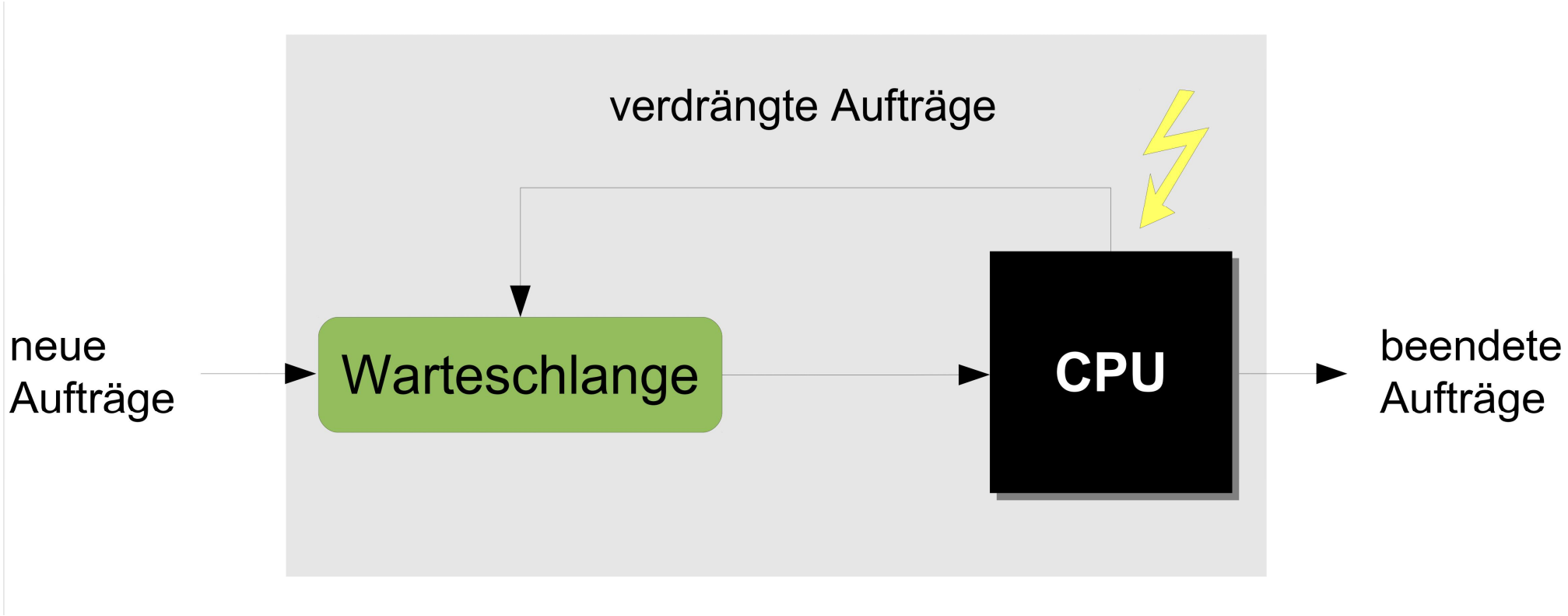
Scheduling

- Sorgt für den geordneten Ablauf konkurrierender Prozesse
- Grundsätzliche Fragestellungen
  - Welche Arten von Ereignissen führen zur Verdrängung?
  - In welcher Reihenfolge sollen Prozesse ablaufen?
- Wichtig eher für Großrechner / Server
  - viele rechenbereite Prozesse
  - Mischung aus Stapel- und TimeSharing-Betrieb
- Ziele einer Schedulingstrategie (scheduling algorithm)
  - benutzerorientiert, z.B. kurze Antwortzeiten
  - systemorientiert, z.B. optimale CPU-Auslastung
- Keine Schedulingstrategie kann alle Bedürfnisse erfüllen.

Scheduling



Ein einzelner Scheduling-Algorithmus charakterisiert sich durch die Reihenfolge von Prozessen in der Warteschlange und die Bedingungen, unter denen die Prozesse der Warteschlange zugeführt werden.

Was passiert bei einem Taskwechsel?

