EESTI NSV TEADUSTE AKADEEMIA TOIMETISED. 24. KÖIDE
FÜÜSIKA * MATEMAATIKA. 1975, NR. 1

ИЗВЕСТИЯ АКАДЕМИИ НАУК ЭСТОНСКОЙ ССР. ТОМ 24
ФИЗИКА * МАТЕМАТИКА. 1975, № 1

https://doi.org/10.3176/phys.math.1975.1.04

УДК 681.3.06 : 51

Р. РАУД, В. КУУСИК

УПРАВЛЕНИЕ РЕСУРСАМИ В ОПЕРАЦИОННОЙ СИСТЕМЕ ОС-32

Операционная система ОС-32 разработана для ЭВМ «Минск-32», к которой, кроме стандартных периферийных устройств, подключаются емкие запоминающие устройства с произвольной выборкой (магнитные диски или барабаны) и устройство дисплея с выносными пультами; при этом учтена возможность подключения и других внешних устройств.

ОС-32 работает в режиме мультипрограммной пакетной обработки с произвольным числом заданий и дистанционной обработки данных одновременно.

Разработка механизма управления ресурсами проводилась в соответствии с требованиями, вытекающими из общей идеологии ОС-32, и с учетом условий работы группы разработчиков.

1. Механизм управления ресурсами должен четко выделяться из всех остальных частей управляющей программы (УП) и ОС [1], а также иметь модульную структуру для обеспечения генерируемости и гибкости УП.

2. Основные модули, автоматически подключаемые к УП (при генерировании), не должны содержать алгоритмов для повышения эффективности работы системы в экстремальных условиях, но возможность подключения к УП таких алгоритмов в виде отдельных модулей должна быть учтена.

3. Основные модули должны быть достаточно общими для обеспечения управления всеми ресурсами, что реализуется специальными модулями для каждого типа ресурсов.

4. Распределение ресурсов осуществляется по приоритетам.

В настоящей работе рассматриваются основные проблемы, возникающие при управлении ресурсами в мультипрограммной ОС, анализируются некоторые существующие методы решения этих проблем и приводится описание общего механизма управления ресурсами в ОС-32.

Проблемы

В мультипрограммной вычислительной системе (ВС) возможно параллельное выполнение (продвижение) нескольких процессов. При пакетной обработке самый общий из них — это процесс продвижения заданий через ВС, при этом каждое задание пользователя (см. [1]) является его подпроцессом. Процессы могут создавать подчиненные процессы (передавая им все или несколько своих ресурсов).

Для управления ресурсами важно только перемещение процесса из одной стадии в другую в смысле требования и освобождения ресурсов. Резервированием (claim) ресурса назовем осведомление процессом системы о будущих намерениях использовать ресурс. Требованием (request) ресурса назовем просьбу процесса закрепить за ним ресурс.

Резервирование передает распределителю ресурсов информацию о будущих требованиях процесса и не дает никаких преимущественных прав. В некоторых ОС резервирование может отсутствовать. Подобная информация используется главным образом для планирования ресурсов и предотвращения тупиковых ситуаций. Закрепление ресурса дает процессу право на его использование.

Освобождением (release) ресурса процесс информирует систему о ненужности этого ресурса вообще или на некоторое время. Ресурс считается перехватываемым (preemptive), если он может быть захвачен другим процессом (например, более приоритетным) без каких-либо действий со стороны владеющего ресурсом, однако после освобождения ресурса возможно восстановление его состояния (в данной системе) в момент захвата и закрепление за прежним владельцем.

Примером перехватываемого ресурса может служить процессор, при захвате которого требуется обычно только запоминание небольшого количества рабочих регистров устройства управления и вычислителя.

Примером перехватываемого ресурса является, например, том магнитной ленты, восстановление состояния которого обычно чрезмерно дорого.

Степень возможности перехватываемости зависит от конкретной ОС — точнее, от объема организующих программ.

Ресурс считается разделяемым (sharable), если два или большее число процессов могут независимо использовать его. Например, пакет магнитных дисков может быть разделяемым между процессами, которые имеют файлы в этом пакете дисков.

Если неразделяемый ресурс требуется больше, чем одному процессу, он должен быть закреплен за одним процессом в каждый отдельный момент времени. В некоторых случаях ресурс может меняться из разделяемого в неразделяемый и из перехватываемого в неперехватываемый.

