УДК 681.3

DOI: http://dx.doi.org/10.20535/2219-3804162017114105

Зинченко С. В.[1], **Зинченко В. П.**[2], *доцент, к.т.н.*

ОСОБЕННОСТИ СИСТЕМ РЕАЛЬНОГО ВРЕМЕНИ

En The purpose of the work is to analyze the features and functioning of the RTS, RT operating systems (RTOS) and the development of applied programs (AP) for RTS; RTS standards; characteristics and analysis of the RTOS; problems of extensions of RT based on Linux and Windows.

The main differences between the RTOS and the general purpose OS are shown: the main task is to have time to react to events at the site; an RTOS is a tool for creating a specific SRT. The following characteristic functions of RTOS are distinguished: multitasking with preemption; realizing priorities for threads; predictable synchronization mechanisms; providing inheritance of priorities; behavior is known and predictable; for all scenarios of the STR the maximum response time is determined.

It is shown that the processing of events in SRT depends on the field of application and hardware; events are reported via IRQ, so the key parameter is the response time of the interrupt latency. There is no generally accepted method for determining this parameter, but it can be predicted/ computed. The reaction time for an event is calculated by the value of its components, taking into account the worst situation for the SRT (the microprocessor is loaded, and actions that block interrupts). The evaluation of response times to the interruption of the RTOS is performed based on the data of the target SRT testing. Estimating the processing time for parallel events is determined by the context switching time.

It was noted that the RTOSRTS is running on industrial computers in connection with which the RTOS should work in diskless execution, support, as much as possible, the microprocessor and types of special equipment, have tools for creating and debugging AP.

The RTS standards (POSIX, DO-178A/ B/ C, ARINC-653, ED-12B, OSEK/ VDX (OSEK), SCEPTRE, TCSEC, etc.) are analyzed, their characteristics are given and it is shown that the importance of standards is that they are an axiomatic basis that defines the definitions of objects and concepts. It is shown that, from the point of view of the programmer, "Common Criteria ..." can be considered as a set of libraries which helps to write the tasks on security, typical security profiles, etc.

Such properties of common RTOS are considered: POSIX compliance; rapidity and efficiency; they are modular and scalable; they support two-stage interrupt processing, have planning schemes, high resolution timers and counters; there are mechanisms for swapping and protecting memory; tasks and the kernel are executed in a single address space; correspond to Host-Target or SelfHosted.

The analysis of the RTOS (CHORUS; LynxOS; OS-9; pSOS; RTC; VRTX; VxWorks; QNX, etc.) is presented in the format: name, main characteristics: type; architecture; standard; Properties as RTOS; operating system (host); processors

[1] *Институт кибернетики им. В. М. Глушкова НАН Украины*
[2] *НТУУ «Киевский политехнический институт им. Игоря Сикорского», кафедра автоматизации экспериментальных исследований*

(target); communication lines host-target; the minimum size; means of synchronization and interaction; development tools.

The problems of adaptation to the requirements of the RT are considered. It is shown that the adaptation of Linux to the requirements of the RT is carried out in such directions: in accordance with the POSIX standard; the support of special hardware, running Linux from ROM, the resolution increase of the system timer; implementation of the preemption mechanism for the kernel. There are two ways to implement preemption for the Linux kernel: to rewrite the kernel, to implement the mechanism of preemption as a microcoder for scheduling interrupts and tasks.

W'NT adaptation, based on the elements of the RT, was carried out in two ways: (1) the use of the conventional RTOS core as an addition to the core W'NT (the technology of using two OS on one computer); (2) PB integrating into W'NT by avoiding delays and hangs with the help of the RT extensions.

The use of digital technologies in the aviation and space industry raises the need for training specialists, which in turn requires the availability of teaching aids, teaching staff and a technical training base. The experience of applying RTS in solving problems of automation of experimental research allows organizing the training of students in studying the principles of constructing and implementing components of the RTS.

Ua Розглянуті особливості систем реального часу (СРЧ), функціонування СРЧ, операціонних систем РЧ (ОСРЧ) і розробки додатків для них; стандарти СРЧ; характеристики і аналіз ОСРЧ; проблеми розширень РЧ на основі Linux і Windows.

Введение

Системы реального времени (СРВ) – важное и актуальное направление современных технологий, так как все автоматизированные комплексы, от которых зависят жизни и здоровье людей, являются СРВ. Проблемы СРВ затрагивают как фундаментальные вопросы (многозадачность, прерывания, межзадачное взаимодействие и синхронизация), так их и детали функционирования [1].

