сообщения
объединенного
института
ядерных
исследований
дубна



Р10-81-342

А.И.Островной, И.М.Саламатин

# ДИСЦИПЛИНА ИСПОЛНЕНИЯ ПАРАЛЛЕЛЬНЫХ ПРОЦЕССОВ В СИСТЕМЕ САНПО

1981

## 1. ВВЕДЕНИЕ

Разработчики систем автоматизации экспериментов и управления технологическими процессами часто встречаются с проблемой ввода, накопления, предварительной обработки и вывода информации, поступающей параллельно по нескольким /синхронным или асинхронным/ каналам. При создании программного обеспечения для таких систем указанная проблема вырождается в проблему программирования параллельных процессов. Этой теме посвящено значительное число работ.

Проблема параллельного программирования имеет много аспектов [1], таких, как автоматическое распараллеливание алгоритмов, формальная теория параллельных вычислений, языки и методы параллельного программирования, организация параллельных вычислений /включая синхронизацию взаимодействующих процессов/.

В данной работе описана организация и дисциплина исполнения параллельных процессов в системе САНПО /система автоматического накопления и предварительной обработки/. Система является частью технологического комплекса программного обеспечения [2], предназначенного для генерации прикладных программных систем автоматизации экспериментов /САЭ/ на реакторе ИБР-2. Генерация САЭ средствами САНПО осуществляется на основе описания прикладной системы на специализированном языке высокого уровня [3], который включает средства программирования параллельных процессов. Комплекс САНПО реализован для ЭВМ типа СМ-3 [4].

## 2. ПОНЯТИЕ ПАРАЛЛЕЛЬНЫХ ПРОЦЕССОВ В СИСТЕМЕ САНПО

Параллельными процессами мы будем называть фрагменты программы, которые могут исполняться одновременно во время ее работы. Если программа исполняется только одним процессором, то при любом способе построения ее фрагменты исполняются последовательно. Однако при наличии соответствующих средств /набора процессоров/ такие программы допускают параллельное исполнение [1], т.е. одновременное получение результатов в различных фрагментах программ.

В САНПО этими фрагментами являются процессы, точнее, элементарные процессы. Совокупность процессов в САНПО и представляет собой параллельно исполняемую программу. Мы будем говорить о параллельно исполняемых процессах или просто о параллельных процессах /ПП/.

Организация ПП в САНПО включает в качестве одного из основных элементов событие. События - это поименованные объекты языка и системы САНПО, выступающие в двух качествах. С одной стороны, событие - это логическая переменная, принимающая значения TRUE и FALSE, с другой - оно является некоторым признаком /или сигналом/,отражающим состояние элементов данных /ЭД/ системы [2,3,5]. Событие как сигнал может находиться в активном состоянии /что соответствует значению TRUE / и в пассивном /значение FALSE /. Операцию приведения данного события в активное состояние мы называем операцией установки флага события,или объявления события, а обратную операцию - операцией сброса флага события.

Назначение событий - отражение состояния ЭД в системе. Активное состояние события соответствует тому, что информация из ассоциированного или ассоциированных с ним ЭД готова к обработке. Помимо этого, события могут отражать любую желаемую ситуацию, сложившуюся в процессе работы прикладной системы /ПС/.

Для любого события можно задать некоторое действие /один или несколько операторов/, которое будет инициировано после приведения данного события в активное состояние. Такая последовательность операторов вместе с соответствующим событием составляет элементарный процесс /ЭП/. ЭП может находиться в пассивном или активном состоянии в зависимости от состояния ассоциированного с ним события.

В результате работы одного ЭП могут быть приведены в состояние готовности другие ЭП /объявлены соответствующие события/. Два ЭП называются функционально связанными, если в результате исполнения одного из них всегда или при определенных условиях другой ЭП приводится в активное состояние.

Процесс - это совокупность функционально связанных ЭП с определенными для них приоритетными отношениями, которые устанавливаются специальными инструкциями и регламентируют соответствующую дисциплину исполнения ЭП.

