Aufgabe 1. Sinn von Kernel- und User-Mode

Der User-Mode beschränkt den Befehlssatz auf unbedenkliche Anweisungen und auf die Daten der jeweiligen Prozesse. Außerdem wird dem Betriebssystem exklusiv ermöglicht, Hardwareessourcen wie die CPU-Zeit auf die Prozesse zu verteilen.

Aufgabe 2. Systemaufruf

Der Prozess fragt beim Betriebssystem die Ausführung eines Kernel-Mode-Befehls an. Daraufhin überprüft das Betriebssystem, ob die Ausführung zulässig ist und führt in diesem Fall den Befehl aus, nachdem in den Kernel-Mode gewechselt wurde. Anschließend wird wieder in den User-Mode gewechselt und die Kontrolle an den Prozess zurückgegeben.

Aufgabe 3. Von 'rechnend' nach 'bereit'

Beim Preemptive Scheduling kann einem Prozess die Nutzung der CPU entzogen werden, wodurch der Prozess direkt in den Zustand 'bereit' wechselt.

Aufgabe 4. Prozesskontrollblock

1. Zustand
2. Datenstack
3. Flags
4. Elternprozess
5. Geschwister- und Kindprozesse

Aufgabe 5. Prozesse und Threads

Ein Thread ist ein "leichtgewichtiger" Prozess. Ein Prozess kann aus einem oder mehreren Threads bestehen. Mehrere Threads eines Prozesses teilen sich die Resourcen des Prozesses. Der Kontextwechsel zwischen Threads ist einfacher als der zwischen Prozessen.

Aufgabe 6. Scheduling-Ziele

1. Garantierte Mindestzuteilung der CPU-Zeit für jeden Prozess ("Fairness")
2. Effizienz durch Auslastung der CPU
3. Minimierung der Antwortzeit

Die Ziele widersprechen sich teilweise (z.B. 3. und 1.). Deshalb werden in verschiedenen Betriebssystemen Ziele unterschiedlich stark verfolgt.

Aufgabe 7. SJF und FCFS

	Vorteile	**Nachteile**
Shortest Job First	Auf schnell zu erledigende Aufgaben braucht nicht gewartet zu werden	Bei häufig erzeugten kurzen Prozessen werden lange Prozesse niemals ausgeführt. Die vorherige Kenntnis oder auch Schätzung der Laufzeit eines Prozesses ist nicht realistisch.
First Come First Serve	Einhaltung des Fairness-Prinzips	Unter Umständen kommt es durch Warten auf längere Prozesse zu schlechten mittleren Ausführungszeiten

Beide Scheduling-Verfahren sind Non-preemptive, was normalerweise nachteilig ist.

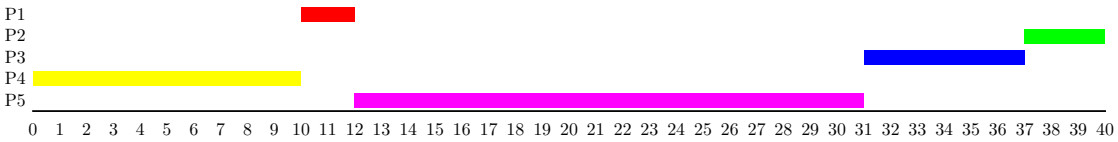
Aufgabe 8. RR und FCFS

Round Robin erweitert FCFS um Zeitscheiben. Dadurch kann die CPU-Zeit unabhängig von der Reihenfolge der Erzeugung effizient auf die Prozesse verteilt werden.

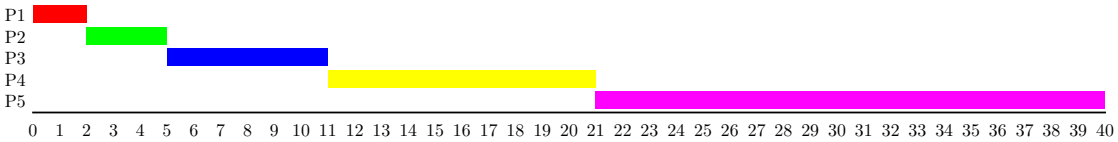
Aufgabe 9. Preemptive oder Non-preemptive

FCFS:	Non-preemptive
SJF:	Non-preemptive
RR:	Preemptive
PS:	Non-preemptive

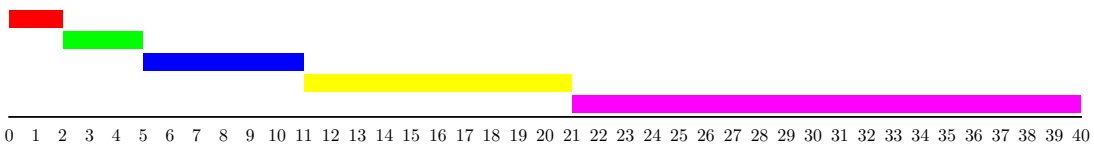
Aufgabe 10. Scheduling für P1 bis P5



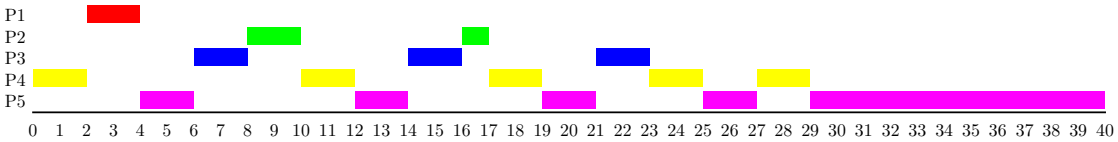
First Come First Serve



Shortest Job First

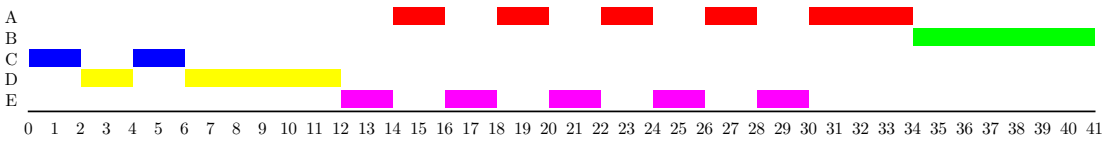


Shortest Remaining Time Next



Round Robin

Aufgabe 11. Scheduling mit Priorität



Priority Scheduling