Тупиковой называется ситуация (deadlock situation), когда ресурсы распределяются между процессами таким образом, что продвижение двух или большего числа процессов невозможно без перехвата одного или более неперехватываемых ресурсов. Например, когда два параллельных процесса имеют каждый половину доступных устройств управления магнитными лентами (УМЛ), а для завершения обоим требуется две третьих УМЛ.

Установлено [2], что необходимым условием возникновения тупиковой ситуации является закрытая цепочка процессов, за каждым из которых закреплен неразделяемый и неперехватываемый ресурс и каждый из которых требует такого же ресурса, закрепленного за следующим процессом в цепочке. Возникновение тупиковой ситуации невозможно при перехватываемых ресурсах, а также при ресурсе, который через определенное время после закрепления освобождается независимо от распределения других ресурсов (например, канал ввода-вывода, который после завершения обмена освобождается системой).

При появлении тупиковой ситуации нужно определить процессы и ресурсы, которых это касается, и перераспределить некоторые неперехватываемые ресурсы. Однако это приводит к большим затратам времени, а также требует наличия специальных программных средств в ОС.

Поэтому большое внимание уделяется различным методам предотвращения тупиковой ситуации. Так, Дж. В. Хавендер [3] считает необходимым соблюдение одного из следующих условий.

1. Все требования процесса на ресурсы должны следовать в обусловленном порядке.

2. Процесс должен зарезервировать все параллельно используемые ресурсы раньше требования на какое-либо из них.

3. Требуя дополнительные ресурсы, процесс должен быть в состоянии освободить все ранее им занятые.

4. Если требование отвергается, процесс должен быть в состоянии использовать альтернативный ресурс.

Однако в [3] отмечалось, что каждое из этих требований, пригодное для некоторых ресурсов, может оказаться неприемлемым для других.

Первое условие положительно в том смысле, что не требует трудоемкого контроля за действительным распределением ресурсов, но в то же время налагает на пользователя строгие ограничения и может привести к неэффективному использованию дефицитных ресурсов.

Второе условие требует от системы запоминания и использования информации о будущем. Алгоритм А. Н. Хабермана [4] (дополненный Р. С. Холтом [5]) требует указания максимального количества единиц из всех используемых процессом ресурсов. Состояние системы считается безопасным в смысле возникновения тупиковой ситуации, если всегда есть хотя бы один процесс, который может завершиться с использованием только имеющихся и свободных ресурсов. Распределитель ресурсов проверяет все требования на ресурсы и удовлетворяет только те из них, после которых состояние системы остается безопасным. В случае перегрузки системы, когда большинство ресурсов закреплены, — распределитель ресурсов отвергает большинство требований ввиду малого количества безопасных состояний. Это может привести к блокировке многих процессов, владеющих дефицитными ресурсами.

При третьем условии процесс разбивается на шаги, между которыми все ресурсы должны быть в перехватываемом состоянии. Этот метод неэффективен при кратковременном использовании ресурса и длинном шаге процесса, так как все ресурсы должны быть заказаны перед шагом.

Четвертое условие, очевидно, неприменимо для большинства ресурсов.

Дж. Х. Ховард предлагает следующую классификацию допускаемых стратегий [2].

1. Обнаружение ситуации тупика. Это периодически выполняемый алгоритм, который по существующему распределению ресурсов и отвергнутым требованиям определяет наличие тупика, а также процессы и ресурсы, которых это касается. После обнаружения тупика система должна его ликвидировать путем перераспределения причастных к тупику ресурсов. Как отмечает Дж. Х. Ховард, этот подход приводит к неоправданным издержкам не только за счет алгоритма обнаружения, но и в связи с трудоемкостью организации перехвата неперехватываемых ресурсов, при частом появлении тупиковой ситуации. Повторяемость же последней сильно зависит от конфигурации конкретной вычислительной установки (см., напр., [6]).

2. Избегание (avoiding) тупика. Это означает выполнение второго условия Дж. В. Хавендера, например, посредством алгоритма А. Н. Хабермана, который обсуждался выше.

3. Предотвращение (prevention) тупика. Сюда относится соблюдение первого, третьего или четвертого из условий Дж. В. Хавендера. Предотвращение, в отличие от избегания, требует меньшего контроля за требованиями на ресурсы во время работы. В стратегии предотвращения наиболее совершенным является метод упорядочения ресурсов. Неперехватываемые ресурсы группируются в классы требований $C_1, \ldots, C_k$. Если процесс владеет ресурсом класса C_i, он может требовать ресурсов класса C_j только при условии $i > j$. Соблюдение этого правила делает возникновение закрытых цепочек невозможным.

