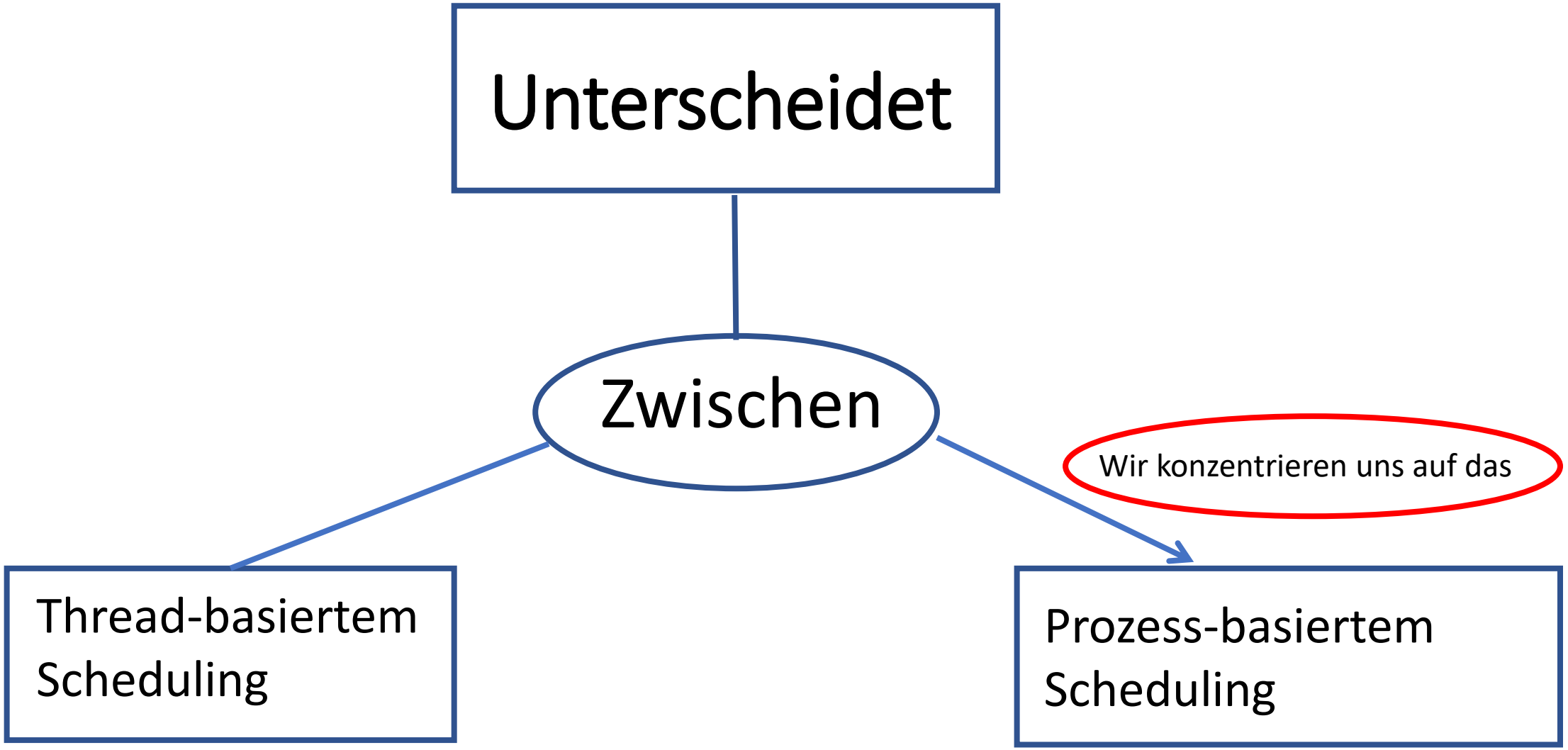
Prozess-Scheduling

Im Betriebssystem


Unterscheidet
Zwischen
Wir konzentrieren uns auf das
Thread-basiertem Scheduling
Prozess-basiertem Scheduling

Funktionsweise

- Scheduler dient als Verwaltung der Prozesse
- Scheduler bestimmt Reihenfolge und Länge der CPU-Nutzung der Prozesse
- Scheduling erfolgt nach unterschiedlichen Strategien und Systemen
- Ausgeführter Prozess kann die ihm Verfügbaren Ressourcen nutzen
- Scheduler übermittelt Prozess an Dispatcher

Wichtig

Der Dispatcher übergibt die Kontrolle der CPU an den vom Scheduler ausgewählten Prozess

In einem PC mit nur einem Prozessor kann immer nur ein Prozess ausgeführt werden
Die anderen Prozesse müssen darauf warten bis die CPU wieder frei ist und sie der CPU zugewiesen werden.

