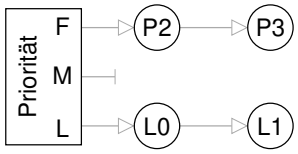
Aufgabe V2: Bereitmenge und Aufgreifstrategie

Ein fiktives Betriebssystem unterstützt die drei Prioritäten **F**lott, **M**oderat und **L**ungern. Die Leerlaufprozesse haben die niedrigste Priorität L, alle anderen Prozesse entweder F oder M. Prozesse mit gleicher Priorität sind nach dem Zeitpunkt geordnet, an dem sie in die Bereitmenge gelangen: Wer zuerst kommt, mahlt zuerst.
Dieses Betriebssystem arbeitet auf einem Rechner mit zwei Prozessoren (CPU0, CPU1). Es existieren vier normale Prozesse (P0–P3) mit flotter Priorität, die auf jedem Prozessor ablaufen können. Die beiden Leerlaufprozesse (L0, L1) sind dem jeweiligen Prozessor fest zugeordnet. Prozesse erhalten nur Rechenzeit, wenn kein Prozess mit einer höheren Priorität bereit steht. Zwischen Prozessen mit gleicher Priorität wird nach Ablauf einer Zeitscheibe umgeschaltet.
Die folgende Tabelle beschreibt den Systemzustand zu Beginn der Beobachtung. P0 und P1 rechnen auf CPU0 bzw. CPU1, in der Bereitmenge steht P2 vor P3. Unter der Tabelle ist eine chronologische Liste von Ereignissen aufgeführt, die die Prozessorzuteilung oder die Bereitmenge ändern können. Ereignisse wirken sich sofort auf alle Prozessoren aus.

CPU0	**CPU1**	**Bereitmenge**
P0	P1	{P2, P3; L0, L1}
...	...	...



1. Zeitscheibe CPU0 läuft ab
2. Zeitscheibe CPU1 läuft ab
3. P2 vermindert eigene Priorität
4. Zeitscheibe CPU1 läuft ab
5. Zeitscheibe CPU0 läuft ab
6. P1 vermindert Priorität von P3
7. P0 endet
8. Zeitscheibe CPU1 läuft ab
9. P1 erhöht Priorität von P3
10. P1 endet
11. P2 endet
12. P3 endet

a) Ergänzen Sie die Tabelle um eine Zeile für jedes Ereignis. Tragen Sie jeweils den Zustand nach den Änderungen ein, die sich durch das Ereignis ergeben. Prozesse mit moderater Priorität können Sie zum Beispiel durch ein Minuszeichen kenntlich machen ($P2^-$).

b) Bei welchen Ereignissen würde sich das System anders verhalten, wenn sie sich nicht sofort auf den anderen Prozessor auswirkten?

#	**CPU0**	**CPU1**	**Bereitmenge**
0	P0	P1	{P2, P3; L0, L1}
1	P2	P1	{P3, P0; L0, L1}
2	P2	P3	{P0, P1; L0, L1}
3	P0	P3	{P1; $P2^-$; L0, L1}
4	P0	P1	{P3; $P2^-$; L0, L1}
5	P3	P1	{P0; $P2^-$; L0, L1}
6	P0	P1	{$P2^-$, $P3^-$; L0, L1}
7	$P2^-$	P1	{$P3^-$; L0, L1}
8	$P2^-$	P1	{$P3^-$; L0, L1}
9	P3	P1	{$P2^-$; L0, L1}
10	P3	$P2^-$	{L0, L1}
11	P3	L1	{L0}
12	L0	L1	{}

Mit den ersten beiden Ereignissen rotieren die Prozesse durch, wie man es intuitiv erwartet. Im dritten Schritt reduziert P2 seine Priorität. Dadurch verliert er sofort den Prozessor und wandert an das Ende der Bereitmenge, abgesehen von den Leerlaufprozessen. Solange noch mindestens zwei Prozesse mit höherer Priorität existieren, wird P2 keine Rechenzeit mehr erhalten. Das ist bei den folgenden beiden Ereignissen zu beobachten.
In Schritt 6 wird die Priorität von P3 vermindert, der dadurch ebenfalls den Prozessor verliert. In dieser Situation hätte das Ablaufen von Zeitscheiben keinen Effekt mehr, die beiden verbleibenden Prozesse mit flotter Priorität sind fest auf den Prozessoren gebucht, auf denen sie gerade rechnen.
Mit dem Ende von P0 wird ein Prozessor frei, auf dem sich anschließend die beiden Prozesse mit moderater Priorität abwechseln könnten, wenn Zeitscheiben ablaufen. Das ist allerdings laut Aufgabenstellung nicht der Fall. Statt dessen übernimmt P3 mit dem nächsten Ereignis den Prozessor, da sich seine Priorität wieder erhöht. P0 verschwindet aus der Betrachtung.
Während sich die restlichen Prozesse nach und nach beenden, kommen auch die Leerlaufprozesse zum Zuge. Sie sind den Prozessoren fest zugeordnet, deshalb nimmt sich CPU1 den Prozess L1, obwohl in der Tabelle L0 vor L1 steht.

In Schritt 6 reduziert P1 auf CPU1 die Priorität von P3, der gerade auf CPU0 abläuft. Wenn CPU1 das Ereignis nicht sofort an CPU0 meldet, läuft $P3^-$ zunächst weiter und P0 bleibt in der Bereitmenge.

In Schritt 9 erhöht P1 auf CPU1 die Priorität von P3, der gerade in der Bereitmenge liegt. Wenn CPU1 das Ereignis nicht sofort an CPU0 meldet, läuft $P2^-$ zunächst weiter und P3 bleibt in der Bereitmenge.