- Wechsel vom Benutzer- in den Kernel-Modus
- Sicherung des Zustands des laufenden Prozesses
  - Akku
  - Register
  - Speicherzuordnungstabelle
- Auswahl eines neuen Prozesses
- Laden der neuen Speicherzuordnungstabelle
- Start des Prozesses
- Cache neu laden

? Anzahl Taskwechsel/Kontextwechsel unter Windows:

Prozessverhalten

- CPU-intensive Prozesse
  - lang anhaltender Bedarf an Rechnungen
- E/A-intensive Prozesse
  - nur kurze Berechnungen, häufige (evtl. auch kurze) E/A

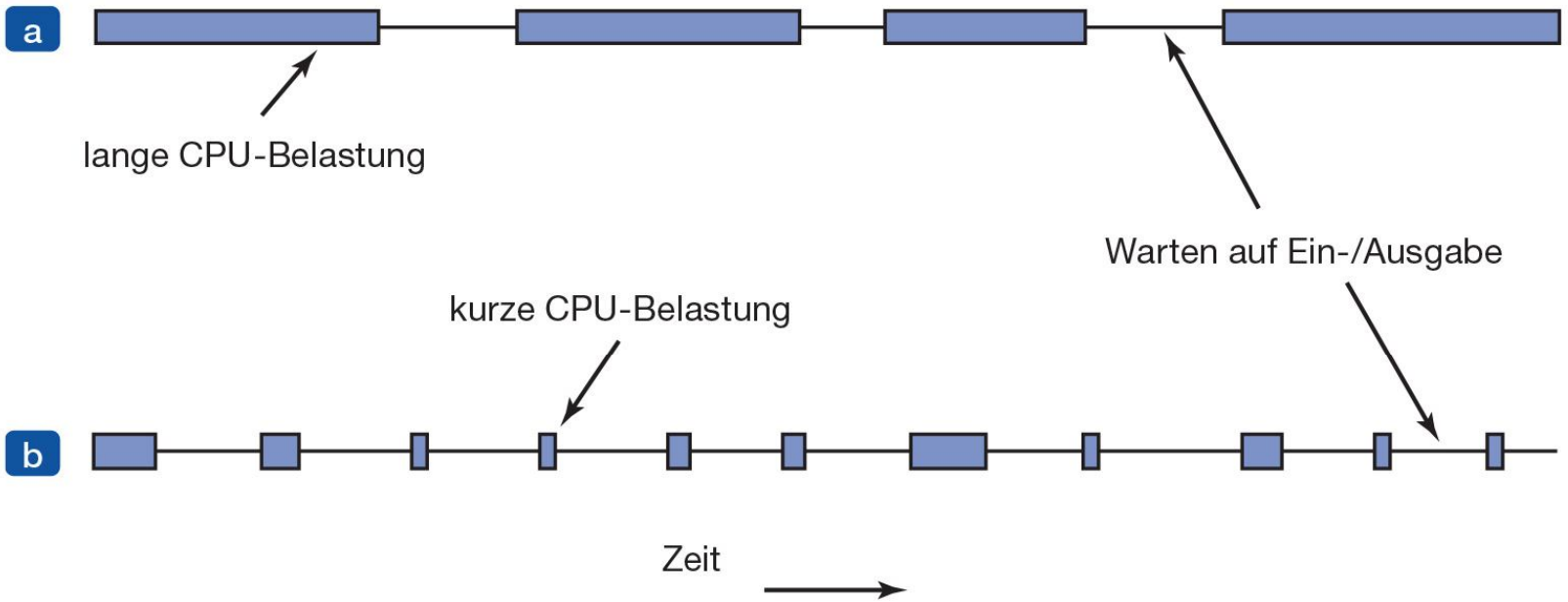


Abbildung 2.39: Häufung von CPU-Gebrauch alternierend mit Zeiträumen, in denen auf E/A gewartet wird: (a) ein CPU-intensiver Prozess; (b) ein E/A-intensiver Prozess.

