Übungsblatt 3

Abgabe: 13. Mai 2019

Hinweis: für die Aufgaben 3.2 bis 3.4 finden sie im Lernraum im Abschnitt „Übungsblätter“ bei Interesse Templates für die Lösungen in LaTeX- und Word-Format.

Aufgabe 3.1: IPC und Threads (2+1+1+1 = 5 Punkte)

a) Pipes und Shared Memory sind zwei Konzepte zum Austausch von Informationen zwischen Prozessen. Stellen Sie die Vor- und Nachteile der beiden Konzepte gegenüber.

b) In der Vorlesung wurde gesagt, dass named Pipes sehr ähnlich zu Dateien sind. Worin bestehen die wesentlichen Unterschiede zwischen Dateien und benannten Pipes?

c) Was sind die wesentlichen Unterschiede zwischen Prozessen und Threads?

d) Warum ist es im Allgemeinen nicht sinnvoll, zu viele Threads innerhalb eines Prozesses zu verwenden?

Aufgabe 3.2: Einfache Scheduling-Strategien (1 + 1 + 1.5 + 1.5 = 5 Punkte)

Gegeben sei ein Rechnersystem mit einer CPU und 6 Prozessen $P_1, \ldots, P_6$. Die folgende Tabelle gibt an, zu welchen Zeitpunkten die Prozesse das System betreten sowie für wie viele Zeiteinheiten sie die CPU benötigen:

Prozess	**Ankunftszeitpunkt**	**Bedienzeit**
P_1	0	4
P_2	1	2
P_3	2	7
P_4	3	4
P_5	7	2
P_6	11	5

Ein **Ankunftszeitpunkt** von t bedeutet, dass der Prozess zu diesem Zeitpunkt bereits im Zustand `ready` auf Zuteilung der CPU wartet und direkt vom Scheduler berücksichtigt werden kann, also noch in der gleichen Zeiteinheit die CPU nutzen kann. Der Kontextwechsel zwischen der Bearbeitung zweier Prozesse soll vernachlässigt werden. Hat ein Prozess für die unter **Bedienzeit** angegebenen Zeiteinheiten die CPU belegt, verlässt er sofort das System, ohne weitere CPU-Zeit zu beanspruchen.

Geben Sie für jede der im Folgenden aufgeführten Scheduling-Strategien an, welcher Prozess wann die CPU belegt und wie groß die durchschnittliche Wartezeit ist.

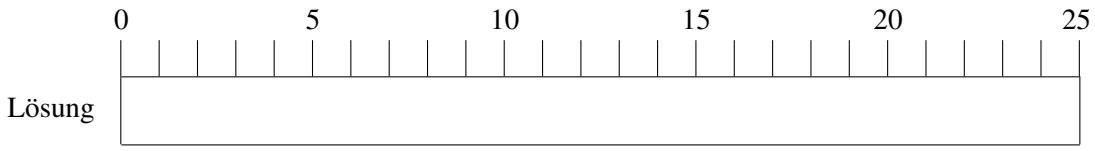
a) FIFO

b) SPT

c) SRPT

d) HRN

Die CPU-Belegungszeiten können Sie z.B. graphisch anhand des folgenden Rasters darstellen:



Aufgabe 3.3: Scheduling und I/O-Nutzung (3,5 + 1,5 = 5 Punkte)

Gegeben sei ein Rechnersystem mit einer CPU und 3 Prozessen $P_1, \ldots, P_3$. In der folgenden Tabelle ist für diese Prozesse angegeben, zu welchen Zeitpunkten sie das System betreten und für wie viele Zeiteinheiten sie die CPU benötigen:

Prozess	Ankunftszeitpunkt	Bedienzeit	Ausführungsmuster
P_1	0	10	PPPPPPPPPP
P_2	1	11	PPPPPPPPPPP
P_3	2	4	PIIIIPIIIIPIIIIP