Dispatcher

Der Dispatcher übergibt die Kontrolle
der CPU an den vom Scheduler ausgewählten Prozess

Weitere Aufgaben:

- Kontextwechsel
- Umschaltung in den Benutzermodus
- Springt in die Adresse des ausführenden Prozesses

Dispatcher-Latency: Zeit zwischen Prozess-Stop und Prozess-Start

Ziele für ein gutes Scheduling

- **kurze Antwortzeiten**
  Zeit zwischen Eingabe und Reaktion

- **hoher Durchsatz**
  Maximierung der Prozessanzahl pro Zeiteinheit

- **hohe Auslastung der Prozessoren**
  Maximierung der CPUs-Beschäftigungszeiten

- **Einhaltung von Deadlines**
  Interessant für Echtzeit-Scheduling

- **niedriger Durchlaufzeit**
  Minimierung der Zeit zwischen Prozess-Start und Prozess-Beendigung

Hinweis

Ziele sind teilweise Widersprüchlich, wenn man sie miteinander Vergleicht

Anforderungen an den Scheduler

Aus der Benutzersicht:

- Minimierung der Durchlaufzeit

(Ausführungszeit + Wartezeiten)

- Minimierung der Antwortzeit (interaktive Systeme)
- Einhaltung von Zeitschranken (Deadlines)

=> Besonders wichtig für Echtzeit-Betriebssysteme

Aus der Systemsicht:

- Maximierung des Durchsatzes

- Maximierung der Prozessorauslastung

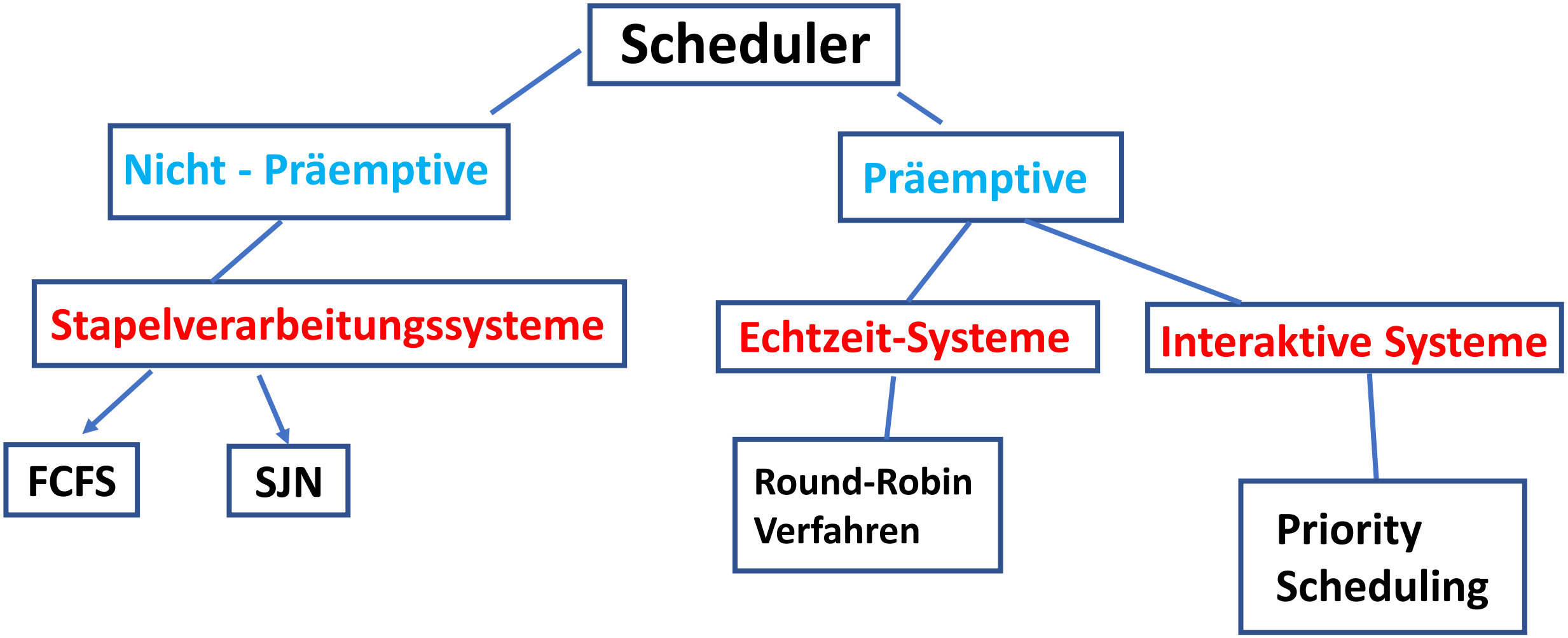
- Balance. Auslastung aller Ressourcen

- Fairness. Gleichwichtige Prozesse sollen möglichst gleich behandelt werden

Übersicht von Begriffen

CPU-Auslastung	Beschäftigungszeit (in Prozent).
Ausführungszeit	Effektive CPU-Zeit, die ein Prozess von der CPU in Anspruch nimmt.
Wartezeit	Zeit, die ein Prozess in Warteschlangen verbringt.)
Durchsatz	Anzahl der Prozesse, die pro Zeiteinheit ausgeführt werden.
Antwortzeit	Zeit zwischen Anforderung und Antwort.
Turnaround Time	Durchschnittszeit der Verarbeitung einzelner Prozesse.
Overhead	Zeitaufwand für die Entscheidung + Kontextwechsel.

Gesamtübersicht



Präemptiv und Nicht -Präemptiv

Präemptiv

aktive Prozesse können wieder in den Bereit Zustand versetzt werden

=> kommt wieder in die Warteschlange

Nicht-Präemptiv

Prozess arbeitet solange bis er sich selbst abschaltet
oder er
von außen blockiert wird
Bsp: PowerPoint, Word

Stapelverarbeitungssysteme

Funktion

- Scheduler fügt neue Prozesse in die Warteschlange und bestimmt ihre Position
- Prozesse werden nacheinander abgearbeitet und nach jedem Prozess kommt nächste

Ziele:

- **Durchsatz:** Maximierung der erledigten Prozesse pro Stunde.
- **Durchlaufzeit:** Minimierung der Zeit vom Start bis zum Ende eines Prozesses
- **CPU-Auslastung:** Die CPU ist immer ausgelastet

Einsatzbereich

- Nicht für interaktive Systeme gedacht
- wird häufig bei Großrechnern genutzt

First-Come First-Served (FCFS)

Grundlegende Funktion

- Reihenfolge der Prozesse wird mit ihrer Einreihung in die Warteschlange festgelegt
- Der Prozess der zuerst "eingetroffen" ist, wird auch als erstes Bearbeitet
- Prozesse werden komplett bearbeitet bis der nächste drankommt
- nicht unterbrechbarer Stapelalgorithmus
- einfachste Scheduling Algorithmus

Vorteile

- einfache Implementierung
- niedriger Scheduling Overhead
- Effizienz (kaum Leerlauf)

Nachteile

- niedrige CPU-Auslastung
- ungeeignet für interaktive und Mehrbenutzersysteme

=> möglicherweise lange Wartezeiten für Benutzer falls größere Prozesse vor ihnen drankommen