Ziele von Schedulingstrategien

- Alle Systeme
  - Fairness – jeder Prozess bekommt einen fairen Anteil an Rechenzeit
  - Policy Enforcement – vorgegeben Strategien werden durchgesetzt
  - Balance – alle Teile des Systems sind ausgelastet
- Stapelverarbeitungssysteme
  - Durchsatz – Maximierung der Jobs pro Stunde
  - Durchlaufzeit – Minimale Zeit vom Start bis Ende
  - CPU-Auslastung
- Interaktive Systeme
  - Antwortzeit – schnelle Reaktion des Systems
  - Proportionalität – Erwartungen des Benutzers erfüllen
- Echtzeitsysteme
  - Deadlines einhalten - Datenverlust vermeiden
  - Vorhersagbarkeit - Qualitätseinbußen in Multimediasystemen vermeiden

Scheduling Stapelverarbeitungssystemen

- First-Come-First-Served-Scheduling
- Shortest-Job-First-Scheduling
- Shortest-Remaining-Time-Next-Scheduling

First-Come-First-Served-Scheduling

- Rechenbereite Prozesse werden in eine Warteschlange eingeordnet und gemäß ihrer Ankunftszeit von der CPU abgearbeitet
- Blockiert ein Prozess, dann kommt der nächste Prozess an die Reihe.
- Wird der blockierte Prozess wieder rechenbereit, wird er neu am Ende der Warteschlange einsortiert (kooperativ)
- Vorteile
  - einfach
  - keine Unterbrechung von Prozessen durch Scheduler (Selbstaufgabe)
- Nachteil
  - hohe Durchlaufzeit von E/A-intensiven Prozessen

Shortest-Job-First-Scheduling

Abbildung 2.41: Ein Beispiel für Shortest-Job-First-Scheduling: (a) Ablauf der vier Jobs in der Originalfolge. (b) Ablauf in der Shortest-Job-First-Reihenfolge.

- Durchschnitt der Laufzeit im Fall a:
  (8 + 8+4 + 8+4+4 + 8+4+4+4) / 4 = 14
- Durchschnitt der Laufzeit im Fall b:
  (4 + 4+4 + 4+4+4 + 4+4+4+8) / 4 = 11

Shortest-Job-First-Scheduling

- Einsatz in der Stapelverarbeitung
- Laufzeit muss bekannt sein, bzw. abgeschätzt werden können
- Die bereiten Prozesse werden gemäß ihrer Laufzeit in der Warteschlange sortiert und abgearbeitet (kooperativ)
- Vorteile
  - einfach
  - keine Unterbrechung von Prozessen durch Scheduler (Selbstaufgabe)
- Nachteil
  - Laufzeit muss bekannt sein
  - Prozesse müssen verfügbar sein

Shortest-Remaining-Time-Next-Scheduling

- unterbrechende Version von Shortest Job First
- Scheduler wählt immer den Prozess aus, dessen **verbleibende Zeit** am kürzesten ist.
- Sobald ein neuer Job ankommt, wird dessen **gesamte Laufzeit** mit der **verbleibenden Zeit** des aktuellen Prozesses verglichen.
- Benötigt neue Prozess weniger Zeit -> stopp und neue Prozess beginnt.
- Dieses Modell ist für neue, kurze Jobs sehr günstig.

Scheduling interaktive Systeme

- Round-Robin-Scheduling
- Prioritätsscheduling
- Multi-Level Feedback Scheduling
- shortest-Process-Next-Scheduling
- Garantiertes Scheduling (Guaranteed Scheduling)
- Fair-Share-Scheduling

Round-Robin-Scheduling

- Jeder bereite Prozess darf gewisse Zeit Q (Quantum) laufen
  - Läuft am Ende von Q Prozess immer noch, wird unterbrochen.
    -> danach an das Ende der Warteschlange
  - Vorzeitige Blockade oder Beendigung -> sofortige Weitergabe
- **Präemptives** Verfahren
- Zeitscheibe: feste Zeit Z, typische Werte für Z: 10-100 ms
- Vorzeitiger Prozessorentzug durch Blockierung oder Terminierung des Prozesses möglich