Недостатки такого подхода были обсуждены выше. Положительными сторонами предотвращения можно считать отсутствие издержек во время распределения ресурсов, а также отсутствие так наз. балансирующего эффекта на метод планирования (алгоритмы избегания тупика в значительной степени снижают роль, например, приоритетного планирования [5]). Порядок ресурсов следует устанавливать в соответствии с наличием ресурсов в конкретной вычислительной установке.

Дж. Х. Ховард предлагает смешанную стратегию — разбивать ресурсы на классы требований и к каждому из них применять любой метод (например, избегание), касающийся ресурсов только этого класса. Таким образом, метод может быть выбран в соответствии с характером ресурсов и с учетом возможностей ОС. Такой подход реализован в ОС на CDC-6600 в университете штата Техас.

При параллельном выполнении процессы должны синхронизировать свои действия, т. е. обмениваться сообщениями. Для этой цели предложено несколько механизмов.

Один из них [7] основан на так наз. переменной события (EVENT VARIABLE), которая представляет собой булевскую переменную (E) и список процессов, ждущих происшествия события. E может принимать значения «случилось» и «не случилось». Для обработки переменной события предусмотрены следующие операции.

WAIT (E) — проверяет значение E. Если оно в состоянии «не случилось», то процесс блокируется и включается в список ждущих за событием. В противном случае процесс может продвигаться дальше.

CAUSE (E) — дает переменной E значение «случилось», и процессы в очереди за событием разблокируются.

RESET (E) — дает переменной E значение «не случилось».

Как отметил Е. В. Дийкстра [8], в большинстве случаев описанным механизмом трудно пользоваться в связи с некоммутативностью операций WAIT и CAUSE. Он предлагает для синхронизации [9, 8] так наз. семафоры, которые обрабатываются P- и V-операциями.

P-операция уменьшает значение семафора (s) на один; если $s \geqslant 0$, то процесс, выполняющий P-операцию, может продолжаться, если же $s < 0$, то процесс блокируется и включается в список ожидающих этого семафора.

V-операция увеличивает значение s на один. При $s \leqslant 0$ один из ожидающих процессов разблокируется.

Естественно рассматривать сообщения процессов как ресурсы, а отправителя — как владельца ресурса. Сообщение может быть разделяемым или неразделяемым в зависимости от характера сообщения и намерения получателя. Например, сообщение о доступности некоторого файла с общими данными — разделяемое для процессов, которые только читают его, но неразделяемое для процесса, собирающегося модифицировать содержание файла.

Семафоры представляются эффективными для неразделяемых ресурсов. С разделяемыми ресурсами возникают трудности, особенно, если

иметь дело с неопределенным количеством параллельных процессов и различным использованием (разделяемый, неразделяемый) семафора. В ОС-32 реализован механизм синхронизации, объединяющий возможности обоих описанных выше механизмов.

Управление ресурсами в ОС-32

Механизм управления ресурсами составляет ядро ОС. Он должен иметь четкий интерфейс между системой управления файлами (СУВВ в [1]), системой программирования и другими частями ОС, т. е. должен быть отдельно выделяемым и иметь стандартизированные связи с другими частями ОС.

Реализация некоторых языков программирования (напр., ПЛ/1) требует возможности разветвления программ. К ветви могут обращаться другие ветви программы; их требования должны выполняться поочередно. По изложенным соображениям казалось естественным управлять ветвью как ресурсом. Любая программа в ОС-32 рассматривается как ветвь. В смысле управления ресурсами каждая ветвь представляет собой ресурс. Ветви называются активными ресурсами, так как они могут (в принципе) требовать любого другого ресурса и создавать новые ресурсы (ветви, файлы).

Каждый файл по существу — тоже ресурс. Файлов в системе может быть много, и каждый из них используется сравнительно редко. Поэтому нецелесообразно фиксировать количество файлов и иметь их описания (как ресурсов) в системе постоянно, как в [6]. По приведенным выше соображениям в ОС-32 имеется возможность создания новых ресурсов по возникновению надобности в них.

В смысле управления ресурсами ВС представляет собой динамически меняющийся конгломерат ресурсов. Ресурс может быть свободным или занятым действиями какого-либо процесса.

