GdPI

Henri Dohrendorf

May 9, 2025

1

1. Wenn zwei Prozesse gleichzeitig auf eine Resource zugreifen wollen. Die Abfolge derZugriffe Ändert das Ergebnis. zum Beispiel
   Überweisungen:
   A: +10Euro
   B: -5Euro
   Wenn A und B gleichzeitig den neuen Kontostand berechnen und ddann Überschreiben kann eine der Überweisungen untergehen.

2. Mutex sind Datenstrukturen die den Zugriff auf eine Gemeinsame Resource regeln. Mutex enthalten iene Variable die anzeigt ob die Recource gerade ungenutzt ist und falls nicht eine Queue die die Prozesse auflistet die auf diese Resource warten.

3. Down ist der Versuch die Resounce zu reservieren. Falls sie Frei ist wird sie belegt, falls nicht wird der Prozess angehalten und in die Queue eingereiht. Up ist die Freigabe der Ressource, falls die Queue leer ist wird der Status auf leer gesetzt, falls nicht wird der nächste Prozess aktiviert.

2

Zeile	i	Queue A	Queue B	aktiver Prozess	a	b
4	**1**	–	–	**A**	**false**	**true**
5	**1**	**A**	–	**A**	**false**	**true**
9	**1**	**A**	–	**B**	**false**	**false**
10	**2**	**A**	–	**B**	**false**	**false**
11	**2**	–	–	**B**	**false**	**false**
8	**2**	–	–	**B**	**false**	**false**
9	**2**	–	**B**	**B**	**false**	**false**
6	**3**	–	**B**	**A**	**false**	**false**
7	**3**	–	–	**A**	**false**	**false**
4	**3**	–	–	**A**	**false**	**false**
5	**3**	**A**	–	**A**	**false**	**false**
10	**4**	**A**	–	**B**	**false**	**false**
11	**4**	–	–	**B**	**false**	**false**
8	**4**	–	–	**B**	**false**	**false**
6	**5**	–	–	**B**	**false**	**false**
7	**5**	–	–	**B**	**false**	**true**
4	**5**	–	–	**B**	**false**	**true**

3

1. Ohne Mutex gibt es keinen Grund weshalb die Prozesse sich abwechseln würden. Falls ein Prozess an einer schlechten stellepuasiert wird da die Prozessorzeit ausläuft(bei so einem kurzen Beispiel unwahrsscheinlich) kann next_add zweimal als gleiicher Wert addiert werden und dann an der falshen stelle inkrementiert werden.

2. die Prozesse werden nun abwechselnd aufgerufen aber next_add wird zweimal mit dem gleichen wert benutzt ohne inkrementiert zu werden

3. I:
   global int sum = 0;
   global int next_add = 1;
   global mutex a = true ;
   global mutex b = false ;

A:
while (next_add < 11) {
    down (a);
    sum += next_add ;
    next_add += 1;

    up (b);
}
B:
while (next_add < 11) {
    down (b);
    sum += next_add ;
    next_add += 1;

    up (a);
}
Da Prozesse nicht tatsächlich gleichzeitig laufen brauchen wir zwei verschiedene Variablen um das abwechselnd ausführen zu garantieren.
In diesem Fall wäre es auch das effektiv gleiche Ergebnis ohne b zu benutzen.

4

1. Ohne Mutex könnten beide Spieler denken das sie den ersten Zug haben und der erste Zug von P1 oder P2 von dem anderen überschrieben wird.
   Auch danach könnte es je nach implementation auch weiterhin konflikte geben. Falls Züge überschrieben werden gibt es trotzdem das problem das aktionen aus der Liste zweimal ausgeführt werden können (falls aus der Liste nehmen auch danach entfernen meint).

2. Es lässt sich wie in Aufgabe 1 lösen.
   Aber angenommen die Prozesse werden gleichzeitig gestartet und aktivieren down bevor der andere Spieler seinen Zug schon durch hat und den nächsten startet könnten wir das so machen:
   global mutex a = true; Player:
   while(GameNotFinished){
       down(a);
       action = takeAction();
       actions.append(aktion);
       actionAusführen(action);
       results.append(eval());
       up(a);
   }
   Player 1 und 2 sind hier zwei instanzen von Player