МОДЕЛЬ УПРАВЛЕНИЯ ПАМЯТЬЮ ДЛЯ МИКРОЯДЕР ВТОРОГО ПОКОЛЕНИЯ M-M / S-CD

Е. И. Клименков

Белорусский государственный университет информатики и радиоэлектроники
Минск, Беларусь
E-mail: Evgeny.Klimenkov@gmail.com

Представлен новый подход к управлению памятью в контексте микроядер второго поколения, получивший название M-M/S-CD, который был разработан в духе принципа минимализма. Данный подход был разработан с нуля на основе аналитической модели памяти компьютерной системы. Модель управления памятью M-M/S-CD выносит все механизмы и политики, связанные с управлением памятью, из пространства ядра в пространство приложений пользовательского режима. Единственной задачей ядра операционной системы является обеспечение соблюдения двух требований, которые гарантируют, что система памяти будет оставаться в замкнутом состоянии.

Ключевые слова: микроядро второго поколения, управление памятью.

Введение

Архитектура операционных систем на базе микроядер второго поколения в настоящее время по-прежнему наиболее перспективная. Появившаяся в рамках проекта Mach [1] архитектура на основе микроядра приобрела большую популярность в научной среде в связи с рядом существенных преимуществ, которые она предлагала. Второе поколение микроядер появилось благодаря работам Йохана Лидтке в рамках проекта L4, который добавил принцип минимализма в основу архитектуры микроядра. Принцип минимализма гласит: «Концепция переносится внутрь микроядра, тогда и только тогда, когда ее реализация за пределами ядра, т. е. любая конкурирующая реализация, будет препятствовать реализации необходимых функциональных возможностей системы» [1].

Управление памятью (УП) является одной из основных и неизбежных функций операционной системы (независимо от ее архитектуры). Управление памятью не может быть полностью вынесено за пределы ядра в пространство пользователя, поскольку память является одним из наиболее важных системных ресурсов и ее управление требует доступа к привилегированным средствам процессора.

Существует два распространенных подхода к управлению памятью в микроядрах второго поколения: рекурсивное построение адресного пространства [2, 3, 5] и управление памятью на основе мандатов [4], мы считаем, что они по-прежнему являются усложненными и избыточными с точки зрения принципа минимализма.

Аналитическая модель

Память компьютерной системы может быть представлена двумя сущностями: физическим пространством (PS) и виртуальным пространством (VS), где первая

представляет набор всех страниц физической памяти системы, а вторая представляет собой набор всех страниц виртуальной памяти системы. Эти два набора, в свою очередь, могут быть разделены на четыре набора: страницы физической памяти, отображенные на внешние ресурсы (PMP); страницы физической памяти, не отображенные на внешние ресурсы (PHP); страницы виртуальной памяти, отображенные на внешние ресурсы (VMP) и страницы виртуальной памяти, не отображенные на внешние ресурсы (VHP). В соответствии с этим управление памятью можно рассматривать как установление соответствия между физическим пространством и виртуальным пространством:

$$\begin{aligned} PS &\mapsto VS \\ (PMP, PHP) &\mapsto (VMP, VHP), \end{aligned} \tag{1}$$

где всегда должны удовлетворяться два условия:

$$\forall P_i \in PMP, \exists (P_i \mapsto P_j), P_j \in VMP, \tag{2}$$

$$\forall P_i \in PHP, \nexists (P_i \mapsto P_j), P_j \in VMP, \tag{3}$$

где P – страница памяти (виртуальной или физической, в зависимости от того, к какому набору страниц она принадлежит.

