Dokumentation

im Studiengang

AIN 4

Philosophenproblem

Mit OpenPEARL

Referent : Patrick Kroner

Vorgelegt am : 04.11.2017

Vorgelegt von : Patrick Kroner
Matrikelnummer: 253417
Rottweilerstraße 87, 78120 Furtwangen
patrick.kroner@hs-furtwangen.de

Inhaltsverzeichnis

1 Einleitung

Als Eistiegsprojekt mit OpenPEARL, haben wir das Philosophenproblem als geeignetes Beispiel ausgewählt. Dadurch wollen wir uns Grundkenntnisse im Programmieren mit OpenPEARL aneignen. Dadurch sollte uns der Einstieg in spätere Projekte leichter fallen. Bei OpenPEARL handelt es sich um eine Programmiersprache die Pearl-Programme platformunabhängig ausführbar machen will. Wird das Programm compiliert erhält man platformunabhängigen C++-Code.

1.1 Philosophenproblem

Hierbei handelt es sich um eine bekannte Multithreading Aufgabe. Programme die auf Multithreading basieren, haben mehrere Prozess die zeitgleich abgearbeitet werden. In der Aufgabenstellung sitzen fünf Philosophen an einem runden Tisch, und jeder hat einen Teller mit Spaghetti vor sich. Zum Essen von Spaghetti benötigt jeder Philosoph zwei Gabeln. Allerdings waren im Haushalt nur fünf Gabeln vorhanden, die nun zwischen den Tellern liegen. Die Philosophen können also nicht gleichzeitig speisen. Jeder darf nur die beiden Gabeln nehmen, die rechts und links von ihm liegen.

1.2 Umsetzung

Um eine gerechte Verteilung der Gabeln zu erreichen, bietet es sich an Semaphoren zu verwenden. Bei Semaphoren handelt es sich um Objecte die den Zustand frei und besetzt haben können. Da jeder Philosoph zwei Gabeln benötigt, gibt es die Möglichkeit zwischen einzelner und doppelter Semaphoren abfrage. Die Semaphorens stellen hier die Gablen dar. Eine einzelne Abfrage von Semaphoren würde die gewünschte Funktion nicht hervorbringen. Im laufendem Programm würde sich dieses aufhängen, da alle Philosophen eine Gabel halten und keine weitere aufnehmen können. Demnach müssen die Semaphoren gleichzeitig abgefragt werden. Dies ermöglicht, dass die Philosophen nur Semaphoren beanspruchen, wenn dieser auf beide sofort zugreifen kann.

2 Umsetzung

2.1 Compilerproblem

Bei dem erstellen der Software stießen wir auf einen Compiler-Fehler. Wir konnten zwar Seamphoren gleichzeitig abrufen, aber wenn es schon eine Kombination aus Semaphoren gab, konnte man keine Ringbildung erzeugen. Es sei denn es gibt nur 2 Semaphoren insgesamt. Ohne Patch des Compilers konnte man das Problem umgehen, indem man den letzten Philosophen in der Reihe die Semaphoren einzeln zugreifen lässt.

Nachdem der Compiler gepatcht wurde, konnte man auch bei dem letzten Philosophen eine doppelte Semaphoren abfrage durchführen.

2.2 Code-Beispiel

Listing 2.1: Philosophenproblem mit OpenPEARL

```
1  MODULE(philo);

3  SYSTEM;
       stdout: StdOut;
5
   PROBLEM;
7          SPC stdout DATION OUT SYSTEM ALPHIC;
           DCL termout DATION OUT ALPHIC DIM(*,80) FORWARD ↩
              STREAM CREATED(stdout);
9          DCL (g1,g2,g3,g4,g5) SEMA PRESET(1,1,1,1,1);
           DCL console SEMA PRESET(1);
11 !Bla
   main: TASK PRIO 10 MAIN;
13         OPEN termout;
           PUT 'STARTET' TO termout BY A, SKIP;
15         CLOSE termout;
           ACTIVATE p1;
17         ACTIVATE p2;
           ACTIVATE p3;
```

```
19          ACTIVATE p4;
            ACTIVATE p5;
21  END;
    p1: TASK PRIO 5;
23          TO 3 REPEAT;
            REQUEST g1,g2;
25          REQUEST console;
            OPEN termout;
27          PUT 'Phil1 - lsst' TO termout BY A, SKIP;
            CLOSE termout;
29          RELEASE console;
            AFTER 1 SEC RESUME;
31          RELEASE g1,g2;
            END;
33  END;
    p2: TASK PRIO 5;
35          TO 3 REPEAT;
            REQUEST g2,g3;
37          REQUEST console;
            OPEN termout;
39          PUT 'Phil2 - lsst' TO termout BY A, SKIP;
            CLOSE termout;
41          RELEASE console;
            AFTER 1 SEC RESUME;
43          RELEASE g2,g3;
            END;
45  END;
    p3: TASK PRIO 5;
47          TO 3 REPEAT;
            REQUEST g3,g4;
49          REQUEST console;
            OPEN termout;
51          PUT 'Phil3 - lsst' TO termout BY A, SKIP;
            CLOSE termout;
53          RELEASE console;
            AFTER 1 SEC RESUME;
55          RELEASE g3,g4;
            END;
57  END;
```

```
  p4: TASK PRIO 5;
59          TO 3 REPEAT;
            REQUEST g4,g5;
61          REQUEST console;
            OPEN termout;
63          PUT 'Phil4 - lsst' TO termout BY A, SKIP;
            CLOSE termout;
65          RELEASE console;
            AFTER 1 SEC RESUME;
67          RELEASE g4,g5;
            END;
69 END;
  p5: TASK PRIO 5;
71          TO 3 REPEAT;
            REQUEST g1,g5;
73          REQUEST console;
            OPEN termout;
75          PUT 'Phil5 - lsst' TO termout BY A, SKIP;
            CLOSE termout;
77          RELEASE console;
            AFTER 1 SEC RESUME;
79          RELEASE g1;
            RELEASE g5;
81          END;
  END;
83 MODEND;
```

Eidesstattliche Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbständig verfasst und hierzu keine anderen als die angegebenen Hilfsmittel verwendet habe. Alle Stellen der Arbeit die wörtlich oder sinngemäß aus fremden Quellen entnommen wurden, sind als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt oder an anderer Stelle veröffentlicht.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben kann.

Furtwangen, den 04.11.2017 Patrick Kroner