УДК 681.3.066

И.Б. ТУРКИН, Т.С. НИКИТИНА

Национальный аэрокосмический университет им. Н.Е. Жуковского "ХАИ", Украина

КОНЦЕПЦИИ РЕАЛИЗАЦИИ ФУНКЦИОНАЛЬНОЙ БЕЗОПАСНОСТИ И НАДЕЖНОСТИ В ОПЕРАЦИОННЫХ СИСТЕМАХ РЕАЛЬНОГО ВРЕМЕНИ

Анализируются методы обеспечения надежности и безопасности операционных систем реального времени (ОСРВ). Рассматриваются стандарты безопасности для ОСРВ и существующие разработки в данном направлении, обсуждается концепция реализации выделенных разделов, архитектура ОСРВ. Предлагается реализация ключевых механизмов работы с виртуальной памятью, взаимодействие процессов и потоков в рамках стандарта POSIX.

авиация, операционная система реального времени, безопасность, надежность, выделенные разделы, процессы, потоки, стандарты

Введение

Актуальность проблемы. На данный момент многие из государственно важных объектов находятся под управлением ненадежных операционных систем MS Windows и MS DOS. Некоторые объекты используют существующие операционные системы на основе операционной системы (ОС) Unix, однако и это не дает необходимой безопасности, а также эти системы слишком громоздки и требовательны к ресурсам, чтобы использовать их на объектах военно-промышленного комплекса – например в авиации. И за рубежом, и на отечественных предприятиях предпринимаются попытки создать собственную ОС. Уже создано несколько различных систем, которые применяются в критически важных объектах, требующих максимальной надежности и безопасности.

Так, например, система Tornado/vxWorks используется на марсоходах Spirit и Opportunity, а также на некоторых авиалайнерах, система LynxOS [1, 2] используется в авиации, на флоте, встраивается в современные танковые машины. Широчайшее применение находит ОС QNX [3]. Все они – ОСРВ с высокой степенью надежности и безопасности. Две основные причины вынуждают в качестве альтернативы предлагать собственные разработки:

– в системах критического применения закрытость исходного кода создает потенциальную опасность;

– специфическая элементная база либо решаемая задача не позволяет эффективно использовать существующие программные продукты [4].

Основные требования и стандарты для ОСРВ в области авиации

Для разработки надежной и безопасной ОСРВ необходимо соблюдать целый ряд требований: система должна работать в реальном времени, быть безопасной, надежной, функциональной; потреблять минимум аппаратных ресурсов; должна быть способной к расширяемости (модульность), исходный код должен быть открытым; должна соответствовать требованиям современных стандартов безопасности и функциональности. Для обеспечения работы в реальном времени ОСРВ должна давать отклик на любые непредсказуемые внешние воздействия в течение предсказуемого интервала времени.

Разрабатываемая ОСРВ должна принадлежать к классу жестких систем реального времени, т.е.:

– никакое опоздание не приемлемо, ни при каких обстоятельствах;

– результат, выданный с опозданием – бесполезен;

– нарушение крайнего срока времени отклика рассматривается как катастрофический отказ;

– цена превышения времени отклика бесконечно высока.

Требования открытости системы предполагают ее соответствие основным правилам и стандартам для UNIX-подобных систем, таких как POSIX. Механизмы реализации всех системных требований безопасности, функциональности и реального времени должны соответствовать международным стандартам в отраслях, где данная ОСРВ будет использоваться.

К примеру, в настоящее время определены следующие основополагающие стандарты для ОСРВ в области авиации:

– ARINC 653 (Avionics Application Software Standard Interface) разработан компанией ARINC в 1997 г. Этот стандарт определяет универсальный программный интерфейс APEX (Application/Executive) между ОС авиационного компьютера и прикладным ПО;

– CCITSE (Common Criteria for Information Technology Security Evaluation) – это набор требований и условий секретности, одобренных Агентством национальной безопасности и Национальным институтом стандартов и технологий США, а также соответствующими органами других стран. CCITSE определяет уровни гарантии секретности – EAL (Evaluation Assurance Level). Common Criteria оценивает не только безопасность и надежность продуктов, но и процессы их разработки и поддержки [5];

– MILS (Multiple Independent Levels of Security/Safety) делает возможной математическую верификацию программного ядра системы путем уменьшения функциональности за счет предъявления к системам четырех обязательных групп требований;

– POSIX (Portable Operating System interface for unIX) определяет переносимый интерфейс операционных систем на уровне исходных текстов. Основная спецификация разработана как спецификация IEEE 1003.1 и одобрена в качестве международного стандарта ISO/IEC 9945-1:1990 [6, 7].