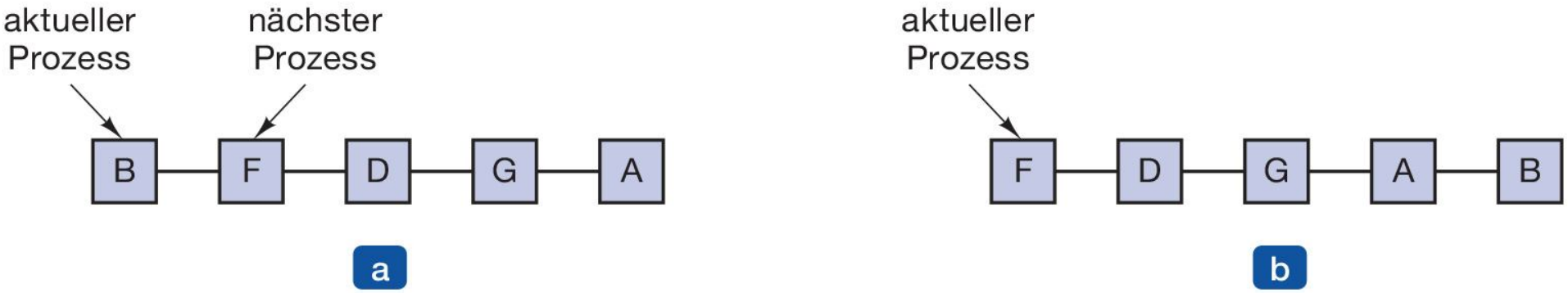
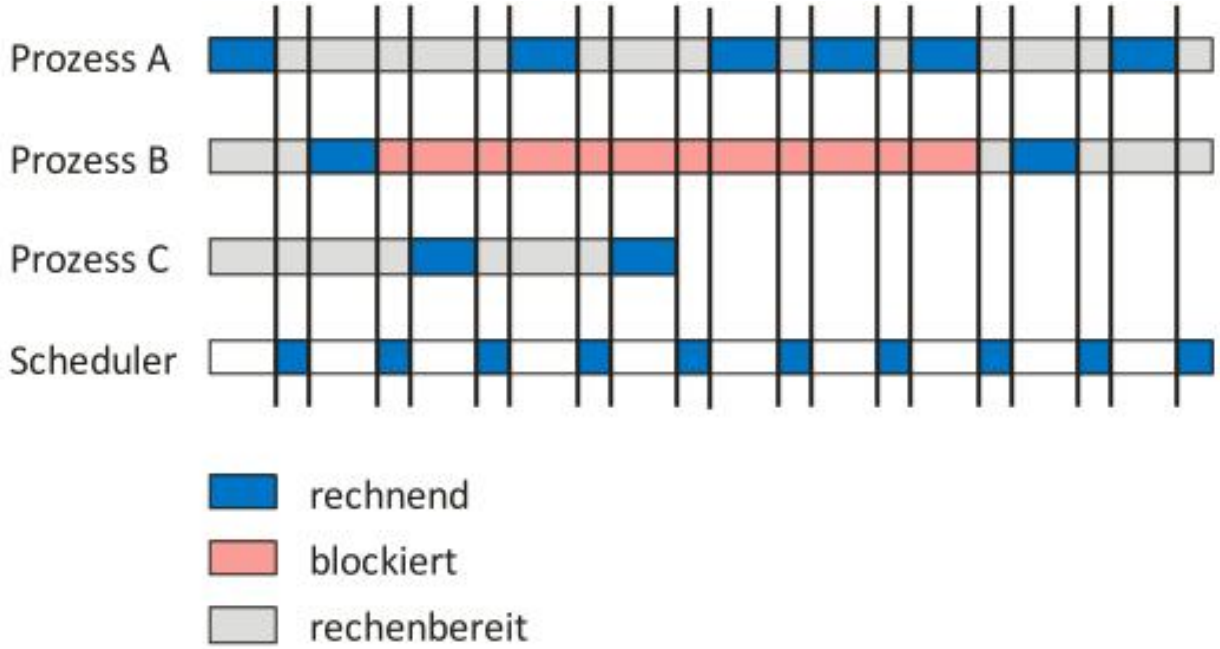


Abbildung 2.42: Round-Robin-Scheduling: (a) Liste der lauffähigen Prozesse; (b) Liste der lauffähigen Prozesse, nachdem *B* sein Quantum aufgebraucht hat.