## 3. ДИСЦИПЛИНА ИСПОЛНЕНИЯ СОВОКУПНОСТИ ПАРАЛЛЕЛЬНЫХ ПРОЦЕССОВ

Элементарный процесс является по существу единицей программирования ПП в САНПО. Поэтому порядок инициирования ЭП в целом и составляет дисциплину исполнения ПП как совокупности связанных ЭП.

Порядок инициирования ЭП зависит от их приоритетов. Введены приоритетные отношения ЭП двух типов: абсолютные и относительные.



Абсолютные приоритеты определяются в конечном счете порядковыми номерами соответствующих описаний ЭП в таблице оперативной базы данных /ОБД/ [2,5]. Способ формирования этой таблицы описан в работе [3].

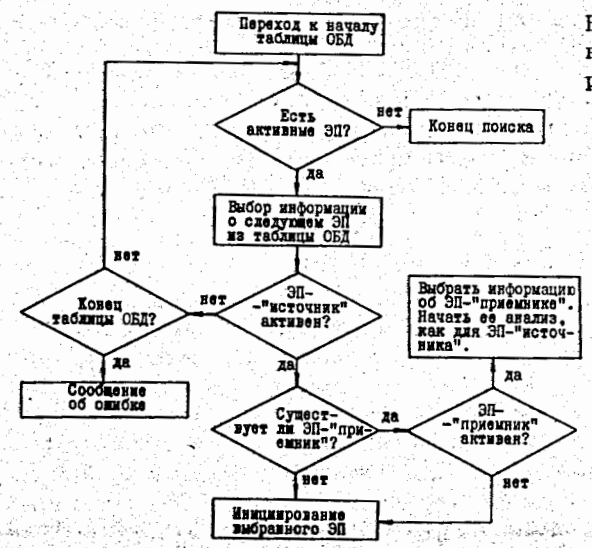
Относительные приоритеты определяются иерархическими отношениями, которые носят характер подчинения. Эти отношения устанавливаются инструкцией языка САНПО [3], имеющей вид:

/HIERARCHY_ ⟨имя события⟩ → ⟨имя события⟩

В этой инструкции стрелка обозначает факт установления связи между событием, имя которого указано слева от стрелки /будем его называть событием - "источником"/, и событием, имя которого стоит справа от стрелки /его будем называть событием-"приемником"/. Относительный приоритет трактуется следующим способом: если оба события в иерархической паре активны, то событие - "источник" имеет меньший приоритет по сравнению с событием - "приемником", в остальных случаях приоритеты этих событий находятся в согласии с их абсолютными приоритетами.

Заметим, что иерархические отношения определяются для пар событий и не обладают свойством транзитивности, т.е. если определены отношения E1→E2 и E2→E3, то это не означает, что имеет место отношение E1→E3. Если описанное отношение обозначать стрелкой, то от одного события не может исходить более одной стрелки, но приходить может произвольное их количество.

ЭП инициируются монитором САНПО в соответствии с их состоянием /инициируются только активные ЭП/ и приоритетными отношениями. Монитор выполняет поиск активного ЭП с высшим приоритетом. Для этого он сканирует список ЭП в порядке убывания их абсолютных приоритетов и выбирает первый встретившийся активный ЭП. Далее проверяется возможность инициировать найденный ЭП. Монитор проверяет наличие другого, иерархически связанного с выбранным, ЭП, т.е. ЭП, ассоциированного с событием - "приемником". Если такого ЭП нет или он находится в пассивном состоянии, то инициируется выбранный ЭП. Если же такой ЭП существует и находится в активном состоянии, то монитор учитывает, что этот новый ЭП обладает более высоким приоритетом по отношению к ранее найденному. Дальнейшая работа монитора сводится к аналогичным проверкам, но уже по отношению к новому ЭП, заменяющему своего предшественника по иерархической паре. На рисунке приведена блок-схема алгоритма выбора ЭП для инициирования. В подписях на этом рисунке приняты следующие обозначения: ЭП, ассоциированный с событием-"источником", назван ЭП-"источником", а ЭП, ассоциированный с событием-"приемником" - ЭП-"приемником".