Менеджером называется комплекс программных модулей, реализующий общий механизм распределения ресурсов и синхронизации.

Для менеджера ресурс определяется тремя понятиями (V, Q, P).

V — вектор, каждая часть которого содержит информацию о состоянии ресурса (занято, свободно), о процессе, за которым ресурс закреплен, и дополнительную информацию для процедур управления ресурсом (P). Количество частей V соответствует количеству равноценных частей физического ресурса (например, восемь устройств УМЛ). Предполагается, что процессу безразлично, какая именно часть ресурса закрепляется за ним.

P состоит из трех процедур управления ресурсом: закрепления, освобождения и перехвата (есть только у перехватываемых ресурсов), задачи которых очевидны по названиям.

Q — это упорядоченная очередь процессов, ждущих за ресурсом. В очереди каждое требование процесса представлено элементом ожидания, содержащим адрес контрольного блока процесса, приоритет процесса, адрес блока описания ресурса и адрес поля дополнительной информации, а также ссылку на следующий элемент ожидания в очереди за данным ресурсом. Элементы ожидания находятся в таблице резервирования ресурсов (см. ERT [6]). Каждый элемент упакован в две ячейки памяти. Содержание элемента ожидания укомплектовано таким образом, чтобы часто происходящие операции (требование, освобождение) сделать наиболее быстрыми, а сравнительно редкие операции (напр., освобождение при аварийном окончании процесса) только возможными.

Менеджер формирует элемент ожидания при поступлении требования процесса из числа пустых элементов в таблице (пустые элементы также связаны ссылками) и уничтожает (т. е. связывает в цепочку к пустым элементам) после освобождения ресурса по этому требованию.

В функции менеджера при поступлении требования от процесса входит формирование элемента ожидания, включение последнего в очередь за требуемым ресурсом и активизация процедур управления этим ресурсом, если имеется соответствующее количество свободных частей ресурса или если при перехватываемом ресурсе удалось перехватить их в нужном количестве.

При освобождении менеджер активизирует процедуру освобождения и затем, если очередь Q не пуста и наиболее приоритетное требование может быть удовлетворено, — процедуру закрепления.

Проблема предотвращения тупика решается предложенной Дж. Х. Ховардом [2] смешанной стратегией как наиболее гибкой.

Контроль за порядком требований на ресурсы осуществляется менеджером, а алгоритмы избегания тупика (напр., алгоритм А. Н. Хабермана [4]) внутри классов ресурсов реализуются отдельными модулями, и обращения к ним содержатся уже в процедурах управления этими ресурсами. О необходимости и содержании этих модулей следует судить по конкретной обстановке в ВС. Как пишет Д. Дж. Фрейли в [6], тупиковая ситуация в ОС MACE на CDC-6600 в течение года была обнаружена только дважды, хотя механизм управления ресурсами не содержал алгоритмов избегания (предотвращения) тупика.

Механизм управления ресурсами ОС-32 дает возможность выбирать наиболее подходящий (в смысле скорости и объема) алгоритм, учитывая характер использования конкретных ресурсов в данной ВС.

Синхронизация процессов в ОС-32 осуществляется механизмом семафоров. Различие между описанным механизмом управления ресурсами и механизмом семафоров обусловлено тем, что ресурс — сообщение простое и не требует специальных процедур управления и дополнительной информации. Благодаря этому сокращается как объем элемента ожидания, так и информация для создания ресурса сообщения, а также упрощается обработка соответствующих требований, поступающих от процессов. С целью предоставления возможности использования семафора как в разделяемом, так и в неразделяемом виде, введены, в отличие от [9], следующие изменения:

1) значение семафора указывает количество владельцев, а не количество ожидающих за семафором;

2) к семафору добавлен указатель вида использования (f). При выполнении P-операции процесс указывает требуемый вид использования. Указатель f семафора получает значение владельца(-ев) семафора. У семафора в неразделяемом виде может быть только один владелец, в разделяемом — несколько. Процесс блокируется и включается в очередь за семафором согласно приоритету процесса, если он выполнял P-операцию а) в неразделяемом виде и $s > 0$, или б) в разделяемом виде и семафор использовался в неразделяемом виде.