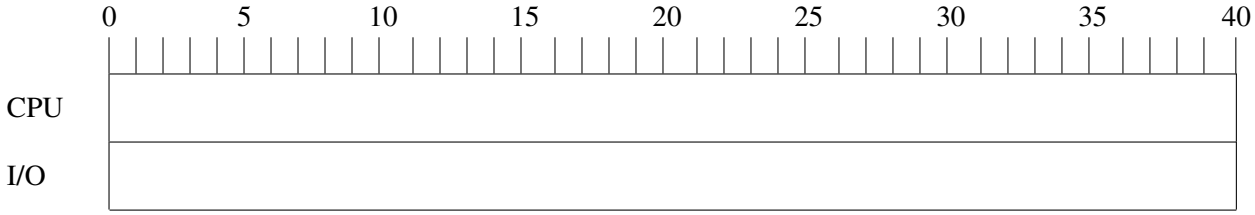
Ein **Ankunftszeitpunkt** von t bedeutet, dass der Prozess zu diesem Zeitpunkt bereits im Zustand `ready` auf Zuteilung der CPU wartet und direkt vom Scheduler berücksichtigt werden kann, also noch in der gleichen Zeiteinheit die CPU nutzen kann. Der Kontextwechsel zwischen der Bearbeitung zweier Prozesse soll vernachlässigt werden. Hat ein Prozess für die unter **Bedienzeit** angegebenen Zeiteinheiten die CPU belegt, verlässt er sofort das System, ohne weitere CPU-Zeit zu beanspruchen.

Darüber hinaus enthält die Tabelle das **Ausführungsmuster** der Prozesse. Dieses gibt an, wie sich die Prozesse während ihrer Ausführung verhalten. Ein *P* steht für 'Processing', d.h. der Prozess ist für die jeweilige Zeiteinheit ausführbereit und kann die CPU nutzen. (Prozesse P_1 und P_2 sind durchgehend rechenbereit und können für 10 bzw. 11 Zeiteinheiten direkt die CPU nutzen, wenn sie sie zugeteilt bekommen.) Das *I* steht für 'I/O-Operation', d.h. ein Prozess benötigt für eine Zeiteinheit Zugriff auf ein I/O-Gerät, bevor er weiter ausgeführt werden kann. (Prozess P_3 benötigt nach jeweils einer Zeiteinheit CPU-Nutzung für vier Zeiteinheiten Zugriff auf ein I/O-Gerät, bevor er wieder die CPU nutzen kann. Er kann seine Bedienzeit also nicht am Stück abarbeiten.) Es werde angenommen, dass das I/O-Gerät immer zur Verfügung steht und ohne Verzögerung genutzt werden kann. Greift ein Prozess auf das I/O-Gerät zu, gibt er sofort die CPU frei und lässt den Rest seiner zugeteilten CPU-Zeit verfallen.

a) Welcher Prozess belegt wann die CPU und wann das I/O-Gerät? Geben Sie die Lösung jeweils unter Verwendung der folgenden Scheduling-Strategien an:

   i) FIFO
   ii) Round Robin mit Quantum 5
   iii) Round Robin mit Quantum 1. Gibt ein Prozess P_x die CPU frei und ein anderer Prozess P_y zeitgleich das I/O-Gerät, so wird Prozess P_x vor P_y in die Warteschlange der CPU eingereiht. Ebenso werden Prozesse, die neu erzeugt werden, gegenüber Prozessen, die zum gleichen Zeitpunkt die CPU freigeben, bevorzugt behandelt.

   Zur Darstellung der CPU- und I/O-Belegungszeiten können Sie z.B. eine graphische Lösung anhand des folgenden Rasters vornehmen:



b) Welche der drei Scheduling-Strategien arbeitet die Prozesse insgesamt am schnellsten ab? Ist dies bei allen Ausführungsmustern so? Wenn ja: begründen Sie Ihre Antwort. Wenn nein: geben Sie ein Gegenbeispiel mit drei Prozessen an, bei dem eine der beiden anderen Scheduling-Strategien früher alle Prozesse abgearbeitet hat; in Ihrer Lösung dürfen bei Bedarf alle Prozesse das I/O-Gerät nutzen.

Aufgabe 3.4: Multilevel Feedback Queueing (5 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	2	5
B	0	1	8
C	1	0	2
D	4	1	2
E	5	3	15
F	7	1	4
G	13	0	5
H	15	0	3
I	18	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(5),B(8)	-
1	...	...	...	...	...	...
2	...	...	...	...	...	...