Round-Robin-Scheduling Beispiel



- Problem
  - kritische Einstellung der Zeitspanne, die ein Prozess rechnen darf
    - hoher Wert: hohe Antwortzeiten, wenig Kontextwechsel
    - niedriger Wert: kurze Antwortzeit, viele Kontextwechsel
  - Prozesse müssen präemptiv sein

Prioritätsscheduling

- Jedem Prozess wird eine Priorität (Wichtigkeitsstufe) durch den Anwender oder ein gesondertes Verfahren zugeordnet.
- Laufen wird nun stets der oder die bereiten Prozesse mit der höchsten Priorität;
- laufende Prozesse mit geringerer Priorität werden verdrängt. Bei gleicher Prioritätsstufe wird meist Round-Robin oder FCFS angewendet.
- Prioritäten können
  - statisch oder
  - dynamisch vergeben werden

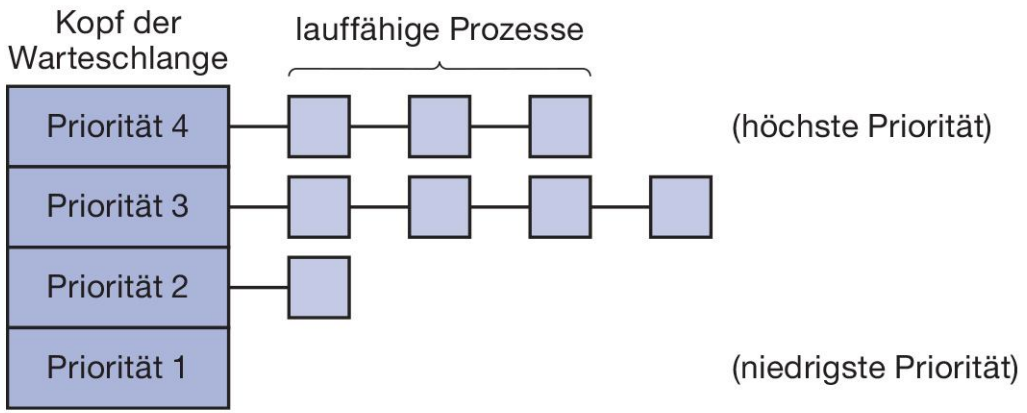


Abbildung 2.43: Eine Schedulingstrategie mit vier Prioritätsklassen.

Prioritätsscheduling

- Mögliches Problem:
  - Starvation – Verhungern (ein Prozess wird nie ausgewählt)
- Die Umsetzung erfolgt mit
  - mehreren Ready Queues für unterschiedliche Prioritäten (wenig Prioritäten)
  - geordneter Ready Queue (sonst)
- Wichtig
- E/A-intensive Prozesse, sollten schnell Rechenzeit bekommen
  - wenig Blockierung der E/A-Ressourcen
  - halten bei der nächsten E/A-Operation wieder an

Multi-Level Feedback Scheduling

- Ein Prozess wird abhängig von seiner bisherigen Laufzeit priorisiert.
- Wird ein Prozess neu erzeugt, dann erhält er die höchste Priorität.
- Wenn dieser Prozess das erste Mal beendet wird, dann wird auch seine Priorität schrittweise herabgestuft.
- Präemptiv mit festem Zeitintervall
- Mögliches Problem ist Starvation, wenn häufig neue Prozesse eintreten
- Lösung
  - Größere Zeitintervalle, je niedriger die Priorität
  - Prozess allmählich wieder hoch priorisieren

Shortest-Process-Next-Scheduling

- Shortest-Job-First-Strategie für interaktive Prozesse
- Schema:
  auf Befehl warten - Befehl ausführen - auf Befehl warten - Befehl ausführen ...
- Befehl entspricht Job, somit Antwortzeit minimieren durch kürzesten zuerst.
- Problem: Abschätzen der Laufzeit der Befehle
- Ansatz: Historischer Verlauf der Befehle
  gleitender Mittelwert, gewichteter Mittelwert...

Garantiertes Scheduling (Guaranteed Scheduling)

- Versprechungen bezüglich der Geschwindigkeiten machen und einhalten
  - Bsp.: Wenn n Benutzer eingeloggt sind, bekommt jeder etwa 1/n der CPU-Leistung
  - Ähnlich auf einem Einbenutzersystem mit n laufenden Prozessen
- Nachverfolgen der zugewiesenen CPU-Zeit zur verbrauchten

Fair-Share-Scheduling

- Bisherige Betrachtung unabhängig von Nutzer
  - Startet Benutzer 1 neun Prozesse und Benutzer 2 einen Prozess, bekommt Benutzer 1 90% der CPU-Zeit und Benutzer 2 10%.
- Nach Fair-Share belegt jeder **Benutzer** gleiche CPU-Zeit (hier 50%) unabhängig von Anzahl der Prozesse

Aufgaben

- Vergleich der Scheduling-Verfahren

Prozesse

Prozess	Ankunftszeit	Laufzeit
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

In welcher Reihenfolge werden die Prozesse bei folgenden Scheduling-verfahren ausgeführt und nach welcher Laufzeit sind sie jeweils beendet?