Первое условие (2) называется предотвращением утечки памяти. В контексте языков программирования высокого уровня утечкой памяти называют удержание блока памяти после окончания его фактического использования, что может повлечь за собой исчерпание доступной памяти компьютерной системы или доступного адресного пространства. В отличие от языков высокого уровня, утечка памяти в контексте задачи управления памятью со стороны операционной системы относится к исключение блока памяти из пула памяти используемой системой. В нормальных условиях ожидается, что вся доступная память системы должна быть использована для выполнения полезной работы, и потеря последней ссылки на блок памяти предотвращает использование этого блока памяти для дальнейшего выполнении полезной работы. В связи с этим подсистема управления памятью ядра операционной системы должна предусматривать меры, направленные на удержание как минимум одной ссылки (отображения на страницу виртуальной памяти) для каждой страницы физической памяти, связанной с внешними ресурсами.

Второе условие (3) называется предотвращением нарушения доступа. Как было сказано выше, как ФП, так и ВП может включать страницы, которые не связаны с внешними ресурсами. Однако процессор ведет себя по-разному при попытках осуществления доступа к страницам памяти, не связанным с какими-либо внешними ресурсами, в зависимости от того, в каком пространстве производятся эти попытки доступа. Доступ к такой странице в пространстве ФП практически полностью невидим как для прикладных программ, так и для операционной системы. Данные, записываемые в страницу физической памяти, не связанную с внешним ресурсом, будут игнорироваться, а чтение данных с такой страницы будет возвращать читателю случайные данные, захваченные с шины данных. Однако в обоих случаях логика процессора не будет ни генерировать сигнала об ошибке, ни предпринимать никаких любых других операций в ответ на нарушение доступа. В противоположность этому процессор может обнаружить нарушение доступа при обращении к страницам виртуальной памяти, не отображенным на страницы физической памяти и в ответ вызвать исключение. Такое исключение вызывает операционную систему для корректной обработки на-

рушения доступа к памяти. В результате ВП может быть использовано для защиты программного обеспечения от доступа к страницам ФП, не связанным с внешними ресурсами. Предотвращение нарушение доступа предполагает обеспечение соблюдение правил, в соответствии с которыми ни одна из страниц ФП, не связанная с внешними ресурсами, никогда не должна быть отображена ни на одну из страниц ВП.

Удовлетворения двух описанных выше условий достаточно, чтобы перевести память компьютерной системы в состояние закрытой согласованной системы. В связи с этим обеспечение удовлетворения управления памятью двум этим условиям является единственной деятельностью, связанной с управлением памятью, которая должна быть реализована на стороне ядра операционной системы. Остальная деятельность и политики, связанные с управлением памятью, могут быть безопасно реализованы на стороне менеджеров памяти пользовательского режима, которые могут свободно манипулировать отображениями между страницами ФП и ВП до тех пор, пока описанные выше условия будут удовлетворяться.

Управление памятью M-M / S-CD

Существуют два требования, предъявляемые к управлению памятью системы, описанные в аналитической модели выше:

- предотвращение утечек памяти;
- предотвращения нарушений доступа.

Соблюдение требования предотвращения нарушений доступа может быть обеспечено за счет предоставления гарантий от архитектуры ядра, согласно которым ФП, созданный сторонним провайдером, будет сохраняться в неизменном состоянии. Это означает, что ни одной страницы не будет добавлено или удалено в/из ФП на протяжении всего срока службы системы. Предполагается, что сторонний провайдер (например, загрузчик системы или загрузчик узла системы) обязан передать ядру множество начальных виртуальных адресных пространств, состоящих исключительно из страниц, отображенных на страницы ФП, связанные с внешними ресурсами, такими как RAM, ROM, регистры устройств ввода-вывода, отображенные на память. Соответственно в данном множестве не должно быть ни одной страницы, отображенной на страницу ФП не связанную с внешним ресурсом.

