

# REFERAT

# Parallele Programmabläufe:

R S.242-246

- 1.Einführung
- 2.Synchronisation
  - 2.1.Monitor-Konzept
  - 2.2.Rendez-vous-Konzept
  - 2.3.Bolt-Synchronisation
- 3.Behandlung Ausnahmesituationen

## 1.Einführung:

Im Bereich der Prozeßtechnik ist es oft erforderlich, daß zur Lösung eines Anwenderproblems parallele Abläufe notwendig sind. Daher stellen Programmiersprachen für diesen Bereich, wie PEARL oder Industrial Real-Time Fortran, spezielle Sprachelemente zur Vereinbarung, zum Aufrufen und zur Ablaufbeeinflussung von Prozessen bereit.

Als Beispiel die Programmiersprache ADA:

Jeder Prozeß wird in einer umfassenden Programmierereinheit vereinbart (z.B.: Unterprogramm, anderer Prozeß). Diese Programmierereinheit heißt Vorgänger (bzw. Vaterprozeß) des Prozesses. Werden im Vereinbarungsteil mehrere Prozesse vereinbart, so werden diese, auch zum Rumpf des Vorgängers, parallel ausgeführt und als Geschwisterprozesse bezeichnet. Wenn ein Vorgänger aktiviert wird, werden alle Prozesse die in seinem Deklarationsteil vereinbart wurden ebenfalls gestartet. Die Abarbeitung der Befehle innerhalb eines Prozesses erfolgt sequentiell. Ein Prozeß ist beendet, wenn seine Anweisungsfolge abgearbeitet ist oder er von außen beendet wurde. Ein Vorgänger ist erst dann beendet, wenn seine eigene Anweisungsfolge und alle im Deklarations vereinbarten Prozesse abgearbeitet wurden.

Dies ist eine Möglichkeit, Prozesse und ihre Verwaltung zu definieren. In manchen anderen Sprachen ist es zusätzlich zum Beispiel auch möglich, einzelne Anweisungen zur parallelen (kollateralen) Abarbeitung freizugeben.

z.B.: ALGOL 68: (x:=7,y:=a,z:=1750)

Die in der Klammerstehenden und durch Komma getrennten Anweisungen können (müssen aber nicht!) parallel ausgeführt werden. Prozesse können genauso wie Prozeduren mit Parametern vereinbart und aufgerufen werden, es gilt auch meist die Blockstruktur für sie.

## 2.Synchronisation:

Bei Programmiersprachen wurden verschiedene Konzepte eingeführt, mit denen der Programmierer seine parallelen Prozesse, die asynchron zueinander ablaufen, auf mehr oder weniger eindeutige, sichere Weise synchronisieren kann. Die Konzepte sind teilweise erst bei wenigen Sprachen eingebunden, und wird auf diesem Teilgebiet der Informatik besonders im Bezug auf verteilte Systeme noch geforscht und weiterentwickelt. Das Semaphore-Konzept ist am meisten verbreitet, da es auch relativ einfach einzubinden ist.

### 2.1.Monitor-Konzept:

Das Konzept wurde von P. Brinch-Hansen und C. Hoare entwickelt. Ein Monitor besteht aus dem von ihm zu verwaltenden Daten und nach außen bekannten Zugriffsprozeduren. Auf seine Daten kann nur innerhalb des Monitors zugegriffen werden, der damit eine Art abstrakter Datentyp realisiert. Wenn ein Monitor oder ein Prozeß von einem anderen Prozeß aufgerufen wird, wird der Monitor quasi ein Bestandteil des Prozesses und kann somit nicht mehr von anderen Prozessen aufgerufen werden. Damit ist das Prinzip des gegenseitigen Ausschlusses bei der Behandlung gemeinsamer Ressourcen gewährleistet. Eine Monitorprozedur kann aus Sicherheitsgründen nicht rekursiv aufgerufen werden. Das folgende Beispiel zeigt die Verwaltung eines Puffers, der von einem "Erzeuger-Prozeß" mit Zeichen gefüllt und von einem Drucker-Treiber geleert wird. Der Monitor garantiert, daß nie gleichzeitig von beiden Prozessen auf den Puffer zugegriffen wird. Das Monitor-Konzept ist u.a. in den Sprachen MODULA und CONCURRENT PASCAL realisiert.

## 2.2 Rendez-vous-Konzept:

Beim Monitor-Konzept wurde der Monitor bzw. aufgerufene Prozeß Teil des aufrufenden Prozesses. Beim Rendez-vous-Konzept kann die aufgerufene Prozedur in einem anderen Prozeß liegen, der parallel zum aufrufendem Prozeß abläuft.

Es gelten folgende Einschränkungen:

- Solald ein Prozeß ein Rendez-vous anmeldet wird dieser unterbrochen bis der aufgerufenen Prozeß zum Rendez-vous bereit ist. Dies kann sofort, später, aber auch nie der Fall sein.
- Ein Prozeß, der von einem anderen Prozeß einen Aufruf erwartet, muß dies als Anweisung enthalten. Er wird dann bis asynchron bzw. parallel zu anderen Prozessen bis zu dieser Anweisung ausgeführt. Liegt ein Aufruf von einem anderen Prozeß vor, wird in der Abarbeitung des Prozesses fortgefahren, andernfalls wird der Prozeß solange ausgesetzt, bis ein Aufruf erfolgt.