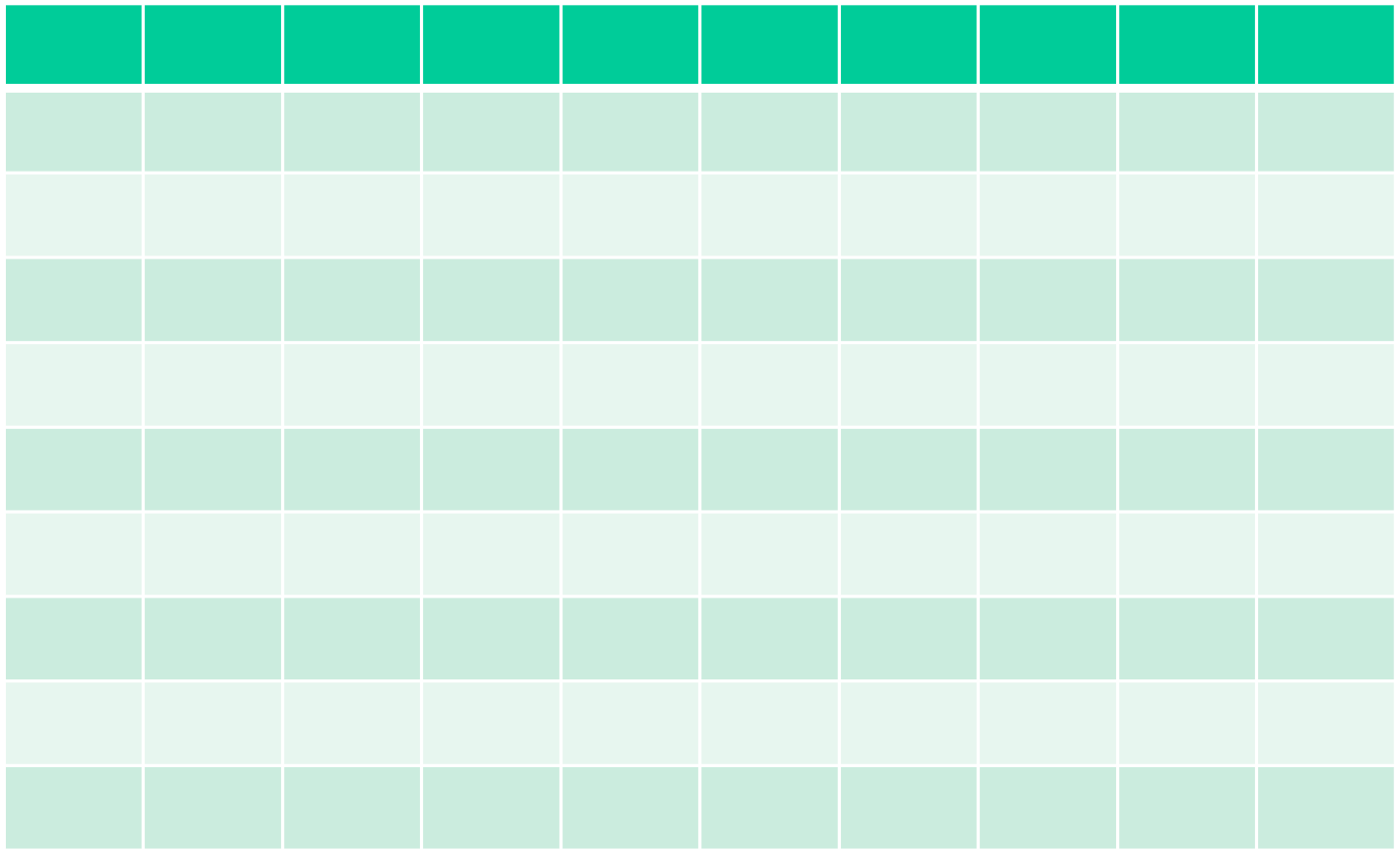
a) FCFS (First come first serve)

b) SJF (Shortest job first)

c) RR (Round Robin, timeslice = 1)