В отличие от классической проблемы утечки памяти, рассматриваемой в контексте языков программирования высокого уровня, утечки памяти в контексте управления памятью ядра означают потерю последнего отображения на страницу ФП, и, следовательно, потерю способности включать отображение на эту страницу ФП в любое виртуальное адресное пространство для будущего фактического использования приложениями. Удовлетворение требования предотвращения утечки памяти требует явного применения ограничений на операции управления памятью на стороне ядра. Причиной для этого является то, что отображения между страницами ФП и страницами ВП могут устанавливаться и разрушаться динамически, так как виртуальные адресные пространства могут создаваться и уничтожаться ядром в течение времени и страницы ВП, не связанные со страницами ФП, могут преобразовываться в страницы ВП, связанные со страницами ФП и наоборот. В связи с этим ядро должно применять к управлению памятью политику, в соответствии с которой будет невозможно потерять последнее отображение на страницу ФП.

Оба описанных выше требования могут быть удовлетворены путем применения памяти модели управления памятью, которую мы назвали M-M\S-CD. Эта модель предполагает разделение всех страниц ВП, отображенных на страницы ФП, на два класса: ведущие страницы и ведомые страницы. Каждая страница ФП связана с внешним ресурсом и также жестко связана с ровно одной ведущей страницей ВП. Таким образом, ведущая страница представляет соответствующую ей страницу ФП в системе управления памятью. Управление памятью M-M\S-CD предполагает, что ведущая страница не может быть ни создана, ни разрушена непосредственно ядром. Набор ведущих страниц доставляется в ядро сторонним провайдером, таким как загрузчик, в процессе инициализации ядра и остается неизменным в течение всего срока службы системы. Единственной операцией, которая может быть применена к ведущей странице, является операция перемещения между различными слотами страниц в одном и том же виртуальном адресном пространстве или между различными виртуальными адресными пространствами, представленными в системе. В отличие от ведущих страниц, имеющих статический характер, ведомые страницы имеют динамический характер и могут создаваться из ведущей страницы и уничтожаться неограниченное количество раз на протяжении всего срока службы системы. У каждой ведущей страницы может быть множество ведомых страниц одновременно. В результате с точки зрения модели памяти M-M\S-CD память компьютерной системы рассматривается как замкнутая система по отношению к ведущим страницам, которые представляют страницы ФП, связанные с внешними ресурсами (см. рис. 1).

Описанная замкнутая модель управления памятью компьютерной системы M-M\S-CD включает только три концептуальные операции над страницами памяти:

- Перемещение ведущей страницы (*svas, spfn, tvas, tpfn*) – атомарная операция, которая удаляет ведущую страницу из слота страницы *spfn* из виртуального адресного пространства *svas* и вставляет ее в пустой слот страницы *tpfn* в виртуальном адресном пространстве *tvas*.
- Создание ведомой страницы (*svas, spfn, tvas, tpfn*) – создает копию ведущей страницы из слота *spfn* в виртуальном адресном пространстве *svas* и вставляет ее как ведомую страницу в пустой слот *tpfn* в виртуальном адресном пространстве *tvas*.
- Уничтожение ведомой страницы (*svas, spfn*) – удаляет ведомую страницу из слота *spfn* в виртуальном адресном пространстве *svas*.

В целях различения страниц различных типов в нашей реализации прототипа на платформе Intel x86 использованы два из трех доступных для использования битов дескриптора страницы. Один из этих битов используется для обозначения ведущей страницы, второй – для страницы реализации блокировки страницы, необходимой для реализации атомарности операций.

