TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Systemarchitektur, Professur für Betriebssysteme

Betriebssysteme und Sicherheit, WS 2021/22

5. Aufgabenblatt – Synchronisation

Bearbeitungszeitraum: 15.11.2021 – 19.11.2021

Wettlaufsituation (race condition): Gegeben seien mehrere Prozesse, die sich *unabhängig* voneinander um die zeitweise exklusive Nutzung derselben Betriebsmittel bewerben. Seien `A` und `B` Code-Abschnitte innerhalb dieser Prozesse, die bei der Ausführungsreihenfolge `A;B` das System in den Zustand 1 überführen und bei der Reihenfolge `B;A` in den Zustand 2. Eine Wettlaufsituation liegt vor, wenn die parallele Ausführung von `A` und `B` (`A|B`) zu einem anderem Zustand als 1 oder 2 führen kann.

Kritischer Abschnitt (critical section): Abschnitt eines zu einem Prozess gehörenden Programms, innerhalb dessen auf die gemeinsam genutzten Betriebsmittel zugegriffen wird.

Wechselseitiger Ausschluss (mutual exclusion): Koordinierung der Abläufe der beteiligten Prozesse so, dass die kritischen Abschnitte jeweils nur von einem Prozess betreten werden können.

Aufgabe 5.1 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.

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

Zeigen Sie anhand einer geeigneten parallelen Ausführung, wie diese Funktion zu einem negativen Kontostand führen kann, *trotz korrekter Benutzung*. Beschreiben Sie dabei nochmals die Begriffe *Wettlaufsituation* und *kritischer Abschnitt*. Welche anderen Fehler können noch bei einer parallelen Ausführung dieser Funktion entstehen?

Aufgabe 5.2 Zur Lösung von Wettlaufsituationen nutzt man das das Konzept des wechselseitigen Ausschluss zur Absicherung von kritischen Abschnitten. Erklären Sie, warum die folgende Lösung mittels Sperrvariablen nicht sicher ist. Worin liegt die prinzipielle Ursache?

```
void acquire(int *lock) {
  while (*lock != 0) { /* wait */ }
  *lock = 1;
}
```

```
void release(int *lock) {
  *lock = 0;
}
```

Aufgabe 5.3 Neben der Anforderung an die Sicherheit einer Lösung für das Wettlaufproblem wird häufig auch noch Lebendigkeit gefordert. Diese Forderung kann folgendermaßen untergliedert werden: Lebendigkeit ist das *Nicht*-Vorliegen von

- Fernwirkung: Ein Thread außerhalb seines kritischen Abschnitts (und außerhalb der Dienste `acquire`, `release`) behindert den Thread-Ablauf.
- Ausgrenzung: Ein Thread wird ständig am Eintritt in seinen kritischen Abschnitt gehindert.
- Verklemmung: Zwei Threads behindern sich gegenseitig am Eintritt in ihren kritischen Abschnitt.

Diskutieren Sie unter diesen Gesichtspunkten (Lebendigkeit und Sicherheit), inwieweit die folgenden fünf Versuche das Wettlaufproblem zwischen zwei Threads T_1, T_2 lösen. (Die Threads sollen die angegebenen Befehlsfolgen jeweils im Zyklus „ewig“ durchlaufen, sofern möglich.)

- Voraussetzung für 1. und 5. Versuch:

```
int next = 1;
```

- Voraussetzung für 2. – 5. Versuch:

```
bool req1 = false, req2 = false;
```

	Thread 1	Thread 2

1. Versuch	`while (next == 2) {`
	`}`
	`/* kritischer Abschnitt */`
	`next = 2;`
		`while (next == 1) {`
		`}`
		`/* kritischer Abschnitt */`
		`next = 1;`

2. Versuch	`while (req2 == true) {`
	`}`
	`req1 = true;`
	`/* kritischer Abschnitt */`
	`req1 = false;`
		`while (req1 == true) {`
		`}`
		`req2 = true;`
		`/* kritischer Abschnitt */`
		`req2 = false;`

3. Versuch	`req1 = true;`
	`while (req2 == true) {`
	`}`
	`/* kritischer Abschnitt */`
	`req1 = false;`
		`req2 = true;`
		`while (req1 == true) {`
		`}`
		`/* kritischer Abschnitt */`
		`req2 = false;`

4. Versuch	`req1 = true;`
	`while (req2 == true) {`
	`  req1 = false;`
	`  req1 = true;`
	`}`
	`/* kritischer Abschnitt */`
	`req1 = false;`
		`req2 = true;`
		`while (req1 == true) {`
		`  req2 = false;`
		`  req2 = true;`
		`}`
		`/* kritischer Abschnitt */`
		`req2 = false;`

5. Versuch	`req1 = true;`
	`while (req2 == true) {`
	`  if (next == 2) {`
	`  req1 = false;`
	`  while (next == 2) {`
	`  }`
	`  req1 = true;`
	`  }`
	`}`
	`/* kritischer Abschnitt */`
	`next = 2;`
	`req1 = false;`
		`req2 = true;`
		`while (req1 == true) {`
		`  if (next == 1) {`
		`  req2 = false;`
		`  while (next == 1) {`
		`  }`
		`  req2 = true;`
		`  }`
		`}`
		`/* kritischer Abschnitt */`
		`next = 1;`
		`req2 = false;`