TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Systemarchitektur, Professur für Betriebssysteme

Betriebssysteme und Sicherheit, WS 2024/25

6. Aufgabenblatt – Deadlocks

Besprechungszeitraum: 03.12.2024 – 06.12.2024

Aufgabe 6.1 Gegeben sei das folgende System paralleler Threads, die jeweils eine Menge von binären Semaphoren `s1` bis `s4` nutzen, die mit 1 initialisiert sind:

Thread A

```
while (1) {
  s1.wait();
  s2.wait();
  work();
  s2.signal();
  s1.signal();
}
```

Thread B

```
while (1) {
  s1.wait();
  s3.wait();
  work();
  s3.signal();
  s1.signal();
}
```

Thread C

```
while (1) {
  s4.wait();
  s1.wait();
  work();
  s1.signal();
  s4.signal();
}
```

Thread D

```
while (1) {
  s3.wait();
  s4.wait();
  work();
  s4.signal();
  s3.signal();
}
```

(a) Wie viele der Threads können sich maximal gleichzeitig in der Funktion `work()` befinden?

(b) Was versteht man unter einem Deadlock? Erläutern Sie mit Bezug auf obiges Beispiel die Bedingungen, die im Zusammenhang mit dem Auftreten von Deadlocks stehen.

(c) Ist das System deadlockfrei? Verwenden Sie zur Untersuchung der Zyklus-Bedingung einen Betriebsmittel-Zuteilungsgraphen.

(d) Welche Konsequenzen hat jeweils das Vertauschen

- der beiden `wait`-Operationen in Thread A,
- der beiden `wait`-Operationen in Thread B,
- der beiden `signal`-Operationen in Thread B?

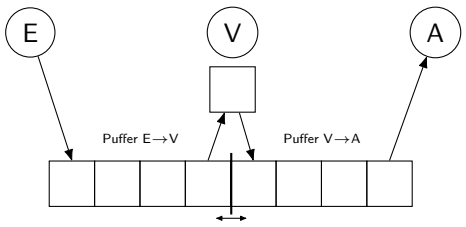
(e) Welche Möglichkeiten gibt es auf Deadlocks zu reagieren?

Aufgabe 6.2 Betrachtet wird im folgenden ein Verfahren zur dynamischen Verhinderung von Deadlocks – der *Bank-Algorithmus*.

(a) Vier Prozesse A,...,D bewerben sich um acht Exemplare eines Betriebsmittels. Der Vektor der maximal benötigten Betriebsmittel sei $(6\ 3\ 8\ 4)^{\mathrm{T}}$, der aktuelle Zustand sei $(1\ 2\ 2\ 1)^{\mathrm{T}}$. Untersuchen Sie, inwieweit – jeweils von diesem Zustand ausgehend – die Forderung nach einem Betriebsmittel-Exemplar durch den Prozess A bzw. durch den Prozess C von dem Bank-Algorithmus bewertet wird. Diskutieren Sie die Gültigkeit der Aussage: „Der Nachfolgezustand eines sicheren Zustands ist sicher“ und ihrer Umkehrung.

(b) Bewerten Sie das Verfahren.

Aufgabe 6.3 Betrachtet werde ein System mit einem Eingabeprozess, einem Verarbeitungsprozess und einem Ausgabeprozess, wobei diese durch zwei Puffer miteinander verbunden sind. Die Prozesse transferieren Daten in Einheiten gleicher Größe. Dabei entnimmt der Verarbeitungsprozess eine Seite aus dem Puffer (gibt diesen Bereich danach frei), speichert den Seiteninhalt in einem lokalen Puffer und füllt dann den Ausgabepuffer. Die Grenze zwischen Eingabepuffer und Ausgabepuffer ist gleitend in Abhängigkeit von der Geschwindigkeit der Prozesse; die einzige Beschränkung ist die Gesamtkapazität des Speichers. Jeder der drei Prozesse arbeitet, solange Eingabedaten für ihn vorhanden sind und Speicherplatz in dem Puffer, den er benötigt, zur Verfügung steht; andernfalls wartet er.