Концепция выделенных разделов

На сегодняшний момент среди всех существующих ОСРВ, LynxOS [5] наиболее приближена к максимальному количеству стандартов безопасности и надежности для устройств и технологий с критическими требованиями – такими как авиалайнеры, спутники, военные корабли, и другие. Поэтому для реализации ОСРВ предлагается взять за основу механизмы и концепции этой ОС. В результате многочисленных исследований в качестве основной была принята концепция изолированных разделов. Каждый раздел полностью изолирован и это разделение относится к процессорному времени, адресному пространству и ресурсам в каждом разделе. Данная концепция гарантирует изоляцию от сбоев и доступность всех ресурсов. В разрабатываемой ОСРВ фиксированные разделы должны обслуживаться как виртуальные машины. Каждый прикладной процесс оперирует внутри его собственной среды операционной системы, как если бы он работал на своем собственном процессоре. Это применимо ко всем ресурсам процессора и именованным пространствам.

Архитектура ОСРВ, обеспечивающая поддержку выделенных разделов

Управление разделами контролируется с помощью таблицы конфигурации виртуальных машин (Virtual-Machine Configuration Table – VCT). Каждый раздел в момент инициализации получает определенный ограниченный объем памяти, который, тем не менее, может быть увеличен или уменьшен после завершения инициализации. Каждый раздел может обслуживать несколько процессов и потоков. Время, выделенное для каждого раздела, является фиксированным, например, в системе с 3 разделами и максимальным выделением времени на один цикл 100мс, циклический обработчик может выделить время первому разделу – 20мс, второму – 30 мс, и третьему – 50мс. Это позволяет запускать разные по

уровню безопасности и по системным требованиям программы в разных разделах с обеспечением бесперерывной и точной работы. Для изолирования процессов в памяти используется Модуль Управления Памятью (Memory Management Unit – MMU). Данный модуль также транслирует виртуальный адрес процесса в памяти в физический. Циклический обработчик в данной системе не только управляет переключением разделов, но и обработкой процессов и потоков внутри них. Следует обратить внимание на то, что скорость переключения между потоками в рамках одного процесса больше, чем при переключении потоков в разных процессах. Базовая архитектура ОСРВ представлена на рис. 1.

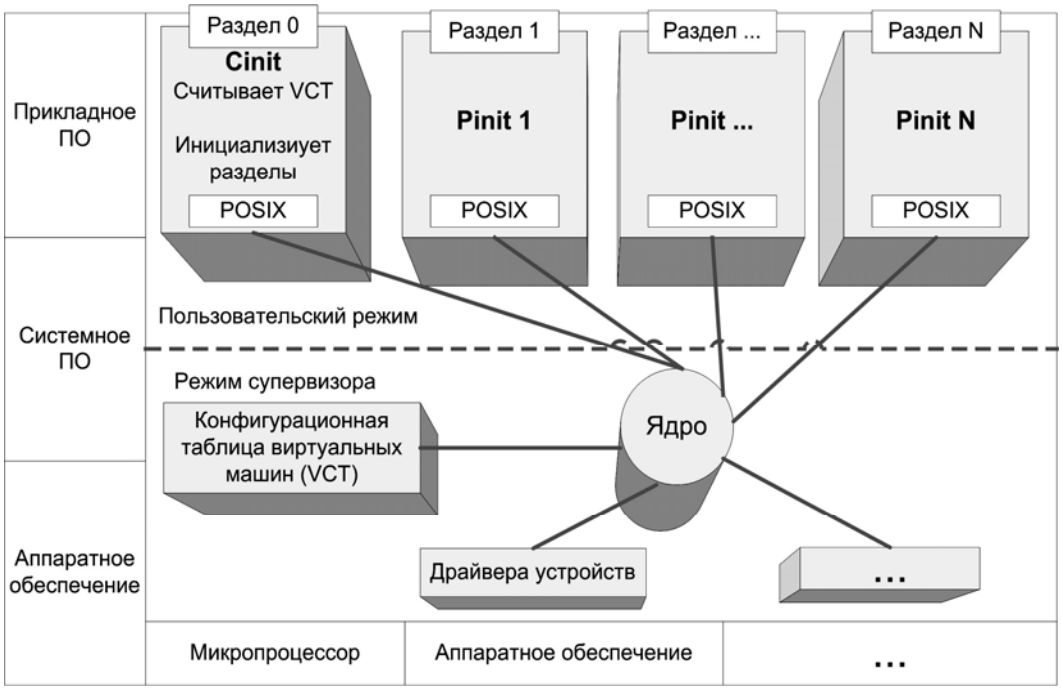

Рис. 1. Архитектура ОСРВ

Основная инициализация (Cinit) – это первый POSIX процесс запущенный после инициализации ядра. Этот процесс получает привилегии root. Он читает конфигурационную таблицу виртуальных машин (VCT) и создает и подключает разделы (виртуальные машины – Раздел1, Раздел2, Раздел N). Инициализация разделов (pinit) – это дочерние процессы cinit, которые создаются в момент инициализации каждого отдельно взятого раздела. В качестве языка описания используются языки программирования Си и Ada-95 и формирует основные требования ARINC 653 к выделенным разделам:

– каждому разделу выделяется непрерывное линейное физическое адресное пространство, и границы этого пространства не могут меняться в процессе работы системы, также ОСРВ должна обеспечивать изолирование информации в адресном пространстве внутри выделенного каждому разделу линейного куска;

– фундаментальной концепцией ARINC 653 является то, что процессы в одном разделе либо не должны влиять на поведение процессов в другом разделе, либо могут влиять на них заранее определенным и контролируемым способом. ARINC 653 требует, чтобы процессорное время выделялось каждому разделу строго циклически (round-robin), причем время владения процессором при этом должно задаваться заранее в конфигурационной таблице;

– приложения в пользовательском режиме не

могут разрушить область кода в режиме супервизора;

– прерывания не должны посягать на время и память в другом разделе.

Взаимодействие процессов в POSIX

На рис. 2. представлено отображение процессов в виртуальной памяти на независимые страницы физической памяти. Стандарт POSIX предполагает, что каждый процесс в системе находится в собственном пространстве памяти. Менеджер памяти (MMU) используется для физического изолирования процессов друг от друга, чтобы они не могли посягать на память других процессов. Каждый процесс изолирован через виртуальную память, и поэтому невозможно, чтобы другой процесс использовал адресное пространство другого процесса. Набор адресов, к которым процесс имеет доступ, называется виртуальным адресным пространством процессов.

Если процесс пытается обратиться к странице, на которой он не отображается, то менеджер памяти генерирует исключение.

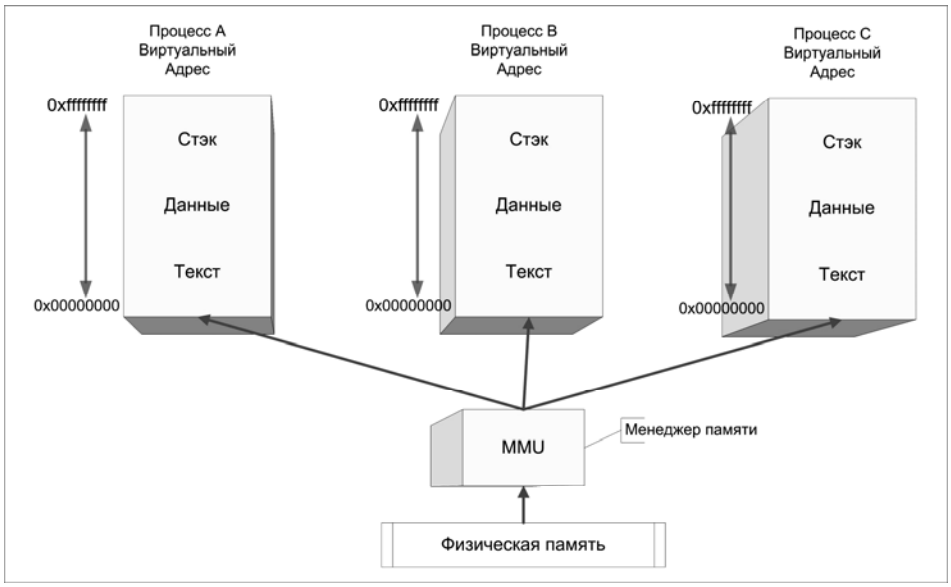

Рис. 2. Отображение процессов в виртуальной памяти на независимые страницы физической памяти

Взаимодействия потоков и процессов в POSIX

На рис. 3. представлена схема взаимодействия потоков и процессов в POSIX для СРВ. Взаимодействие потоков основано на следующих принципах:

– потоки контролируются и выполняются в контексте процесса;

– потоки в одном и том же процессе имеют общий доступ ко всему адресному пространству этого процесса;

– потоки не имеют родительско-дочерних отношений;

– если любой поток вызывает функцию выхода, то процесс прекращает свою работу. Все остальные потоки также прекращают выполнение;

– каждый процесс имеет свое защищенное адресное пространство и обмен данными между процессами возможен только через функции ядра.