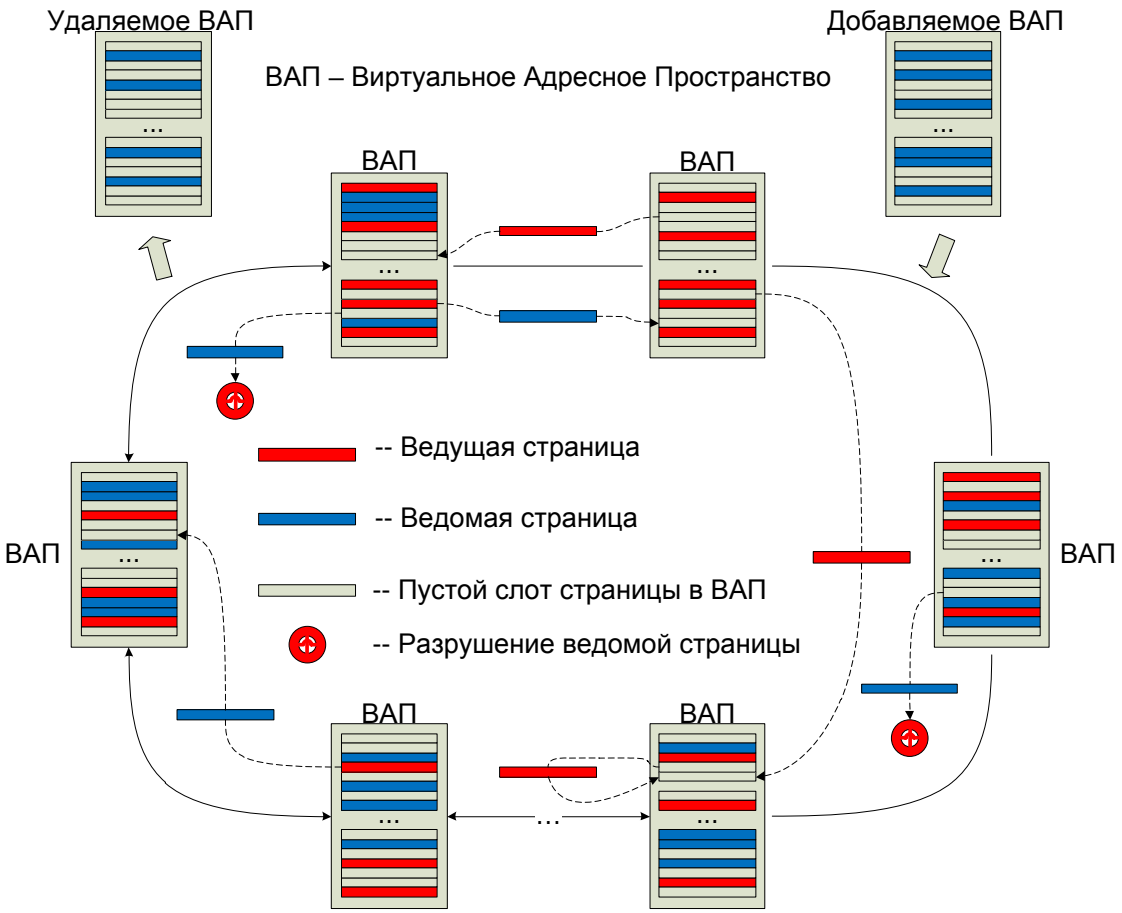

Рис. 1. Замкнутая модель управления памятью M-M/S-CD

Заключение

Модель управления памятью M-M/S-CD позволяет обеспечить целостность управления памятью, вынося при этом все механизмы и политики за пределы ядра.

Библиографические ссылки

1. *Accetta M.* Mach: A New Kernel Foundation For UNIX Development // Technical Conference – USENIX. 1986. P. 93–112.
2. *Alan Au.* L4 User Manual. Version 1.14. / Alan Au, Gernot Heiser // School of Computer Science and Engineering, University of NSW, Sydney 2052, Australia, March. 1999.
3. *Elphinstone K., Heiser G., Liedtke J.* L4 Reference Manual MIPS R4x00 Version 1.11 // School of Computer Science and Engineering, University of NSW, Sydney 2052, Australia, May. 1999.
4. *Gerber S.* Virtual Memory in a Multikernel. Master Thesis (Nr.47), Systems Group, Department of Computer Science, ETH Zurich, Switzerland, 2012.
5. *Kuz I.* L4 User Manual. NICTA L4-embedded API. Version 1.11 // University of NSW, Sydney 2052. Australia, October 2005.
6. *Liedtke J.* On m-Kernel Construction // 15th ACM symposium on Operating Systems Principles. P. 237–250.