Блок-схема алгоритма выбора ЭП для инициирования.

Описанный порядок инициирования ЭП может в процессе работы прикладной системы корректироваться или изменяться специальными операторами, включенными в отдельные ЭП, в зависимости от выполнения или невыполнения каких-либо логических условий.

## 4. ИСПОЛНЕНИЕ ЭЛЕМЕНТАРНЫХ ПРОЦЕССОВ

В системе САНПО различаются два вида ЭП: медленные и быстрые. Различия между двумя видами ЭП состоят в том, что быстрые ЭП обеспечиваются более быстрым механизмом исполнения, а медленные ЭП исполняются в режиме интерпретации.

В общем виде ЭП можно представить в виде события и ассоциированного с ним набора операторов. Инициирование ЭП ведет к тому, что на исполнение посылается первый оператор, а в процессе обработки события /работы данного ЭП/ последовательно выбираются и посылаются на исполнение остальные операторы из соответствующего набора. Этот набор операторов обрабатывает выделенную ему монитором часть информации из ЭД.

Окончание работы ЭП /обработки соответствующего события/ происходит в двух случаях:

1. Один из операторов сбросил флаг события, ассоциированного с данным ЭП /процедура сброса флага события описана в работе [6] /.

2. Закончилось исполнение последнего оператора из набора, включенного в ЭП. В этом случае монитор анализирует состояние ЭД-источника и решает вопрос о сбросе флага или о продолжении обработки события.

Во втором случае, если ЭП остается в активном состоянии, то последовательность операторов из данного ЭП будет инициирована вновь в соответствии с приоритетом данного ЭП после сканирования всего списка ЭП. Таким способом организуются неявные циклы [5,6]. Чаще всего неявные циклы используются в случаях, когда набор операторов, включённых в ЭП, обрабатывает только часть готовой к обработке информации. Процедура исполнения таких неявных циклов состоит в обработке информации последовательными порциями упомянутым набором операторов.

Приведем различия, имеющие место в дисциплинах исполнения медленных и быстрых ЭП:

1. Последовательность операторов быстрого ЭП может включать только функциональные операторы [3], а в медленном ЭП могут присутствовать операторы цикла, условные операторы и т.п.

2. Последовательность операторов быстрого ЭП считается единой операцией. Исполнение этой последовательности не может быть прервано объявлением событий, включенных в более высокоприоритетные ЭП. Медленные ЭП могут быть прерваны после окончания текущего исполняемого оператора.

3. В процессе работы быстрого ЭП стандартной программой, входящей в его состав, не может быть объявлено такое событие, ЭП которого имеет иерархическую связь и более высокий относительный приоритет, чем данный ЭП. В медленных ЭП такие действия допустимы. В этом случае исполнение текущего ЭП будет прервано и будет инициирован наиболее высокоприоритетный ЭП в соответствии с описанным в п.4 алгоритмом.

4. Стандартные программы /СП/, включенные в набор быстрого ЭП, должны иметь один формальный буфер - "источник" [6], имя которого совпадает с именем события, ассоциированного с данным ЭП. Это требование справедливо для СП, которые должны иметь в списке аргументов обращения буфер-"источник" и буфер-"приемник" [6]. Заметим, что для быстрых ЭП монитор с помощью вырабатываемого им системного счетчика [6] обеспечивает передачу одного и того же участка буфера-"источника" для обработки по очереди всем СП, включенным в данный ЭП. Текущий указатель состояния буфера [6] остается неизменным во время исполнения ЭП и передвигается монитором только после окончания исполнения ЭП. Медленные ЭП не имеют таких ограничений на использование в них СП. Текущие указатели состояния буферов здесь изменяются монитором после исполнения каждой отдельной СП в соответствии с обработанным данной СП количеством информации из буферов "источника" и "приемника".