P-операция увеличивает значение s на один, если процесс владел семафором. V-операция уменьшает значение s на один и устанавливает f на разделяемый вид. Затем активизируются все процессы в очереди за семафором с проверкой тех же условий, что и при P-операции, до первого, при котором условия не соблюдаются. Хотя описанный механизм сложнее, чем в [9], он дает возможность использовать семафоры во всех случаях синхронизации и гарантирует четкий интерфейс между механизмом управления ресурсами и другими частями ОС.

Заключение

ВС с ОС-32 можно в смысле управления ресурсами разбить на два уровня — механизм управления ресурсами и ресурсы. Ресурсы делятся на пассивные и активные. Активные ресурсы (ветви) могут в принципе требовать любого другого ресурса в системе и создавать (описывать) новые. Количество ресурсов в системе не фиксируется. Всякий ресурс может быть свободным или занятым действиями какого-либо процесса.

Синхронизация процессов осуществляется с помощью ресурсов сообщений — семафоров.

Механизм управления ресурсами содержит две схемы. Первая схема предназначена для управления семафорами. В зависимости от требования процесса она обеспечивает как разделяемое, так и неразделяемое использование семафора, однако управляет всеми семафорами одинаково без учета характера какого-либо конкретного из них.

Вторая схема предназначена для управления ресурсами, требующими специальных действий при управлении (каждый ресурс имеет процедуры управления, которые выполняют специальные для этого ресурса действия). В зависимости от характера ресурса вторая схема допускает доступ к нему в разделяемой или неразделяемой форме. Ресурс может также управляться как перехватываемый или неперехватываемый (семафор всегда неперехватываемый).

Ресурсы распределяются по приоритетам процессов, но применение алгоритмов избегания тупиковой ситуации имеет сбалансированный эффект, которого нельзя избежать из-за возможности возникновения «искусственного» тупика [5]. Вторая схема не налагает ограничений на продолжение процесса после отказа от закрепления требуемого ресурса. О закреплении процесс может узнать по семафору.

Предотвращение тупиковой ситуации осуществляется комбинированным методом [2]. Определяется порядок классов требования ресурсов, а внутри класса применяется алгоритм избегания тупика.

Механизм управления ресурсами не предотвращает постоянной блокировки процессов с низким приоритетом. Это должно быть сделано на более высоком уровне иерархии (напр., при запускании процессов строго по приоритету).

Механизм управления ресурсами имеет четкую и гибкую модульную структуру, что дает возможность применять наиболее подходящие алгоритмы управления ресурсами и предотвращения (избегания) тупиковых ситуаций для каждой конкретной конфигурации и задач ВС.

ЛИТЕРАТУРА

1. Косе Г. Я., Куусик В. А., Куусик Л. Е., Рауд Р. К., Препринт № 7 Ин-та кибернетики АН ЭССР, Таллин, 1973.
2. Howard J. H., Comm. ACM, **16**, No. 7, 427—430 (1973).
3. Havender J. W., IBM Systems J., **7**, No. 2, 74—84 (1968).
4. Haberman A. N., Comm. ACM, **12**, No. 7, 373—377 (1969).
5. Holt R. C., Comm. ACM, **14**, No. 1, 36—38 (1971).
6. Fraily D. J., Comm. ACM, **16**, No. 5, 323—329 (1973).
7. Dahm D. M., Gerbstadt F. H., Pacelli M. M., Comm. ACM, **10**, No. 12, 772—779 (1967).
8. Dijkstra E. W., Operating System Techniques. A.P.I.S. Studies in Data Processing, No. 9.
9. Dijkstra E. W., Comm. ACM, **11**, No. 5, 341—346 (1968).

Институт кибернетики
Академии наук Эстонской ССР

Поступила в редакцию
10/IV 1974

R. RAUD, V. KUUSIK

OPERATSIOONISÜSTEEMI OS-32 RESSURSSIDE JUHTIMINE

Artiklis analüüsitakse multiprogrammiliste operatsioonisüsteemide ressursside juhtimisel tekkivaid mõningaid probleeme ja kirjeldatakse ENSV TA Küberneetika Instituudis väljatöötatava operatsioonisüsteemi OS-32 ressursside juhtimise ja protsesside sünkroniseerimise meetodit.

R. RAUD, V. KUUSIK

RESOURCE MANAGEMENT IN OS-32

The paper contains an analysis of some resource management problems in a multiprogramming operating system and a description of the resource management method implemented in the operating system OS-32 for MINSK-32 computer.