Dadurch wird gewährleistet, daß beide Prozesse unabhängig voneinander Arbeiten bis das Rendez-vous an einer definierten Stelle Eintritt und nach Aufruf der Prozedur wieder unabhängig weiterarbeiten.

Dieses Konzept wurde z.B. in ADA realisiert und eignet sich im Gegensatz zum Monitor-Konzept auch für verteilte Systeme.

## 2.2. Bolt-Synchronisation:

Die bisherigen Systeme enthielten einen synchronisierten, aber exklusiven Zugriff auf die angeforderte Prozedur bzw. den Programmabschnitt. Die Bolt-Synchronisation kommt dann zur Anwendung, wenn exklusiver mit nicht exklusivem Zugriff konkurriert.

Die Bolt-Synchronisation wurde in PEARL realisiert. Ein Bolt wird dadurch deklariert, daß angegeben wird, wiviele nicht-exclusive Belegungen möglich sind.

Es gibt dann vier Operationen auf die Bolt-Variable:

1.Nicht-Exklusivanforderung:

Ihr wird stattgegeben, wenn keine Exklusivanforderung höherer Priorität ansteht oder die Maximalbelegung noch nicht erreicht wurde.

2.Freigabe der Nicht-Exklusivanforderung

3.Exklusivanforderung

4.Freigabe der Exklusivanforderung

Beispiel:

Die Belegung einer Startbahn auf einem kleinem Flughafen:  
Die Startbahn kann einerseits zum Rollen benutzt werden (dann kann auch mehrere Flugzeuge gleichzeitig auf der Rollbahn sein, nicht-exklusivbelegung) , aber auch zum Starten (Exklusivbelegung).  
Es können max. 2 Flugzeuge gleichzeitig zur Startbahn Rollen.

Die vier Operationen auf die Bolt-Variable im Beispiel:

1.Nicht-Exklusivanforderung:

Ein Flugzeug fordert die Startbahn zum Rollen an.

2.Freigabe der Nicht-Exklusivanforderung:

Im Beispiel wäre dies das Verlassen der Startbahn nach dem Rollen zum Halteort.

3.Exklusivanforderung:

Im Beispiel ist dies die Belegung zum Starten.

4. Freigabe der Exklusivanforderung:  
Erfolgt im Beispiel nach dem Flugzeugstart.

### 3. Behandlung Ausnahmesituationen:

Unter einer Ausnahmesituation sei hier das durch eine Ausnahmebedingung verursachte Unterbrechen der normalen Programmausführung verstanden. Eine Ausnahmesituation kann sein:

- eine datenabhängige Fehlersituation (Division durch Null bzw. eine zu kleine Zahl)
- bestimmte gerätetechnische oder zeitliche Bedingungen (falsch eingestellte Datenraten oder Synchronisationskonflikte)

Normalerweise reagiert das Laufzeitsystem bzw. Betriebssystem automatisch auf die Ausnahmesituation, indem es die Ausnahmesituation behebt oder das Programm terminiert. Meistens hat der Programmierer aber keinen Einfluß auf die Systemreaktion. Neuere Sprachen enthalten Sprachkonstrukte zur Behandlung von Ausnahmesituationen (Industrial-Real-Time-BASIC, PL/1, ADA, PEARL, usw.). Bei einem Programm zur zyklischen Bearbeitung einer Fertigungsaufgabe darf ein Laufzeitfehler nicht zur Termination des Anwenderprogramms führen. Es sollte vorher versucht werden, einen Aufsetzpunkt für eine normale Programmfortführung zu finden oder das Programm definiert zu terminieren.

Zwei Arten von Ausnahmebedingungen:

1. Vom System her vorgegebene, wohldefinierte Bedingungen, die vom Programmierer mittels fest vereinbarter Kennungen oder Namen angegeben werden.
2. Der Programmierer definiert durch logische Ausdrücke die Ausnahmesituation selbst ( $DURATION > 5$ )

In PEARL kann die erste Art von Ausnahmebedingungen des Systems angesprochen werden (OVERFLOW,ENDOFFILE). Sobald eine Ausnahmebedingung erfüllt ist, wird ein bestimmter Programmabschnitt ausgeführt.

Der nach Eintreten der Ausnahmebedingung aufgerufene Programmblock (Handler) kann aus einer Anweisung, einem Block oder einem speziellen Unterprogramm bestehen. Die Zuordnung eines Handlers zur Ausnahmebedingung kann dynamisch im Programm oder statisch bei der Vereinbarung des Handlers erfolgen. Letzteres ist in PEARL der Fall.

In PEARL kann beliebig zw. Systemreaktionen (Kennung SYS) und dem vom Anwender programmierten Handler gewechselt werden.

## Fragen:

1. Nenne vier Möglichkeiten zur Synchronisation von parallelen Prozessen

- Semaphor-Konzept
- Monitor-Konzept
- Rendez-vous-Konzept
- Bolt-Synchronisation

2. Nenne den Unterschied zw. dem Monitor-Konzept und dem Rendez-vous-Konzept

Monitor-Konzept:

Wird beim Aufruf ein Teil des aufrufenden Prozesses

Rendez-vous-Konzept:

Hier kann die aufgerufene Prozedur in einem anderen Prozeß liegen, der parallel zum aufrufenden Prozeß abläuft.

3. Nenne einige Programmiersprachen für Parallele Programmabläufe

PEARL

Industrial Real-Time FORTRAN

ADA  
ALGOL 68  
MODULA  
CONCURRENT PASCAL  
Industrial-Real-Time BASIC  
PL/1