Shortest-Job-Next

Grundlegende Funktion

Auftrag mit der niedrigsten Bearbeitungszeit wird bearbeitet

=> Bearbeitungszeit wird vom System aus Erfahrungswerten eingeschätzt

=>Laufzeiten sind dadurch bekannt

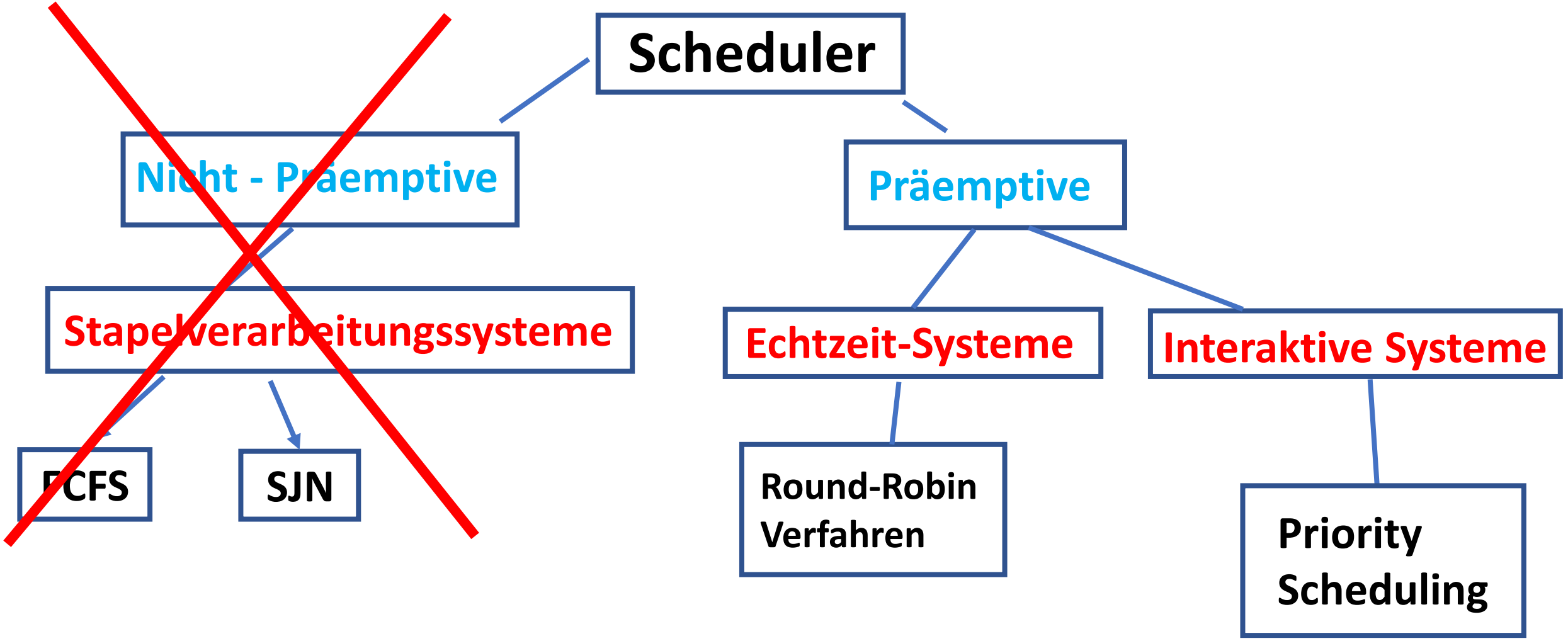
Vorteile

- Minimierung der durchschnittlichen Wartezeit
- Maximierung des Durchsatzes

Nachteile

- niedrige CPU-Auslastung
- ungeeignet für interaktive und Mehrbenutzersysteme
- größere Prozesse bekommen CPU nicht zugeteilt falls sich kleinere Prozesse sich vordrängeln

Gesamtübersicht



Interaktive Systeme

Funktion

Prozesse, welche mit dem Nutzer interagieren werden bevorzugt
(es wird mit Priorisierungen gearbeitet)

Einsatzbereich

Typisch für Arbeitsplatzrechner und Server

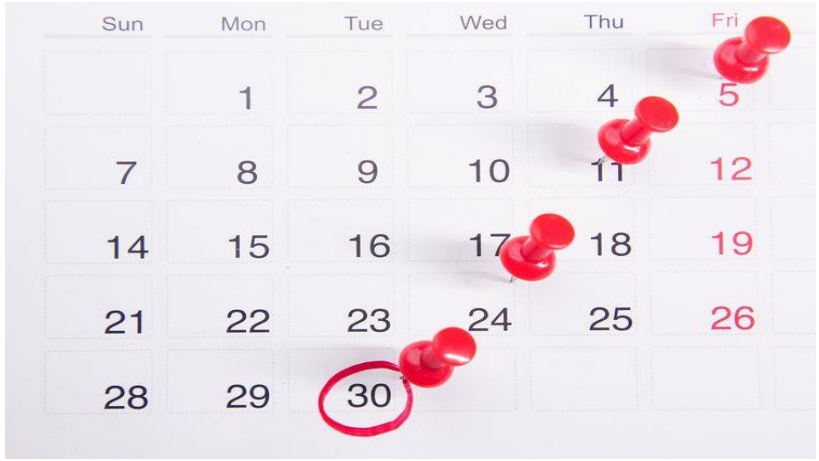
Ziele:

