**Projektbericht**

Alle Tests sind in der Tests.csv Datei abgespeichert und können mit z.B. [Rons Editor](https://www.ronsplace.ca/Products/RonsEditor) übersichtlich geöffnet werden.

Insgesamt wurden vier Tests mit zwei unterschiedlichen Eingabedaten durchgeführt. FCFS, SJF und PSJF wurden pro Test einmal durchlaufen. RR wurde jeweils vier Mal durchlaufen mit jeweils anderen Zeitscheiben (1, 2, 3, 10).

**Kurze Beschreibung der Implementierungen:**   
FCFS:   
Zuerst werden alle Prozesse nach der Ankunftszeit sortiert. Anschließend wird kontrolliert, ob der Prozess schon existiert. Danach werden, Mithilfe der aktuellen Zeit, die Attribute des Prozesses gesetzt und die aktuelle Zeit dann auf die Endzeit des Prozesses gesetzt.

SJF:  
Zuerst werden alle Prozesse nach der Ankunftszeit und Laufzeit sortiert. Danach werden diese Ähnlich wie in FCFS abgearbeitet. Sollte ein Prozess beendet, aber noch kein Nachfolger angekommen sein, wird eine Zeiteinheit gewartet.

RR:  
Zuerst werden alle Prozesse nach der Ankunftszeit sortiert. Bei jedem Durchlauf werden die angekommenen Prozesse in einer separaten Liste gespeichert. Von dieser Liste wird immer das erste Element genommen und die Zeitscheibe wird darüber laufen gelassen. Sobald diese abläuft, wird der Prozess wieder an das Ende der Liste gespeichert und der nächste Prozess wird bearbeitet. Sollte die Laufzeit der Zeitscheibe länger sein, als die restliche verbleibende Zeit des Prozesses, wird nur die restliche Zeit bearbeitet. Sollten alle aktuellen Prozesse beendet sein, aber noch kein Nachfolger angekommen sein, wird eine Zeiteinheit gewartet.

PSJF:   
Zuerst werden alle Prozesse nach Ankunftszeit sortiert. Alle Prozesse, die zur aktuellen Zeit angekommen sind, werden in einer separaten Liste der "laufenden Prozesse" hinzugefügt. Von dieser Liste wird immer der Prozess mit der kürzesten verbleibenden Laufzeit ausgewählt und bearbeitet. Sollte ein Prozess beendet sein, wird er aus der Liste der laufenden Prozesse entfernt und die benötigten Attribute werden berechnet und gesetzt. Sollte kein Prozess laufen oder alle aktuellen Prozesse beendet sein, aber noch kein Nachfolger angekommen sein, wird eine Zeiteinheit gewartet.

**Auswertung der Tests:**

|  |  |  |
| --- | --- | --- |
| Prozess | Ankunftszeit | Laufzeit |
| 1 | 0 | 9 |
| 2 | 15 | 4 |
| 3 | 8 | 13 |
| 4 | 10 | 3 |
| 5 | 12 | 7 |

|  |  |  |
| --- | --- | --- |
| Prozess | Ankunftszeit | Laufzeit |
| 1 | 3 | 5 |
| 2 | 0 | 10 |
| 3 | 5 | 2 |
| 4 | 9 | 7 |
| 5 | 13 | 3 |

**Eingabewerte:**

Test1 mit Umschaltzeit=0; Eingabedaten 1 Test3 mit Umschaltzeit=0; Eingabedaten 2  
Test2 mit Umschaltzeit=2; Eingabedaten 1 Test4 mit Umschaltzeit=2; Eingabedaten 2

**Geschwindigkeit:**

RR2 (30) -> RoundRobin mit Zeitscheibe = 2 mit Endzeitpunkt bei 30 Zeiteinheiten

Test1: Alle Implementierungen gleich schnell  
Test2: FCFS/SJF gleich schnell (44), RR10 (46), PSJF (58), RR3 (62), RR2 (74), RR1 (106)  
Test3: Alle Implementierungen gleich schnell  
Test4: FCFS/SJF/RR10 gleich schnell (35), RR3 (47), PSJF (53), RR2 (55), RR1 (79)