СРВ – это аппаратно-программные комплексы, реагирующие за предсказуемое время на непредсказуемый поток внешних событий. Это означает, что СРВ успевает реагировать на: событие на объекте в течении критического времени (определяется объектом и событием, поэтому время реакции системы должно быть предсказано при ее создании; отсутствие реакции в предсказанное время считается ошибкой); на два и больше одновременно происходящих внешних события в течении критических интервалов времени для этих событий.

Разделяют СРВ на системы жесткого реального времени (РВ) (не допускают никаких задержек реакции ни при каких условиях) и системы мягкого РВ (допускают задержку реакции на события, что может привести к снижению производительности системы в целом). К первым относятся бортовые системы управления, системы аварийной защиты, регистраторы аварийных событий и т.п. Ко вторым относятся компьютерные сети, кото-

рые допускают задержку обработки пакетов (повторные посылки). Данные при этом не теряются, но производительность сети снижается.

ОСРВ/*RTOS* – это операционная система, которая: имеет необходимые средств для организации работы в условиях жесткого РВ; фиксирует сбой в работе, если не удовлетворены временные ограничения; обеспечивает требуемый уровень сервиса в заданный промежуток времени (*POSIX* 1003.1 [2]); обеспечивает не только логическую правильность работы прикладной программы (ПП), но и время получения результата. Отметим, что понятие «РВ» в СРВ отличается от других трактовок [1, 3].

Примененяются СРВ в военной, авиа-космической областях; промышленности; автомобилестроении; энергетике, телекоммуникациях; банковском оборудовании; товарах потребления; компьютерном и офисном оборудовании [3].

Постановка задачи

Анализ особенностей и функционирования СРВ, операционных систем РВ (ОСРВ) и разработки прикладных програм (ПП) для СРВ; стандарты СРВ; характеристики и анализ ОСРВ; проблемы расширений РВ на основе *Linux* и *Windows*.

Отличие ОСРВ от ОС общего назначения (ОСОН)

ОСОН (особенно многопользовательские) ориентированы на оптимальное распределение ресурсов компьютера между пользователями и задачами (системы разделения времени). В ОСРВ – это второстепенная задача, так как главная задача – успеть среагировать на события, происходящие на объекте.

ОСРВ всегда связана с аппаратурой, объектом, событиями на объекте и ориентирована на обработку внешних событий. Именно это приводит к коренным отличиям ОСРВ от ОСОН в структуре, функциях ядра, построении ввода-вывода. Часто внешний интерфейс ОСРВ похож на ОСОН, но ОСРВ устроена совершенно иначе, и ее применение всегда конкретно. ОСОН – это готовый набор приложений, а ОСРВ является инструментом для создания конкретной СРВ.

Характерные функции ОСРВ: многозадачность с вытеснением (*preemptable*); реализует приоритеты для потоков; поддерживает предсказуемые механизмы синхронизации; обеспечивает механизм наследования приоритетов; поведение известно и предсказуемо (задержки обработки прерываний, переключения задач, драйверов и т.п.); для всех сценариев рабочей нагрузки СРВ определяется максимальное время отклика.

Обработка событий в СРВ

Время реакции СРВ на события зависит от области применения: математическое моделирование – *мкс*; радиолокация – *мс*; складской учет – *с*; торговые операции – *мин*; управление производством – *мин*; химические реакции – *ч*. Очевидно, что времена значительно различаются: для каждой задачи необходимы соответствующие аппаратные средства; события сообщаются СРВ посредством запросов на прерывание (*IRQ*), поэтому ключевым параметром является время реакции системы на прерывание (*interrupt latency*). Общепринятого метода определения этого параметра нет, но его можно предсказать /вычислить.

События на объекте регистрируются датчиками, данные с которых передаются в модули ввода-вывода интерфейсы СРВ. Последние, получив и преобразовав информацию от датчиков, генерируют запрос на прерывание в управляющем компьютере, что и является сигналом о произошедшем событии на объекте. Получив сигнал от модуля ввода-вывода, СРВ должна запустить ПП обработки этого события.

Интервал времени – от события на объекте до выполнения первой инструкции в ПП обработки этого события – это время реакции системы на событие. При проектировании СРВ вычисляется этот интервал по значению его составляющих.

Время выполнения действий – от события на объекте до генерации прерывания не зависит от ОСРВ и целиком определяется аппаратурой, а интервал времени – от возникновения запроса на прерывание до выполнения первой инструкции обработчика – определяется целиком свойствами ОС и архитектурой компьютера. Это время всегда оценивается в худшей для СРВ ситуации, то есть в предположении, что: микропроцессор (МП) загружен; возможны другие прерывания и действия, блокирующие прерывания.