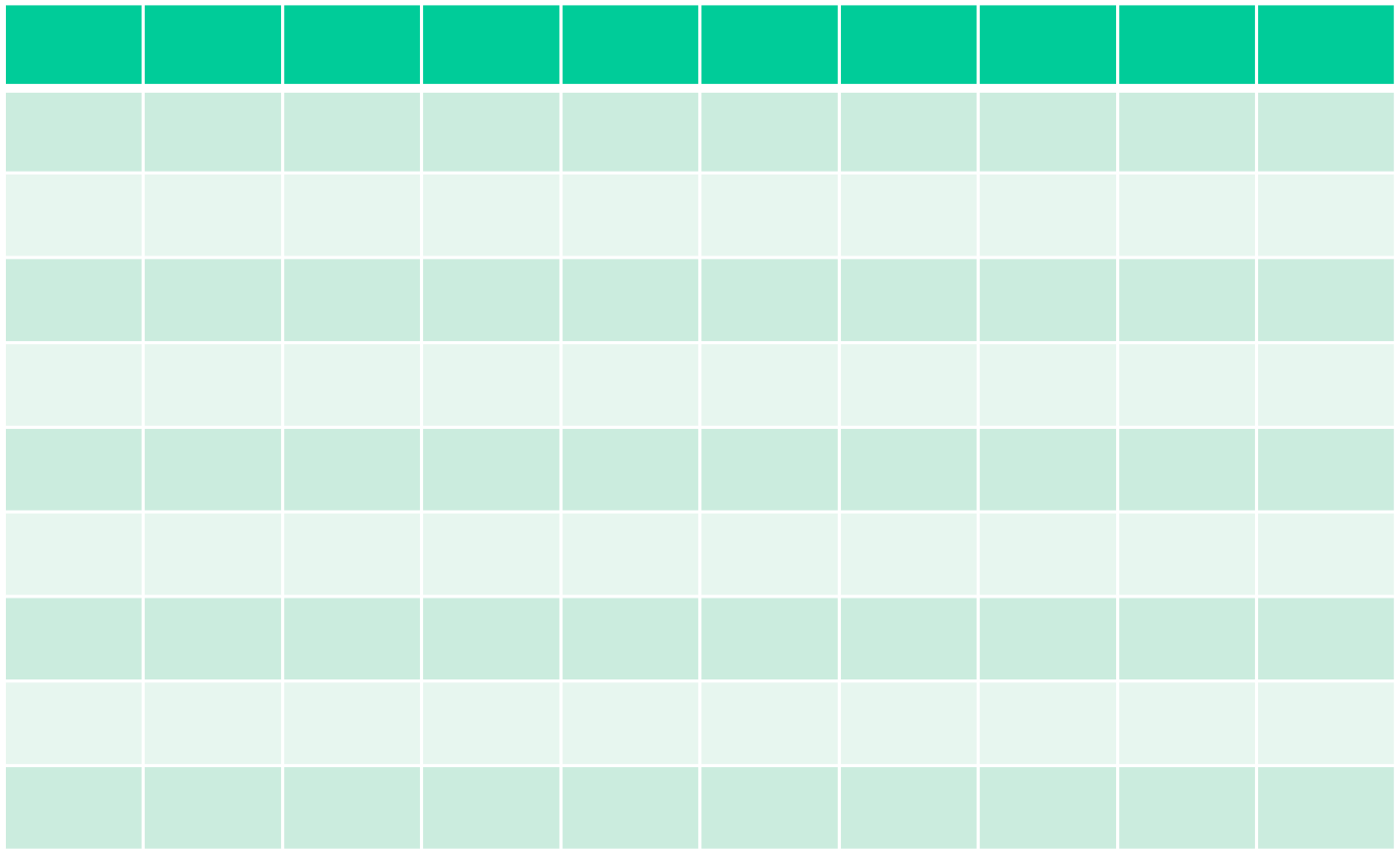
FCFS (First come first serve)

Prozess	Ankunftszeit	Laufzeit
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2



