch niedrig priorisierten Prozess verzögert, der vor dem hoch priorisierten Prozess einen kritischen Abschnitt belegt hat, Verzögerung nicht abschätzbar

Scheduling

- Mögliche Lösung: Prioritätenvererbung (priority inheritance)
  - Niedrig priorisierter Prozess erbt vorübergehend (für Benutzung des kritischen Abschnitts) die Priorität des wartenden hoch priorisierten Prozesses, damit wird eine Begrenzung der Verzögerung möglich
  - In einigen Betriebssystemen ist eine entsprechende Option zur Prioritäten-Vererbung für die Semaphor-Funktion p(S) vorhanden
  - Beispiel Linux: „Realtime-Mutex“ = Mutex + Prioritäten-Vererbung

Scheduling

- Weitere Schedulingverfahren
  - Lotterie-Scheduling: Zufällige Auswahl eines Prozesses für eine zufällige Dauer, z.B. für Dialogbetrieb
  - Fair Share Scheduling: Gruppierung von Prozessen, „faire“ Zuteilung der CPU an die Prozesse einer Gruppe; je mehr Zeit für einen Prozess bzw. eine Gruppe verbraucht wurde, desto geringer wird ihre Priorität
  - Earliest Deadline First (EDF): falls Ausführungszeiten und Deadlines bekannt sind, erfolgt Zuteilung an den Prozess mit der am nächsten liegenden Deadline -> Echtzeitbetrieb

Scheduling

- Rate Monotonic Scheduling (RMS): statische Planung, für periodische (zyklische) Prozesse wird die Priorität als invers zu ihrer Periodendauer festgelegt, d.h. kurze Periode -> hohe Priorität
- Weitere spezielle Verfahren für Multiprozessor-Systeme

Zusammenfassung

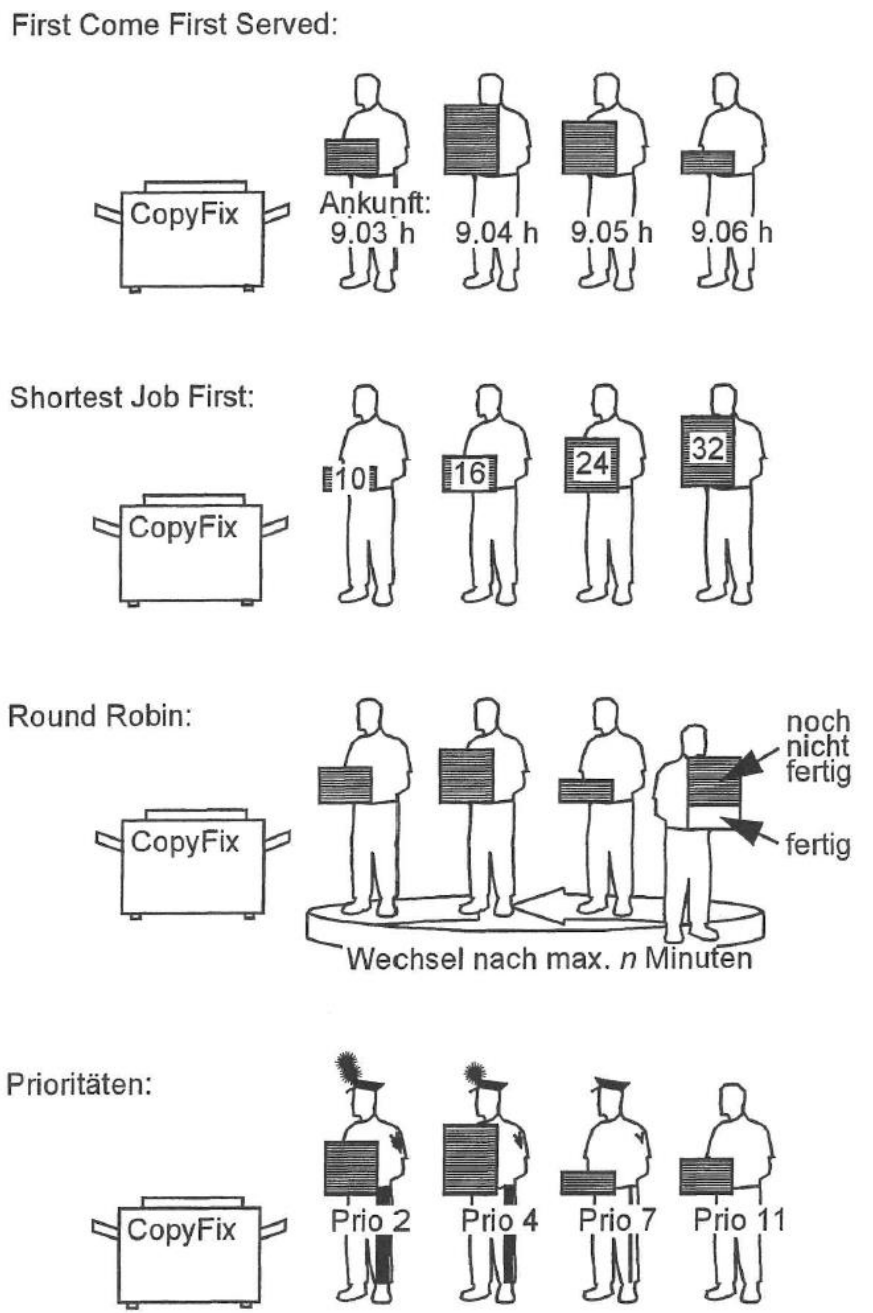


Abb. 3.11 Schedulingstrategien am Beispiel einer Warteschlange vor einem Kopiergerät

Quelle: C. Vogt: Betriebssysteme. Spektrum Akademischer Verlag, 2001