Оценка времен реакции на прерывание ОСРВ (*pSOS*+ ≤4; *VRTX* ≤5; *LunxOS* ≤7; *VxWorks*, *PDOS* ≤4.5 в мкс) выполняется на основе результатов тестирования целевой СРВ известной архитектуры с известными средствами измерения и точным значениями измеряемых промежутков времени.

Так как все ОСРВ являются многозадачными (одновременно могут обрабатывать несколько событий), то оценка времени обработки параллельных событий зависит от времени переключения контекста (*pSOS*+ ≈122; *VRTX* ≈145; *LunxOS* ≈165; *VxWorks* ≈152; *PDOS* ≈90 в мкс).

Отметим, что ОСРВ СРВ выполняется на персональных, промышленных компьютерах (ПрК), контроллерах и встраиваемых системах (составляющая часть оборудования). При выборе МП определяющим является обеспечение требуемой производительности при наименьшей тактовой частоте (тепловыделение). ПрК для управления оборудованием, обычно не имеют монитора и клавиатуры, в связи, с чем ОСРВ должна: работать в

бездисковом исполнении (важен небольшой объем исполняемого кода); поддерживать как можно больше МП и видов спецоборудования; иметь инструментарий для создания и отладки ПП [4].

Стандарты СРВ

Различия в спецификациях ОСРВ и разнообразие МП привели к проблеме стандартизации с целью упрощение переноса ПП из одной системы в другую. При этом, задача обеспечения максимальной скорости работы и компактности СРВ стоит выше задачи стандартизации. Поэтому, с одной стороны, среди СРВ преобладают системы с уникальным устройством, а с другой стороны, многие стандарты носят весьма общий характер. Отметим, что даже ОСРВ, декларирующие свою совместимость с конкретным стандартом, содержат расширения, выходящие за его рамки. Важность стандартов состоит в том, что они являются аксиоматической базой, задающей определения рассматриваемых объектов и понятий.

Известный стандарт *POSIX* (*IEEE Portable Operating System Interface for Computer Environments, IEEE* 1003.1) изначально был разработан для первых версий *UNIX*-систем. Спецификации *POSIX* (>30 стандартов) определяют механизмы взаимодействия ПП и ОС. Для ОСРВ важны 7 из них (1003.*1a-d*/.*1j* /.*21* /.2*h*), и только 3 первых поддерживаются коммерческими ОС.

Некоторые компании предложили в качестве стандарта СРВ свои спецификации. Так поступила компания *TRON* (*the RTOS Nucleus*, Япония), разработавшая спецификации *ITRON*1, µ*ITRON* и *ITRON*2 для 8, 16, 32 – разрядных МП соответственно. Военная и аэрокосмическая отрасли предъявляют жесткие требования к целевым СРВ, что изложено в стандартах *DO*-178B, *ARINC*-653 (США) и *ED*-12*B* (Европа, аналог *DO*-178*B*). Распространен стандарт *OSEK*/*VDX* (*OSEK*), первоначально созданный для систем автомобильной индустрии.

Стандарт *POSIX* был создан как интерфейс сервисов ОС для переносимых ПП, впоследствии расширен особенностями режима РВ, и в настоящее время представляет семейство стандартов *IEEE Std* 1003.*n*, а именно: 1003.*1a*/*OS Definition* (определяет базовые интерфейсы); 1003.1*b* /*Realtime Extensions* (расширения РВ с такими алгоритмами планирования обработки процессов: *SCHED_FIFO* (до завершения в режиме *FIFO*); *SCHED_RR* /*Round Robin* (процессам выделяются кванты времени); *SCHED_OTHER* (произвольный и не переносимый); 003.1*c*/*Threads* (многопоточная обработка внутри процесса); 1003.1*d* (дополнительное расширения РВ; 1003.21 (распределенные СРВ); 1003.2*h* (сервисы) [3].

Стандарт *DO*-178*A* /*B* /*C* создан *RTCA* (*Radio Technical Commission for Aeronautics*) для ПП бортовых авиационных систем. Стандарт предусматривает 5 уровней серьезности отказа (*A* – катастрофический, *B* – опас-

ный, *C* – существенный, *D* – несущественный, *E* – не влияющий). На каждом уровне определен набор требований к ПП, которые гарантируют работоспособность СРВ при возникновении отказов данного уровня.

Стандарт *ARINC*-653 (*ARINC – Avionics Application Software Standard Interface*) определяет универсальный программный интерфейс *APEX* (*Application Executive*) между ОС авиационного компьютера и ПП. Требования к интерфейсу между ПП и сервисами ОС определяются так, чтобы разрешить ПП контролировать диспетчеризацию, связь и состояние внутренних обрабатываемых элементов. В новой редакции *ARING*-653 введена архитектура изолированных (*partitioning*) виртуальных машин [5].

