J. Gutenberg-Universität Mainz
Prof. Dr. Reinhold Kröger
Fabian Meyer M.Sc.

Betriebssysteme
WS 2012/13

Übungsblatt 10
Praktische Übungen

In dieser Übung wird die UNIX-Interprozesskommunikation mittels gemeinsam benutzter Speichersegmente (Shared Memory Segments) vertieft (vgl. Kap. 8.8) und auf klassische Kommunikationsprobleme (vgl. Kap. 5.3) angewendet. Zur Durchsetzung der Atomarität zusammengesetzter Operationen auf einem Speichersegment oder Teilen davon werden häufig Semaphore verwendet, da der Speicher als solcher nur die Unteilbarkeit von einzelnen Zugriffen sicherstellt.

Aufgabe 10.1:

In dieser Aufgabe wird ein Shared Memory Segment als einelementiger Kommunikationspuffer zur Realisierung einer Kommunikationsbeziehung zwischen mehreren Schreiber-Prozessen und einem Leser-Prozess verwendet. Die Schreiber-Prozesse legen Nachrichten ab, der Leser-Prozess liest auf Anforderung durch den Benutzer den Kommunikationspuffer und gibt die aktuell enthaltene Nachricht auf dem Bildschirm (über Standardausgabe) aus. Das Shared Memory Segment ist damit ein gemeinsam benutztes Betriebsmittel aller Prozesse. Zunächst erfolgt die unsynchronisierte Benutzung des Segments durch die Prozesse.

(a) Schreiben Sie ein Programm `create.c`, das ein Shared Memory Segment mit festem vorgegebenen Schlüssel und fester Größe erzeugt, die für die im folgenden betrachtete Kommunikationsaufgabe ausreichend ist.

(b) Schreiben Sie ein Programm `schreiber.c`, dem beim Aufruf über `argv[]` ein Zeichen (z.B. 'A'), eine Länge `length` (≤40) und ein Wartezeit-Parameter `sleep_time` (in sec, z.B. 1) übergeben werden. Das Programm mache sich das in (a) erzeugte Segment zugänglich. Es schreibe dann zyklisch `ITERATIONS=10` mal eine Nachricht in das Shared Memory Segment mit folgendem Aufbau: (`int pid, int i, int length, xx...x`). (Achtung: Es sind die Datentypen binär abzulegen und nicht in Zeichenketten zu wandeln!) Dabei sei `pid` die pid des ausführenden Prozesses, `i` die Nummer des aktuellen Iterationsschritts des gerade schreibenden Prozesses, `length` (s.o) die mit dem Aufruf übergebene Anzahl der folgenden Zeichen `xx...x` ist, wobei `x` für das mit dem Aufruf übergebene Zeichen steht (hierdurch können Sie die Ausgaben von verschiedenen Prozessen einfach unterscheiden). Nach der Ausgabe eines Zeichens `x` lege sich der Prozess für `sleep_time` Sekunden schlafen, bevor er das nächste Zeichen schreibt bzw. mit dem nächsten Iterationsschritt beginnt (zeitliche Dehnung, damit Sie besser beobachten können).

(c) Schreiben Sie ein Programm `leser.c`, das sich zu Beginn das Shared Memory Segment zugänglich macht und anschließend bei Eingabe eines Zeichens von Standardeingabe die Datenstruktur im Shared Memory Segment liest und als lesbare Zeile z.B. in der Form `pid, i: xx...x` ausgibt.

(d) Sie können in einer Datei `run` ein kleines Skript anlegen, in dem Sie mehrere Schreiber-Prozesse im Hintergrund starten.

(e) Beschreiben Sie Ihre Beobachtungen, wenn Sie das Programm `leser` ausführen und wiederholt den Puffer auslesen.

Aufgabe 10.2:

In dieser Aufgabe wird in Erweiterung der Aufgabe 10.1 ein Semaphor zur Sicherung des wechselseitigen Ausschlusses in der Benutzung des Shared Memory Segments als einelementiger Kommunikationspuffer eingesetzt.

Erweitern Sie die Programme aus Aufgabe 10.1 zu Programmen `schreiber_ex.c` und `leser_ex.c`, so dass der wechselseitige Ausschluss in der Benutzung des Puffers zum Ablegen einer (vollständigen) Nachricht bzw. zum Auslesen einer (vollständigen) Nachricht gegeben ist. Ein gegenseitiges Überschreiben des Pufferinhalts durch die Schreiber ist in dieser Lösung nicht ausgeschlossen. Nutzen Sie dazu ein Semaphor und die Operationen P() und V() aus Ihrer Semaphor-Bibliothek. Die Erzeugung und Initialisierung des Semaphors soll in `create.c` erfolgen.

Aufgabe 10.3:

In dieser Aufgabe wird das Problem der fünf speisenden Philosophen (Dining Philosophers Problem, vgl. Kap. 5.3.2) betrachtet. Gehen Sie von der korrekten Lösung des Philosophenproblems aus, wie sie in der Vorlesung (Folien 5-60ff) für N Philosophen vorgestellt wurde. Die Philosophen werden auf UNIX-Prozesse abgebildet. Programmieren Sie die Lösung, indem Sie den gemeinsam benutzten Zustandsvektor `state` aller Philosophen in einem Shared Memory Segment anlegen. Gehen Sie so vor, dass der initiale Prozess alle gemeinsamen IPC-Objekte erzeugt und für jeden Philosophen einen Sohn-Prozess erzeugt, der die Funktion `philosopher` ausführt (eine Überlagerung wird nicht gefordert). Jede Zustandsänderung eines Philosophen (denkend $\rightarrow$ hungrig $\rightarrow$ essend $\rightarrow$ denkend) soll zu einer Ausgabe der Zustände aller Philosophen führen, damit das Gesamtverhalten beobachtet werden kann. Die Zeitdauern eines jeden Philosophen in den jeweiligen Zuständen seien durch Zufallszahlengeneratoren bestimmt. Organisieren Sie die Ausgabe des Programms so, dass alle Ausgaben eines Philosophen untereinander in einer Spalte erscheinen. Testen Sie die Lösung für N=5, 2, 3 und 8 Philosophen.