Übung 4

Abgabe: 07.06.2018 8:30

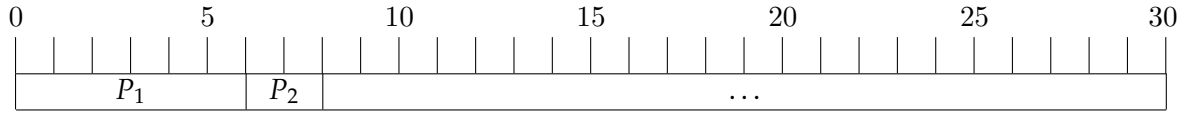
Aufgabe 4.1: Einfache Scheduling-Strategien (1.5+1.5+1.5+1.5 = 6 Punkte)

In der Vorlesung haben Sie einige Scheduling-Strategien kennen gelernt (FIFO/LIFO/SPT/SRPT). In den folgenden Aufgaben geht es darum diese auf Daten anzuwenden.

In der Warteschleife (Queue) seien N Jobs mit Bearbeitungszeiten $t_1 ... t_N$. Index n gibt die Reihenfolge des Ankommens in der Queue an: D.h. Job P_n (mit Bearbeitungszeit t_n) kommt vor Job P_{n+1} (mit Bearbeitungszeit t_{n+1}) an. Folgende Jobs warten in der Queue darauf, abgearbeitet zu werden:

Job	Bearbeitungszeit (t_n)
P_1	6
P_2	2
P_3	7
P_4	3
P_5	4
P_6	6

Visualisieren Sie die Abarbeitung der Jobs bitte in einem Format ähnlich zu dem Folgenden:



Sie können hierzu die Vorlagen auf der letzten Seite dieser Übung benutzen. Bitte beschriften Sie die ausgefüllten Vorlagen mit der entsprechenden Aufgabennummerierung.

a) Wie werden die Jobs nach FIFO abgearbeitet und wie groß ist die durchschnittliche Wartezeit?

b) Wie werden die Jobs nach LIFO abgearbeitet und wie groß ist die durchschnittliche Wartezeit?

c) Wie werden die Jobs nach SPT abgearbeitet und wie groß ist die durchschnittliche Wartezeit?

d) Nun haben wir zusätzlich zur Bearbeitungszeit noch eine Ankunftszeit. Das heißt die Jobs können erst ab dem Zeitpunkt abgearbeitet werden, ab dem sie auch angekommen sind.

Job	Ankunftszeit	Bearbeitungszeit (t_n)
P_1	00	6
P_2	01	2
P_3	02	7
P_4	03	3
P_5	04	4
P_6	11	2

Wie werden die Jobs jetzt nach SRPT (Shortest Remaining Time First) abgearbeitet und wie groß ist die durchschnittliche Wartezeit?

Aufgabe 4.2: Earliest Deadline First (5+1+1 = 7 Punkte)

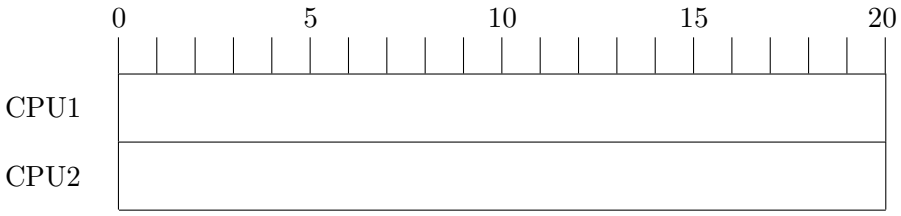
Gegeben sei ein Zwei-Prozessor-System sowie die folgenden Prozesse:

Prozess	Ankunftszeitpunkt	Bedienzeit	Deadline
P_1	0	7	19
P_2	1	6	13
P_3	2	7	13
P_4	4	3	7
P_5	5	3	12
P_6	7	3	14
P_7	11	3	16

a) Stellen Sie den resultierenden EDF-Schedule dar. Gehen Sie dabei von den folgenden Annahmen aus:

- Die Zeit für einen Kontextwechsel werde vernachlässigt.
- Ein Ankunftszeitpunkt von t bedeute, dass ein Prozess auch bereits zur Zeit t eine CPU belegen kann.
- Kommt ein Prozess an und eine der CPUs ist aktuell ungenutzt, wird er dieser zugeordnet. Sind sogar beide ungenutzt, wird der Prozess CPU 1 zugeordnet.
- Kommt ein Prozess an und beide CPUs sind aktuell belegt, werden die Deadlines verglichen. Die Deadlines sind als absolute Zeiten angegeben, nicht als relativ zum Startzeitpunkt zu interpretieren.
  - Liegt die Deadline des ankommenden Prozesses später als die Deadlines beider in Bearbeitung befindlicher Prozesse, so wird die Zuordnung zu einer CPU so lange verzögert, bis ein Prozess die CPU freigibt und kein anderer wartender Prozess eine frühere Deadline hat.
  - Liegt die Deadline des ankommenden Prozesses früher als die Deadline eines der in Bearbeitung befindlichen Prozesse, so wird dieser Prozess verdrängt.
  - Liegt die Deadline des ankommenden Prozesses früher als die Deadlines beider in Bearbeitung befindlicher Prozesse, so wird der mit der späteren Deadline verdrängt. Haben beide in Bearbeitung befindlichen Prozesse die gleiche Deadline, so wird der Prozess auf derjenigen CPU verdrängt, der aktuell weniger Prozesse zugeordnet sind. Sind beiden CPUs gleich viele Prozesse zugeordnet, erfolgt die Verdrängung auf CPU 1.
- Ist ein Prozess einmal einer CPU zugeordnet worden, wird er nur noch von dieser bearbeitet.
- Ein Prozess wird immer vollständig abgearbeitet, auch wenn er seine Deadline bereits verpasst hat.

Zur Darstellung der CPU-Belegungszeiten können Sie z.B. eine graphische Lösung wie folgt vornehmen:



Sie können hierzu die Vorlage auf der letzten Seite dieser Übung benutzen. Bitte lassen Sie sich nicht davon irritieren, dass keine Periode angegeben ist – Strategien wie EDF lassen sich natürlich auch auf nicht-periodische Prozesse anwenden und machen immer dann Sinn, wenn Deadlines einzuhalten sind.

b) Wieviele Deadlines werden verletzt? Welche maximale Verspätung (Zeiteinheiten, um die ein Prozess nach Ablauf seiner Deadline beendet wurde) eines Prozesses tritt auf?

c) Angenommen, ein Prozess sei nicht an eine CPU gebunden, sondern kann innerhalb einer Zeiteinheit zwischen den beiden CPUs verschoben werden (d.h.: die Ziel-CPU hätte eine Zeiteinheit Leerlauf). Verbessert oder verschlechtert dies die obige Situation bzgl. Deadlineverletzungen und maximaler Verspätung?

Aufgabe 4.3: Multilevel Feedback Queueing (7 Punkte)

Gegeben sei ein Multilevel Feedback Queueing (MLFQ) mit vier Prioritätsklassen. Jeder Klasse ist eine eigene Warteschlange, ein eigenes Quantum und eine Bedienstrategie zugeordnet:

Klasse	Quantum	Priorität	Bedienstrategie
0	1	höchste	*FIFO*
1	4		RR_2
2	16		RR_6
3	∞	niedrigste	*FIFO*

Wenn das Quantum eines Prozesses abgelaufen ist, wird er an das Ende der Warteschlange mit nächstniedriger Priorität gestellt. Neuen Prozessen ist eine bestimmte Priorität zugeordnet. Sie werden an das Ende der Warteschlange ihrer Prioritätsklasse angehängt. Es wird am Ende jeder Zeiteinheit überprüft, welcher Prozess am Anfang der nicht-leeren Warteschlange mit der höchsten Priorität steht und dieser dann im nächsten Schritt bearbeitet. Wenn dadurch ein Prozess mit niedrigerer Priorität unterbrochen wird, bevor er sein Quantum aufgebraucht hat, wird er zurück an den Anfang seiner bisherigen Warteschlange gestellt. Dieser Prozess nutzt bei seiner nächsten Aktivierung allerdings nur das Restquantum auf (darf nicht das volle Quantum noch einmal nutzen), bevor er in die nächstniedrigere Klasse verschoben wird. In den unterschiedlichen Warteschlangen werden unterschiedliche Bedienstrategien verwendet: FIFO bzw. Round Robin (RR). Beachten Sie, dass es bei den RR-Strategien neben dem (MLFQ-)Quantum, welches in obiger Tabelle angegeben ist, auch noch ein RR-Quantum gibt (als Index am RR angegeben), welches bei der Bearbeitung der Prozesse innerhalb der RR-Warteschlangen berücksichtigt werden muss.

