Übung 3

Abgabe: 31.05.2018 8:30

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

a) In der Vorlesung haben Sie unter anderem Pipes und Shared Memory als zwei Konzepte zum Austausch von Informationen zwischen Prozessen kennengelernt. 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 Unterschied zwischen Dateien und benannten Pipes?

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

d) Wenn User-Threads verwendet werden, gibt es dann einen Stack pro Thread oder einen Stack für den gesamten Prozess? Wie sieht die Situation aus, wenn Kernel-Threads verwendet werden? Begründen Sie Ihre Antworten.

Aufgabe 3.2: Interprozess-Kommunikation / Prozess-Synchronisation (1+2 = 3 Punkte)

Betrachten Sie das C Programm ipc.c, welches Sie im Anhang zu diesem Übungsblatt finden.

a) Geben Sie eine kurze Beschreibung der wesentlichen Aspekte des vorliegenden Programms. Welche Art der Interprozess-Kommunikation wird hier genutzt? Hierbei muss nicht jede Zeile im Einzelnen beschrieben werden und auf Details wie Includes oder warum beispielsweise shmget die Argumente (IPC_PRIVATE, SEGSIZE, IPC_CREAT|0644) erhält, muss nicht eingegangen werden. (Die Musterlösung enthält 8 kurze Sätze.)

b) Kompilieren Sie das Programm und führen Sie es mehrfach aus. Was fällt auf? Nennen Sie zwei Auffälligkeiten und geben Sie jeweils eine Begründung für die Auffälligkeit. Eine der genannten Auffälligkeit sollte in Zusammenhang mit Kapitel 5 der Vorlesung "Prozess-Synchronisation" stehen. (Der Teil mit dieser Auffälligkeit bringt bei korrekter Bearbeitung 1.5 Punkte.)

Aufgabe 3.3: Round Robin (7 Punkte)

Folgende acht Prozesse seien gegeben und zum Zeitpunkt 0 in der Ready-Queue in der Reihenfolge ihrer Prozessnummern eingereiht:

Job	Bearbeitungszeit (t_n)
P_1	6
P_2	13
P_3	7
P_4	3
P_5	4
P_6	9
P_7	10
P_8	11

Der Scheduler verwendet ein Zeitquantum Q, nach dessen Ablauf zum jeweils nächsten Prozess umgeschaltet wird. Die Zeiten für diese Kontextwechsel zwischen den Prozessen werden vernachlässigt.

Schreiben Sie ein Bash-Script oder C-Programm, das eine Tabelle berechnet, in der für ein gegebenes maximales ganzzahlige Zeitquantum Q (in diesem Beispiel bis zum Wert 13) angegeben wird, wann die einzelnen Prozesse (P_i) beendet wurden und wie groß die durchschnittliche Wartezeit war. Die Bearbeitungszeiten können hierbei fest gespeichert werden, sollten aber im Quelltext einfach zu ändern sein, sodass auch Schedules für andere Bearbeitungszeiten berechnet werden können.

Testen Sie das Programm anhand der obigen Prozesse. Die Ausgabe sollte wie im folgenden Beispiel aussehen:

```
#   Q  P1   P2   P3   P4   P5   P6   P7   P8  Avg.Time
#-------------------------------------------------------
    01 38   63   45   20   28   54   58   61   38.00
    .                                        .
    .                                        .
    .                                        .
    13 06   19   26   29   33   42   52   63   25.87
```