Test1 und Test3 arbeiteten beide mit Umschaltzeit=0. Dadurch hat jede Implementierung nur solange gebraucht, wie auch die Laufzeit der Prozesse war. Damit waren alle gleich schnell.  
FCFS und SJF sind auch in Test2 und Test4 gleich schnell. Der Grund dafür ist, dass die Anzahl der „Umschaltungen“ (Wechsel der Prozesse) gleich ist und die Laufzeit der Prozesse auch.  
In Test4 ist RR10 ebenfalls so schnell, weil die max. Laufzeit der Prozesse 10.0 Zeiteinheiten war. Damit wurde es eigentlich vom Ablauf gleich wie FCFS ausgeführt, auch wenn der Code eigentlich anders funktioniert. Bei diesen Beispielen teilen sich also FCFS, SJF und RR10 die besten Geschwindigkeiten. Danach folgt RR3 und PSJF mit ungefähr gleicher Performance. Den vorletzten Platz belegt RR2 und den letzten RR1.  
Mit Umschaltzeit performen natürlich die Prozesse, die oft umschalten am schlechtesten (RR2, RR1). Dass RR10 so gut abschneidet, oder auch andere RR mit sehr großer Zeitscheibe, liegt an meiner Implementierung. Wenn ein Prozess nur z.B. 3s braucht, bricht die Zeitscheibe nach 3s ab, obwohl sie noch weiterlaufen könnte. Es wäre aber nicht so „vorteilhaft“ das anders zu implementieren also halte ich es so für richtig.

**Durchschnittliche Verweilzeit:**

Test1: PSJF (11.2), SJF (15,2), RR1 (15,4), FCFS/RR10 (15,8), RR2 (16,6), RR3 (17,4)  
Test2: SJF (14,2), FCFS (19,8), RR10 (21,4), PSJF (22,4), RR3 (30,6), RR2 (44,2), RR1 (68,2)  
Test3: PSJF (10,0), SJF (11,2), FCFS /RR10 (12,6), RR2 (13,6), RR3 (14,0), RR1 (14,4)  
Test4: SJF (14,8), FCFS/RR10 (16,6), PSJF (25,2), RR3 (28,0), RR2 (34,6), RR1 (51,4)

Bei Test1, Test3 und Test4 ist wieder zu erkennen, dass sich FCFS und RR10 gleich verhalten, da die Umschaltzeit = 0 ist. Im Gesamten sind die RR1-3 die Verlierer. Dort verweilt ein Prozess im Durchschnitt am längsten. Wohingegen PSJF und SJF immer an der Spitze sind. In Test2 ist PSJF an vierter Stelle, aber mit wenig Differenz. Eine hohe Verweilzeit ist nicht unbedingt schlecht, sondern kann bedeuten, dass die Bearbeitungszeit fairer auf die Prozesse aufgeteilt wird.

**Durchschnittliche Wartezeit:**

Test1: PSJF (4,0), SJF (8,0), RR1 (8,2), FCFS/RR10 (8,6), RR2 (9,4), RR3 (10,2)  
Test2: SJF (7,0), FCFS (12,6), RR10 (14,2), PSJF (15,2), RR3 (23,4), RR2 (37,0), RR1 (61,0)  
Test3: PSJF (4,6), SJF (5,8), FCFS/RR10 (7,2), RR2 (8,2), RR3 (8,6), RR1 (9,0)  
Test4: SJF (9,4), FCFS/RR10 (11,2), PSJF (19,8), RR3 (22,6), RR2 (29,2), RR1 (46,0)

