Vorlesung Systemnahe Informatik
Sommersemester 2023

Prof. Dr. Peter Martini, Dr. Matthias Frank, Lennart Buhl M.Sc.

10. Übungszettel

Ausgabe:	Montag, 19. Juni 2023.
Abgabe:	Sonntag, 25. Juni 2023
Besprechung:	In den Übungen ab Montag, 26. Juni 2023.
Hinweis:	Abgabe erfolgt freiwillig per PDF über eCampus, siehe Hinweise in Aufgabe 1 auf dem 1. Übungszettel Sie können Kontakt zu Ihrer Tutor/in aufnehmen durch E-mail an cs4+ueb-si-XX@cs.uni-bonn.de mit XX als Gruppennummer.

Aufgabe 1: Scheduling

Betrachten Sie die zwei Prozesse P_1 und P_2 mit den Zeiten

$p_1 = 50, t_1 = 25$

$p_2 = 75, t_2 = 30$

wobei p_i die Dauer einer Periode und t_i die Dauer der Bearbeitung des Prozesses P_i sei. Die Deadline d_i sei jeweils gleich der Periodendauer p_i, das heißt, ein Prozess muss abgearbeitet sein, bevor eine neue Instanz davon auftritt.

a) Berechnen Sie zunächst die CPU-Auslastung. Können diese zwei Prozesse durch das Rate-Monotonic-Scheduling (RMS)-Verfahren gescheduled werden? Begründen Sie Ihre Antwort durch ein Gantt-Diagramm.

b) Zeichnen Sie ein Gantt-Diagramm für die beiden Prozesse für das Earliest-Deadline-First (EDF)-Verfahren für eine Dauer von mindestens 200 Zeiteinheiten.

Aufgabe 2: Scheduling, RMS

Es seien die zwei Prozesse P_1 und P_2 mit den Zeiten

$p_1 = 50, t_1 = 10$

$p_2 = 55, t_2 = 40$

gegeben. Dabei sei wieder p_i die Dauer einer Periode und t_i die Dauer der Bearbeitung des Prozesses P_i. Die Deadline d_i sei jeweils gleich der Periodendauer p_i.

a) Überzeugen Sie sich durch ein Gantt-Diagramm, dass die beiden Prozesse erfolgreich durch das RMS-Verfahren gescheduled werden können.

b) Berechnen Sie die CPU-Auslastung und vergleichen Sie das Ergebnis mit dem Theorem von Liu und Layland (vgl. Vorlesung). Was stellen Sie fest?

c) Geben Sie eine oder mehrere hinreichende Bedingungen an, dass zwei Prozesse mit konstanter Perioden- und Bearbeitungsdauer und Deadlines auf den Periodenanfängen garantiert erfolgreich durch das RMS-Verfahren gescheduled werden.

Aufgabe 3: Scheduling: Philosophische Threads - Zusatzaufgabe

Betrachten wir wieder die Problemstellung der fünf Philosophen beim Abendmahl aus der Vorlesung (Beispiel 2, Kapitel 3.1). Die Ausgangssituation ist dabei wieder die folgende:

Es sitzen fünf Philosophen am Tisch und führen das bekannte Philosophenleben: Denken und Spaghetti essen. Diese sind jedoch so glitschig, dass jeder Philosoph hierfür die linke und die rechte Gabel braucht. Zwischen je zwei Philosophen liegt eine Gabel (vgl. Abb. 1).

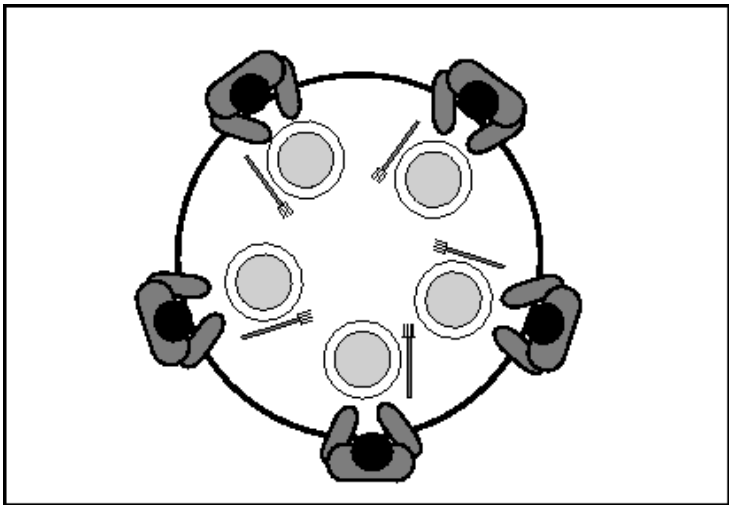

Abbildung 1: Philosophen

Implementieren Sie dieses Problem mit geeigneten Mitteln in einer Programmiersprache Ihrer Wahl, zum Beispiel in Java mit Monitoren. Die agierenden Philosophen sind hierbei als Threads zu modellieren, die Gabeln sind die genutzten Ressourcen.

a) Fangen Sie am besten mit dem Ansatz an, der in der Vorlesung (Kapitel 3, Folie 8 und 9) vorgestellt wurde: ein Philosoph, der essen möchte, nimmt sich zuerst die linke Gabel und danach die rechte. Auf jede Gabel wird (ohne vorheriges Zurücklegen) gewartet, bis sie verfügbar ist. Warum kann hier trotz der Verwendung von Monitoren ein Deadlock auftreten?

b) Erweitern Sie nun ihren ersten Ansatz so, dass ein Deadlock ausgeschlossen ist. Kommentieren Sie Ihre Änderungen, die hierfür notwendig waren. Könnte nun trotzdem einer der Philosophen verhungern, weil er nie beide Gabeln erhält?

Aufgabe 4: IT-Sicherheit - Fünfte Zusatzaufgabe

Seit einigen Jahren gibt es zusätzlich zu den normalen Übungsaufgaben für Systemnahe Informatik einige zusätzliche Aufgaben mit dem Schwerpunkt IT-Sicherheit. Diese Aufgaben gehen über den Inhalt der Vorlesung hinaus und sind somit nicht prüfungsrelevant. Vielmehr möchten wir Anwendungsgebiete vermitteln, wofür Sie die Grundlagen benötigen, die Sie in der Systemnahen Informatik lernen.
Wir versuchen, praxisnahe Aufgaben zu formulieren, an denen Sie tüfteln und herumprobieren können. Manchmal gibt es dafür auch mehrere Lösungen. Wenn Sie bei einer

Aufgabe keinen Ansatz finden, fragen Sie uns gern per E-Mail oder sprechen uns persönlich an. Wir freuen uns auf viele Ergebnisse und Ideen.

Nun zur Aufgabe: Buchverschlüsselung - Implementierung der Encodierung
In der zweiten und vierten Zusatzaufgabe haben Sie die Buchverschlüsselung verwendet, um das Chiffrat in Klartext zu überführen und die Antwort zu verschlüsseln. Entschlüsselung und Verschlüsselung können bei Buchverschlüsselung nicht mit demselben Algorithmus realisiert werden.

Entwickeln und beschreiben Sie nun die Verschlüsselung der Buchverschlüsselung als Algorithmus und programmieren diese in Pseudocode oder einer Programmiersprache Ihrer Wahl. Es soll dabei mindestens die Funktion `encode` vorhanden sein.

Überlegen Sie, wie Sie geschickt die Eingabe verwalten (Eingabe: gegebener Klartext, gegebenes "Buch" zum Verschlüsseln). Welche Sonderfälle sollten beachtet werden? (ggf. Reaktion mit Abbruch u./o. Fehlermeldung).