Institut für Informatik

Prof. Dr. Christoph Scholl
Dr. Tim Welschehold
Alexander Konrad
Niklas Wetzel

Freiburg, 23.12.2022

Betriebssysteme
Übungsblatt 10

Wir wünschen Ihnen Frohe Weihnachten und einen erfolgreichen Start ins Neue Jahr!

Aufgabe 1 (3 Punkte)

Erweitern Sie die in der Vorlesung vorgestellte Softwarelösung zum wechselseitigen Ausschluss aus *Versuch 1b (Strikter Wechsel)*, so dass der wechselseitige Ausschluss für n Prozesse garantiert ist. Geben Sie hierzu an, wie der i-te Prozess aussieht ($0 \leq i < n$).

Aufgabe 2 (4+1 Punkte)

In der Vorlesung wurden mehrere Lösungsversuche vorgestellt, mit denen eine Softwarelösung für den wechselseitigen Ausschluss gefunden werden sollte. In dieser Aufgabe geht es um den *Versuch 3*:

Initialisierung

```
1  flag[0] = false;
2  flag[1] = false;
```

Prozess 0

```
3  while(1)
4  {
5    flag[0] = true;
6    while (flag[1] == true)
7      ;        /* tue nichts */
8
9    Anweisung 1
10   Anweisung 2    } kritische Region
11   ...
12
13   flag[0] = false;
14
15   Anweisung 3
16   Anweisung 4    } nichtkritische Region
17   ...
18 }
```

Prozess 1

```
while(1)
{
  flag[1] = true;
  while (flag[0] == true)
    ;        /* tue nichts */

  Anweisung 5
  Anweisung 6    } kritische Region
  ...

  flag[1] = false;

  Anweisung 7
  Anweisung 8    } nichtkritische Region
  ...
}
```

Es ist sichergestellt, dass kein Prozess unendlich lange in seinem kritischen oder nichtkritischen Abschnitt bleibt.

a) Beweisen Sie, dass der wechselseitige Ausschluss auf den kritischen Abschnitt garantiert ist, das heißt, dass zu keinem Zeitpunkt beide Prozesse gleichzeitig im kritischen Abschnitt sein können. Achten Sie darauf, den Beweis vollständig aufzuschreiben.

   Hinweis: Hinweis: Führen Sie ähnlich wie in der Vorlesung bei den Versuchen 1 und 5 einen Widerspruchsbeweis.

b) Erläutern Sie in eigenen Worten, was der grundlegende Nachteil ist, den alle reinen Softwarelösungen zum wechselseitigen Ausschluss aufweisen.

Aufgabe 3 (2+2+2 Punkte)

Betrachten Sie folgenden Versuch für den wechselseitigen Ausschluss:

Initialisierung

```
1  turn    = 0;
2  flag[0] = false;
3  flag[1] = false;
```

Prozess 0

```
4  while(1)
5  {
6      flag[0] = true;
7      while (flag[1] == true)
8      {
9          if (turn == 1)
10         {
11             flag[0] = false;
12             while (turn == 1)
13                 ; /*tue nichts*/
14             flag[0] = true;
15         }
16     }
17
18     Anweisung 1
19     Anweisung 2     } kritische Region
20     ...
21
22     turn = 1;
23     flag[0] = false;
24
25     Anweisung 3
26     Anweisung 4     } nichtkritische Region
27     ...
28 }
```

Prozess 1

```
while(1)
{
    flag[1] = true;
    while (flag[0] == true)
    {
        if (turn == 0)
        {
            flag[1] = false;
            while (turn == 0)
                ; /*tue nichts*/
            flag[1] = true;
        }
    }

    Anweisung 5
    Anweisung 6     } kritische Region
    ...

    turn = 0;
    flag[1] = false;

    Anweisung 7
    Anweisung 8     } nichtkritische Region
    ...
}
```

Es ist sichergestellt, dass `turn`, `flag[0]` und `flag[1]` in den kritischen und nichtkritischen Abschnitten nicht geändert werden und dass jeder Prozess nur endlich lange im kritischen Abschnitt bleibt. Dieser Algorithmus garantiert, dass die beiden Prozesse niemals gleichzeitig in ihren kritischen Abschnitten sind. Das brauchen Sie an dieser Stelle nicht zu zeigen.

Prüfen Sie, ob die übrigen drei Anforderungen für das Problem der kritischen Region erfüllt sind und begründen Sie jeweils Ihre Antwort (unter der Voraussetzung eines fairen Schedulers, der es vermeidet, einem Prozess unendlich lange keine Rechenzeit zuzuteilen):

2. Wenn ein Prozess in den kritischen Abschnitt will, so muss er nur endliche Zeit darauf warten.
4. Wenn kein Prozess im kritischen Abschnitt ist, so wird ein interessierter Prozess ohne Verzögerung akzeptiert.
5. Alles funktioniert unabhängig von der relativen Ausführungsgeschwindigkeit der Prozesse.

Aufgabe 4 (3+3 Punkte)

In der Vorlesung haben Sie den *Peterson-Algorithmus* für den wechselseitigen Ausschluss für zwei Prozesse kennengelernt. Folgend ist eine vermeintliche Erweiterung auf drei Prozesse aufgeführt.

Initialisierung

```
1  f[0] = false;
2  f[1] = false;
3  f[2] = false;
4  turn = 0;
```

Prozess 0

```
5  while(1)
6  {
7    f[0] = true;
8    turn = 0;
9    while((f[1]==true || f[2]==true)
10           && turn==0)
11   {
12       tue nichts;
13   }
14
15   Anweisung 1      } kritische
16   Anweisung 2      } Region
17   ...              }
18
19   f[0] = false;
20
21   Anweisung 3      } nichtkritische
22   Anweisung 4      } Region
23   ...              }
24 }
```

Prozess 1

```
while(1)
{
  f[1] = true;
  turn = 1;
  while((f[0]==true || f[2]==true)
           && turn==1)
  {
      tue nichts;
  }

  Anweisung 5      } kritische
  Anweisung 6      } Region
  ...              }

  f[1] = false;

  Anweisung 7      } nichtkritische
  Anweisung 8      } Region
  ...              }
}
```

Prozess 2

```
while(1)
{
  f[2] = true;
  turn = 2;
  while((f[0]==true || f[1]==true)
           && turn==2)
  {
      tue nichts;
  }

  Anweisung 9      } kritische
  Anweisung 10     } Region
  ...              }

  f[2] = false;

  Anweisung 11     } nichtkritische
  Anweisung 12     } Region
  ...              }
}
```

a) Ist bei der gezeigten Variante für drei Prozesse der wechselseitige Ausschluss gewährleistet? Begründen Sie Ihre Antwort.

b) Ist der wechselseitige Ausschluss garantiert, wenn man in der oben gezeigten Variante in den `while`-Schleifen bei den Vergleichen mit `turn` statt Gleichheit Ungleichheit fordert (für Prozess `i` `turn` $\neq$ `i`, also beispielsweise für Prozess 0: `turn` $\neq$ `0`)? Begründen Sie Ihre Antwort.

Abgabe: als PDF im Übungsportal bis 13.01.2023 um 12:00