## 5. ОБСУЖДЕНИЕ

Описанная дисциплина и организация параллельных процессов в системе САНПО реализована для мини-ЭВМ СМ-3[4], имеющей один процессор. Очевидно, что в конечном счете все ЭП исполняются последовательно, а представление алгоритмов в виде ПП призвано обеспечить увеличение пропускной способности системы накопления в целом, повышение наглядности, проблемную ориентацию средств программирования и адекватность этих средств решаемой проблеме - проблеме автоматизации эксперимента. Помимо этого, используя ПП для описания системы автоматизации эксперимента, мы получаем хорошую структурную организацию этой системы, что, в свою очередь, способствует облегчению ее реализации, развития и сопровождения, уменьшает вероятность ошибок при разработке и служит увеличению ее надежности.

В случае наличия многопроцессорной вычислительной установки принятая в САНПО идеология описания и организации ПП может быть использована для реализации действительно параллельного исполнения ЭП.

Заметим, что описанная дисциплина исполнения ПП обеспечивается монитором САНПО, для этой цели не применяются внешние прерывания ЭВМ. Помимо описанных ПП в САНПО, как правило, присутствуют процессы, которые также исполняются в "параллельной манере", но инициируются по внешним запросам прерывания ЭВМ. В САНПО такие программы названы "программами обработки прерываний" и включены в подсистемы[2]. Организация исполнения этих программ обеспечивается аппаратными средствами мини-ЭВМ СМ-3[4].

В настоящее время в литературе уделяется большое внимание развитию концепций структурной организации, модульности в параллельном программировании [7,8], развитию средств синхронизации [9-11], программирования ПП без использования семафоров [10,11], т.е. повышению уровня средств параллельного программирования. Предложенные средства описания[3] и реализации ПП в системе САНПО позволяют:

1. Программировать ПП без использования семафоров, причем единицей программирования ПП выступает ЭП.

2. Использовать эффективные структуры управления в виде иерархической взаимосвязи ЭП.

3. Обеспечить технологичность разработки систем автоматизации экспериментов путем расчленения системы реального времени на несколько независимых процессов, которые в конечном счете сводятся к совокупности элементарных действий в виде ЭП, тем самым с необходимостью обеспечивается модульная организация разрабатываемой системы, разработка ее по методу сверху вниз, преемственность отладки прикладных систем автоматизации за счет применения уже отлаженных модулей, ЭП или фрагментов ПП.



Следует иметь в виду, что разработанные средства параллельного программирования ориентированы прежде всего на реализацию алгоритмов, работающих в реальном масштабе времени и в системах автоматизации экспериментов или управления технологическими процессами, но вместе с тем описанный метод реализации ПП можно рассматривать как материал для обобщения в универсальных или ориентированных на другую проблемную область языках программирования.

## ЛИТЕРАТУРА


1. Липаев В.В. Распределение ресурсов в вычислительных системах. "Статистика", М., 1979, с.247.
2. Балука Г. и др. ОИЯИ, Р10-12960, Дубна, 1980.
3. Островной А.И., Саламатин И.М. ОИЯИ, Р10-80-423, Дубна, 1980.
4. Наумов Б.Н., Боярченков М.А., Кабалевский А.Н. "Приборы и системы управления", 1977, №10, с.3-5.
5. Островной А.И., Саламатин И.М. ОИЯИ, Р10-11349, Дубна, 1978.
6. Островной А.И., Саламатин И.М. ОИЯИ, Р10-80-490, Дубна, 1980.
7. Котов В.Е. Параллельное программирование с типами управления. "Кибернетика", Киев, 1979, №3, с.1-13.
8. Brinch Hansen P. Structured Multiprogramming Commun., ACM, 1972, vol.15, No.7, pp.574-578.
9. Дийкстра Э. Взаимодействие последовательных процессов. В кн.: Языки программирования. Ред. Ф.Женюи, пер. с англ. под ред. В.М.Курочкина. "Мир", М., 1972, с.406.
10. Morgan C. Parallel Programming without Synchronization. Technical Report No. 136, Basser Department of Computer Science, University of Sydney, 1978.
11. Ludewig J. PCSL-Process Control Software Specification Language. KfK 2874, Kernforschungszentrum, Karlsruhe, April, 1980.


Рукопись поступила в издательский отдел
21 мая 1981 года.