Folgende Prozesse sollen betrachtet werden:

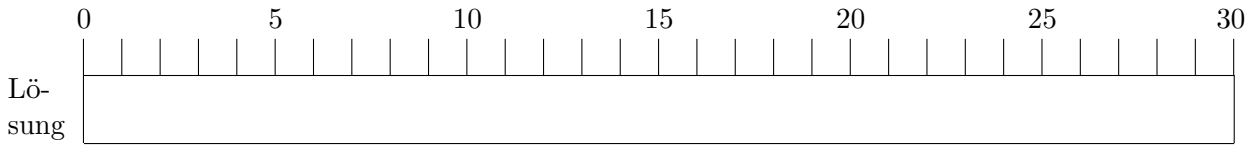
Prozess	Ankunftszeit	Klasse	Bedienzeit
A	0	1	7
B	0	2	6
C	1	0	1
D	2	1	2
E	5	3	17
F	10	1	3
G	13	0	5
H	15	1	3
I	16	1	3

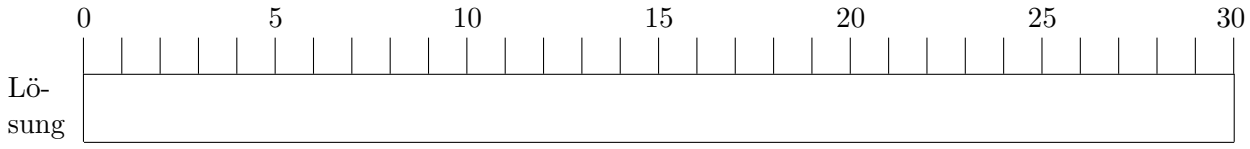
Ein Prozess, der zum Zeitpunkt t ankommt, wird erst ab dem $(t+1)$-ten Zeitpunkt berücksichtigt, d.h. er kann frühestens eine Zeiteinheit nach seiner Ankunft die CPU verwenden. Kontextwechsel werden vernachlässigt.

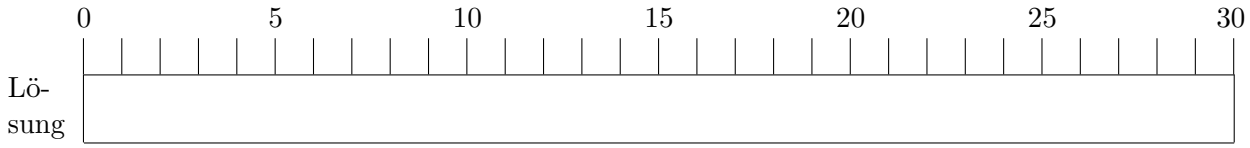
Geben Sie für die ersten 20 Zeiteinheiten an, welche Prozesse sich in welcher Warteschlange befinden (in der korrekten Reihenfolge) und welchem Prozess Rechenzeit zugeteilt wird. Verwenden Sie dabei das folgende Tabellenformat für Ihre Angaben:

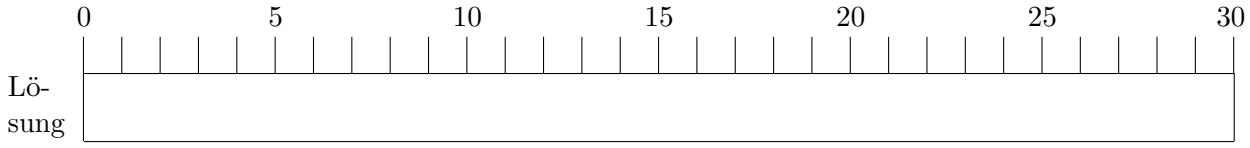
t	Kl. 0 FIFO(1)	Kl. 1 $RR_2(4)$	Kl. 2 $RR_6(16)$	Kl. 3 FIFO	Incoming	Running
0	-	-	-	-	A(7),B(6)	-
1	...	...	...	...	...	...
2	...	...	...	...	...	...

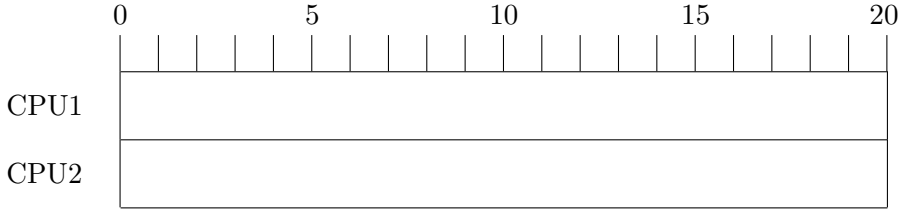
Sie können hierzu die Vorlage auf der letzten Seite dieser Übung benutzen.


0
5
10
15
20
25
30
Lö-
sung


0
5
10
15
20
25
30
Lö-
sung


0
5
10
15
20
25
30
Lö-
sung


0
5
10
15
20
25
30
Lö-
sung


0
5
10
15
20
CPU1
CPU2

t	Kl. 0 FIFO(1)	Kl. 1 $RR_2(4)$	Kl. 2 $RR_6(16)$	Kl. 3 FIFO(-1)	Incoming	Running
0	-	-	-	-	A(7),B(6)	-
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						