Bei Test1 und Test3 verhalten sich FCFS und RR10 mit der gleichen Erklärung wieder gleich. Bei den Tests ohne Umschaltzeit dominiert PSJF, während bei den Tests mit Umschaltzeit SJF gewinnt. SJF schneidet bei allen Tests immer gut bis sehr gut ab. Das liegt daran, dass durch die Sortierung nach Laufzeit die kurzen Prozesse nicht auf die langen Prozesse warte müssen, sondern nur umgekehrt. Also ein Prozess muss im worst case so lange warten wie die zweit längste Laufzeit ist und nicht auf die längste Laufzeit. RR ist hier auch wieder der „Verlierer“, da durch das dauerhafte durchlaufen der Zeitscheibe jeder Prozess mal warten muss.

**Durchschnittliche Reaktionszeit:**

Test1: PSJF (0,4), RR1 (1,4), RR2 (2,8) RR3 (4,2), SJF (8,0) FCFS/RR10 (8,6)  
Test2: PSJF (3,6), RR1 (6,6), SJF (7,0) RR2 (8,6), RR3 (9,0), FCFS (12,6), RR10 (12,8)  
Test3: RR1 (1,8), PSJF (2,2), RR2 (4,2), RR3 (4,8), SJF (5,8), FCFS/RR10 (7,2)  
Test4: RR1 (6,6), PSJF (9,0), SJF (9,4), RR2 (10,0), RR3 (10,4), FCFS/RR10 (11,2)

Bei Test1 und Test3 verhalten sich FCFS und RR10 mit der gleichen Erklärung wieder gleich, performen jedoch in jedem Test am schlechtesten. Das liegt daran, dass theoretisch die Prozesse umgekehrt der Sortierung des SJF ankommen können. Dies würde bedeuten, dass der kürzeste Job auf alle warten müsste. Bei SJF, vor allem bei PSJF würde dies durch die Sortierung im Normalfall vereitelt werden. RR1-3 liegen im Mittelfeld. Logisch natürlich, da hier dauerhaft durchgewechselt wird. Durch die Ankunft der Prozesse ergibt sich bei R1-3 eine durchschnittliche Reaktionszeit von ca. 1,5x der Zeitscheibe.

**Auswahl:**

Für **arbeitsintensive Prozesse**, die einfach bearbeitet werden müssen und hauptsächlich das Ergebnis zählt würde ich zu FCFS/SJF greifen. Diese sind in ihrer Geschwindigkeit ungeschlagen. FCFS mag vielleicht ein bisschen „fairer“ durch die „wer zuerst kommt mahlt zuerst“ Regelung, aber für mich persönlich wäre der Durchsatz an abgeschlossenen Prozessen wichtiger. Auf die gesamte Zeit ist dieser wieder gleich, aber weil die kürzeren Prozesse zuerst abgearbeitet sind, kann man mit deren Ergebnissen theoretisch schon weiterarbeiten.  
Für eine **I/O Umgebung**, wo manche Prozesse einfach in den Hintergrund geschoben werden müssen, da Eingaben vom User oder Ausgaben zum User übermittelt werden MÜSSEN, um eine angemessene Interaktion zu schaffen, würde ich mich für PSJF entscheiden. Egal ob rechenintensive Jobs oder eine Menge Prozesse das Problem sind, durch das unterbrechen des PSJF wird häufig garantiert, dass die meist sehr kleinen und kurzen I/O Prozesse bevorzugt werden und der User kaum Einschränkungen bemerkt. Bei FCFS/SJF würden diese kleinen immer wiederkehrenden Prozesse im Stau verloren gehen und mit RR würde eine Zeitscheibe super fair über alle Prozesse gleiten, aber eben auch über die für den Moment unwichtigen. RR wäre aber tatsächlich meine zweite Wahl für diese Art von Prozessen. Über die Dauer der Zeitscheibe kann ich nichts sagen, da ich überhaupt keine Einschätzung treffen könnte. Vielleicht gibt es ja auch eine dynamische Zeitscheibe, die sich dem Workload anpassen würde.