(a) Zeigen Sie, dass es in diesem System zu einem Deadlock kommen kann.

(b) Schlagen Sie eine Lösung zur Vermeidung von Deadlocks vor unter Beibehaltung der gleitenden Grenze zwischen den beiden Puffern.

Aufgabe 6.4 Gegeben sei folgende Funktion zur Übertragung eines Geldbetrages von einem Nutzer einer Bank zu einem anderen Nutzer. Die Funktion achtet dabei auf ein ausreichend gedecktes Konto und bricht die Überweisung ansonsten ab. Um Wettlaufsituationen zu vermeiden, wird *feingranulares Locking* (Lock pro Nutzer) eingesetzt.

```
bool transfer(int amount, user *a, user *b) {
  lock(a);
  if (a->balance >= amount) {
    a->balance -= amount;
    lock(b);
    b->balance += amount;
    unlock(b);
    unlock(a);
    return true;
  }
  unlock(a);
  return false;
}
```

(a) Zeigen Sie mittels eines geeigneten Ablaufs von mehreren parallelen Aufrufen, dass es zu einem Deadlock kommen kann.

(b) Wie kann man in diesem Fall den Deadlock mittels struktureller Verhinderung lösen?

Klausuraufgabe I

Zwei Threads rufen parallel die nebenstehende Funktion auf. Nennen Sie Aufrufe von `f`, die zu einer Verklemmung (*deadlock*) führen können. Jeder Knoten besitzt dabei einen binären Semaphore, der mit `down()` angefordert und `up()` freigegeben wird.

```
1 void f(Knoten *k1, Knoten *k2) {
2    k1->down();
3    k2->down();
4    // Zugriff auf k1 und k2
5    k2->up();
6    k1->up();
7 }
```

Klausuraufgabe II

In einem System stehen insgesamt 7 Bandlaufwerke bereit. Diese werden von verschiedenen laufenden Prozessen (A,B,C,D) für die Langzeitarchivierung von Daten genutzt. Jeder Prozess benötigt dabei ein Bandlaufwerk während seiner gesamten Laufzeit. Je nach zu verarbeitendem Datenaufkommen sind zu verschiedenen Zeitpunkten aber noch zusätzliche Laufwerke nötig. Es spielt dabei keine Rolle welches Laufwerk ein Prozess zugewiesen bekommt; das Schreiben muss allerdings in einem Stück geschehen, da andernfalls Daten verloren gehen. Um dies zu verhindern, wird jedes Bandlaufwerk immer genau einem Prozess exklusiv zur Verwendung zugewiesen bis dieser es ans System zurückgibt.

Die Prozesse benötigen maximal 4 (A), 2 (B), 6 (C) bzw. 5 (D) Bandlaufwerke gleichzeitig. Um eine möglichst hohe Auslastung der verfügbaren Laufwerke zu gewährleisten, soll ein Laufwerk einem Prozess nur dann bereitgestellt werden, wenn und solange er dieses benötigt.

a) Zeigen Sie, dass es im beschriebenen System zu Verklemmungen kommen kann!

b) Um Verklemmungen vorzubeugen wird der Bank-Algorithmus eingesetzt. Aktuell sind den Prozessen 1 (A), 0 (B), 1 (C) und 3 (D) Laufwerke zugeordnet. Nacheinander fordern nun die Prozesse C und D je ein zusätzliches Laufwerk an. Wie reagiert das System auf diese Anforderungen? Wenden Sie zur Beantwortung der Frage den Bank-Algorithmus an!

c) Kann es im Rahmen des Bank-Algorithmus dazu kommen, dass der Vorgänger eines sicheren Zustandes ein unsicherer Zustand ist? Begründen Sie Ihre Antwort!

d) Nennen Sie eine Alternative zur Verwendung des Bank-Algorithmus sowie je einen Vor- und Nachteil dieser Alternative gegenüber dem Bank-Algorithmus!