Стандарт ARINC как интерфейс для выделенных разделов

Стандарт ARINC определяет интерфейс для выделенных разделов, который предполагает реализацию следующих системных сервисов для выделенных разделов:

– сервисы по управлению разделами;

– сервисы по управлению процессами;

– сервисы ответственные за взаимодействие процессов в одном разделе;

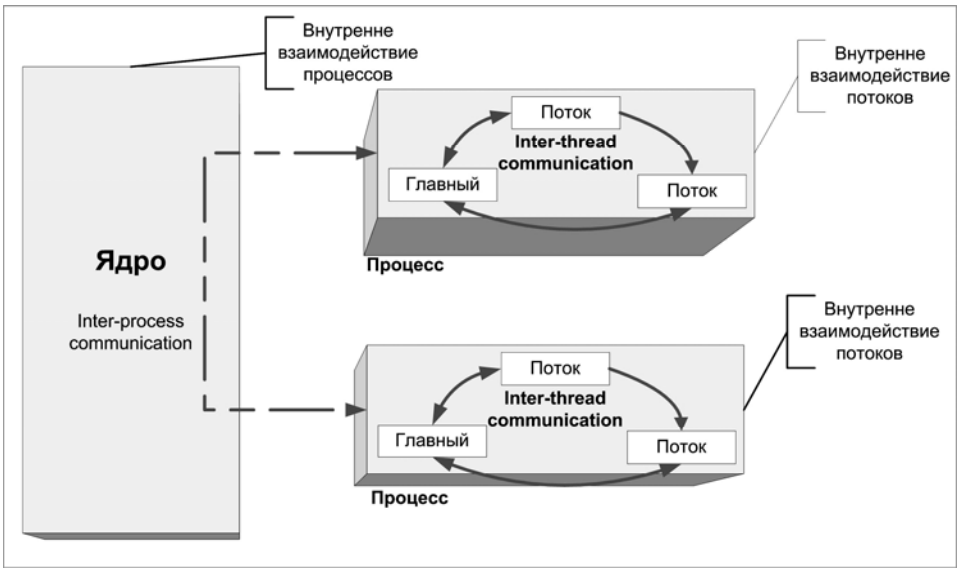

Рис. 3. Взаимодействие потоков и процессов в стандарте POSIX

– сервисы ответственные за взаимодействие процессов в разных разделах;

– обмен сообщениями путем установления канала связи, который должен быть заранее описан в конфигурационной таблице системы;

– обмен через буфер, при котором только один раздел может писать в конкретный буфер, а все другие – только читать. Это обеспечивает возможность широковещательного обмена информацией между разделами.

Заключение

Обобщая результаты теории и практики разработки ОСРВ, можно отметить, что наибольший интерес представляет идеология построения высоконадежных программных средств на основе положений стандарта ARINC, в основе которого лежит механизм выделенных разделов. Для дальнейшего исследования и реализации механизмов выделенных разделов, предлагается использовать открытую, документированную ОС MINIX [8].

Литература

1. Mario Aldea Rivas and Michael González Harbour. Early experience with an implementation of the POSIX.13 minimal real-time operating system for embedded applications. April 2005, Spain.

2. Уильям Стивенс. Unix – взаимодействие процессов. – ЗАО Издательский дом «Питер», 2003. – 658 с.

3. QNX Realtime Operating System. – [Электронный ресурс]. – Режим доступа: http://www.qnx.com/product/qnxrtos.html.

4. Микрин Е.А. Бортовые комплексы управления космическими аппаратами и проектирование их программного обеспечения. – М.: Издательство МГТУ им. Н.Э. Баумана, 2003. – 330 с.

5. LynxOS RTOS: The RTOS for complex real-time systems [Электронный ресурс]. – Режим доступа: http://www.lynuxworks.com/rtos/rtos.php.

6. V. Yodaiken. The RT-Linux approach to hard real-time. – [Электронный ресурс]. – Режим доступа: http://luz.cs.nmt.edu/~rtlinux/whitepaper/short.html.

7. Daniel P. Bovet, Marco Cesati. Understanding the Linux Kernel. O'Reilly. – Dec. 2002. – 784 c.

8. Pablo J. Rogina, Gabriel Wainer. New Real-Time Extensions to the Minix Operating System. Argentina [Электронный ресурс]. – Режим доступа: http://www.dc.uba.ar/people/proyinv/cso/rt-minix.html.

Поступила в редакцию 17.08.2006

Рецензент: д-р техн. наук, проф. И.В. Чумаченко, Национальный аэрокосмический университет им. Н.Е. Жуковского "ХАИ", Харьков.