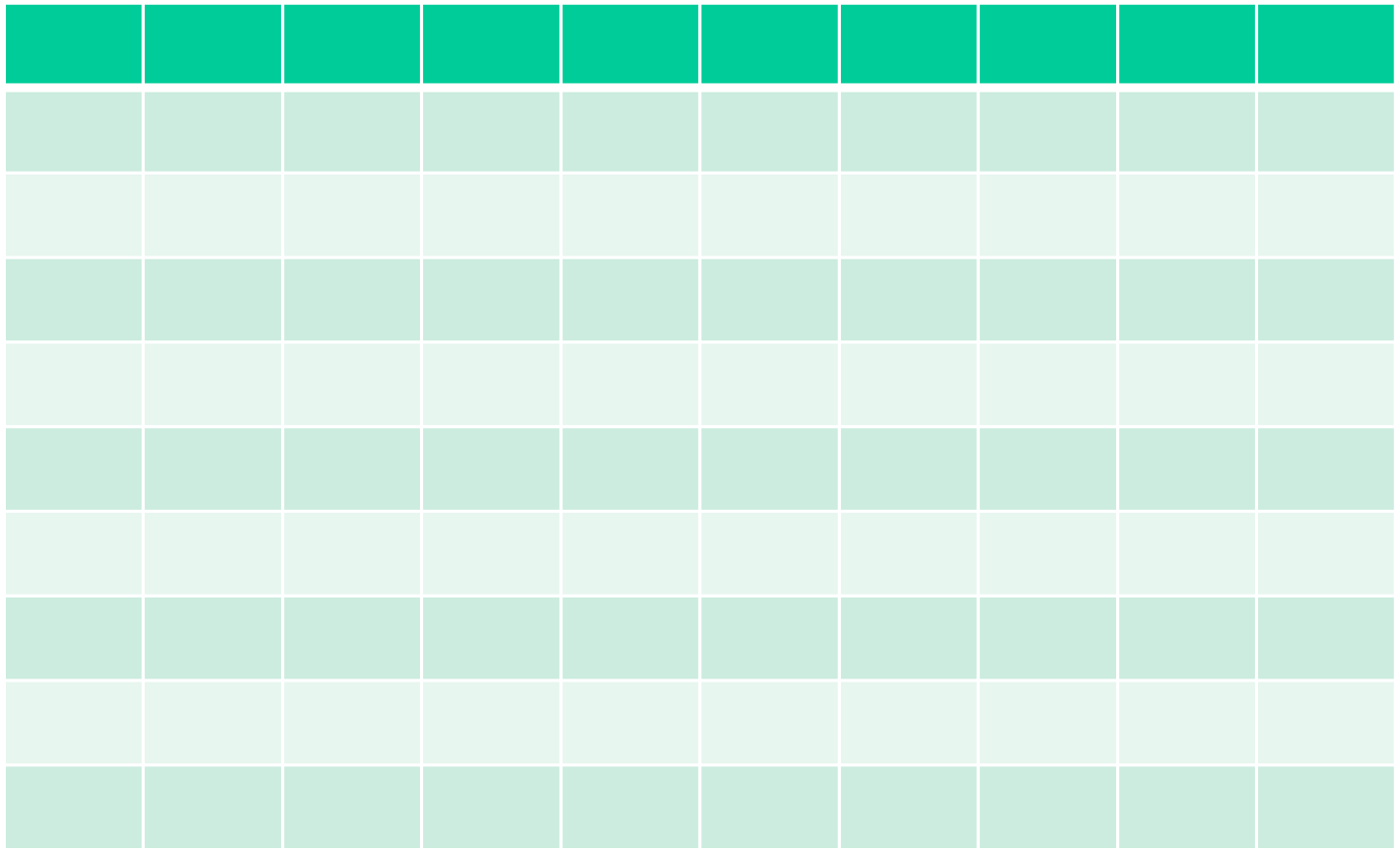
SJF (Shortest job first)

Prozess	Ankunftszeit	Laufzeit
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2



RR (Round Robin, timeslice=1)

Prozess	Ankunftszeit	Laufzeit
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2



Scheduling Echtzeitsysteme

- harte Echtzeitsysteme
  - absolute Deadline
- weiche Echtzeitsysteme
  - Dealine, nicht einhalten unerwünscht aber tolerierbar.

- Echtzeitverhalten durch Unterteilung in vorhersehbare und vorher bekannte Ereignisse
  - Prozesse allgemein kurzlebig.
  - Weit weniger als einer Sekunde vollständig abarbeitbar.