- **Antwortzeit:** schnelle Reaktion auf Anfragen.
- **Proportionalität:** Erwartungen des Benutzers erfüllen

Echtzeit Systeme

Funktion

Prozesse, erhalten Fristen um die Vorhersagbarkeit gewährleisten zu können



Einsatzbereich

Multimedia-Systeme und Sicherheitsanwendungen (Autopiloten und Airbags)

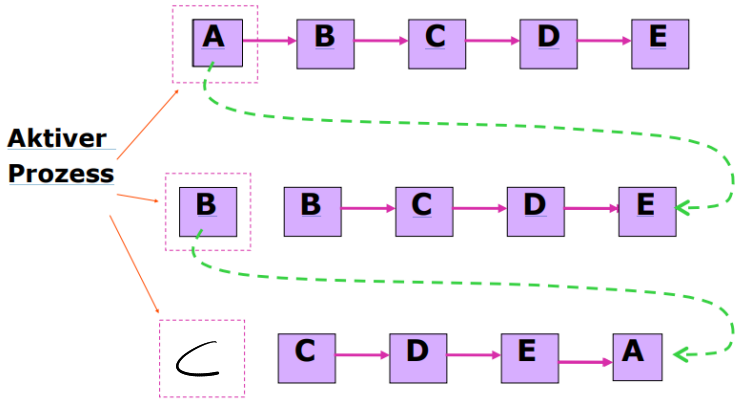
Ziele:

- **Deadlines einhalten:** Datenverlust ausschließen
- **Vorhersagbarkeit:** Qualitätseinbußungen in Multimedia-System vermeiden

Round Robin (Zeitscheibenverfahren)

Grundlegende Fakten

- benutzt im **interaktivem System**
- ältester, einfachster und meist verwendeter **Algorithmus**
- Jeder Prozess erhält einen **Zeitabschnitt (Quantum)**
- Ein **Prozess** steht nur für ein bestimmtes Quantum die CPU zur Verfügung
- Wurde der vorherige **Prozess** nicht komplett bearbeitet, kommt er an das **Ende der Warteschlange**



Wichtige Frage

Wie lange geht so ein Quantum?

10ms bis 100ms

Nachteile

- Prozesse können nicht zuerst bearbeitet werden
  => Müssen warten bis drankommen
- unflexibel und häufig bleiben Ressourcen ungenutzt
- mittelmäßiger Durchsatz

Vorteil

Fairness: Jeder Prozess wird gleich behandelt und kommt dran

Prioritätsbasiertes Scheduling

Grundlegende Funktion

Jeder Prozess bekommt Prioritätszahl zugewiesen
=> Prozess mit höchster Priorität wird ausgeführt
Zuweisung der Prioritäten erfolgt dynamisch

Prioritäten Vergabe

nach internen Kriterien

- **Durchschnitt E/A-Zeit**
- **Speicherbedarf**
- **durchschnittliche CPU-Zeit**
- **Zeitgrenzen**
- **usw.**

nach externen Kriterien

Problem:

Prozesse mit niedriger Priorität
kommen nicht dran
(Prozesse mit hoher Priorität drängeln sich andauernd vor) **Priority-aging**

Lösung

Prozesse mit niedriger Priorität erhöhen die Priorität beim warten

Wichtig

Ein wartender Prozess höherer Priorität, verdrängt einen zurzeit aktiven Prozess niedrigerer Priorität

Schlussfolgerungen

Das Ziel für Alle Scheduler

- Zuteilung der Anwendung an die Prozesse bestmöglich zu ermöglichen, wobei (Abhänging vom System) unterschiedliche Ziele verfolgt werden sollen.
- Fairness bleibt ein Teil von jedem =>Kein Prozess sollte grundlos warten
- Balance sollte eingehalten werden => gesamte vorhandene Hardware sollte ausgeschöpft werden