Стандарт *OSEK*/*VDX* (*OSEK* – автомобильный консорциум (*BMW, Bosch, Daimler Chrysler, Opel, Siemens, Volkswagen*) и университет в Карлсруэ (Германия); *VDX* (*Vehicle Distributed eXecutive*) – французские компании *PSA* и *Renault*). Первоначально *OSEK*/*VDX* был разработан как стандарта открытой архитектуры ОС и *API* для автомобильных СРВ, и со временем распространился на другие отрасли промышленности. *OSEK*/*VDX* включает три стандарта: ОС/*OS*; коммуникаций (*COM*); сетевого менеджера (*NM*). Определен некий реализационный язык (*OIL*). Эффективность *OSEK* состоит в интеграции его компонентов [6].

Стандарт *SCEPTRE* (*Standartisation du Coeur des Executifs des Produits Temps Reel Europeens*). В нем: изложены спецификации для промышленных ПП; даны определения, методы и подходы, используемые в СРВ; определены задачи ОСРВ: адекватность поставленной задаче, безопасность, минимальная стоимость, максимальная производительность, переносимость, адаптивность, модульность [7].

Сервис ОС в стандарте разделен на следующие группы: коммуникации, синхронизация процессов, контроль и планирование задач, управление памятью, временем, прерываниями и оборудованием ввода-вывода, высокоуровневый интерфейс ввода-вывода и управления периферийными устройствами, управление файлами и транзакциями, обработка ошибок и исключений[8].

Основные группы функций стандарта такие: выделение памяти и адресация объектов, создание и удаление объектов, доступ к другим компьютерам. Задачи делятся на два класса. Прямые непосредственные задачи обеспечивают интерфейс между ПП и ее внешним окружением, например, задачи ввода-вывода, обработки событий таймера. Такие задачи активизируются непосредственно сигналом, приходящим с управляемого объекта, посредством механизма прерываний. Косвенные отложенные задачи – это задачи, активизируемые другими задачами, и оперируют данными, полученными от прямых или от других косвенных задач.

Возможные состояния задач: не существует (нет дескриптора); существует (имеет дескриптор); неисполнима (нельзя ни активизировать, ни продолжить); исполнима (можно выполнить); не обслуживается (требует

активизации или успешно завершена); обслуживается (начинает исполнение и не терминирована); ожидает (ожидает выполнения определенного условия для продолжения работы); активна (ожидает освобождения МП от других задач); готова (активна и ожидает МП); выполняется (использует МП).

Стандартом также определены основные виды межзадачного взаимодействия: обмен сигналами /событиями; коммуникация (посредством очередей); исключения; семафоры; клиент-серверное взаимодействие.

TCSEC (*Trusted Computer System Evaluation Criteria*) – стандарт критерии оценки компьютерных систем, разработан Министерством обороны США и известен как «*Orange Book*». В ряде стран были разработаны аналогичные критерии, на основе которых был создан международный стандарт «Общие критерии оценки безопасности информационных технологий» (*Common Criteria for IT Security Evaluation, ISO/IEC* 15408).

В «*Orange Book*» определены 7 уровней защиты: *A*1 – верифицированная разработка; *B*3 – домены безопасности; *B*2 – структурированная защита; *B*1 – мандатный контроль доступа; *C*2 – дискреционный контроль доступа; *C*1 – избирательная защита; *D* – минимальная защита.

«*Common Criteria …*» определяют требования обеспечения безопасности в виде оценочных уровней (*Evaluation Assurance Levels – EAL*). Их семь: *EAL*7 – наивысший уровень предполагает формальную верификацию модели объекта оценки; *EAL*6 – полуформально верифицированный и протестированный; *EAL*5 – полуформально спроектированный и протестированный; *EAL*4 – методически спроектированный, протестированный и пересмотренный; *EAL*3 – методически протестированный и проверенный (более полный чем *EAL*2); *EAL*2 – структурно протестированный; *EAL*1 - функционально протестированный.

В соответствии с требованиями «*Common Criteria …*», например, ОС оцениваются на соответствие ряду функциональных критериев и критериев доверия (профилей защиты). Существуют различные определения профилей защиты в отношении ОС, брандмауэров, смарт-карт и т.п., которые должны соответствовать определенным требованиям безопасности. Например, профиль защиты систем с разграничением доступа (*Controlled Access Protection Profile*) применяется к ОС и призван заменить уровень защиты *C*2 стандарт *TCSEC*. В соответствии с оценочными уровнями доверия сертификация на соответствие более высокому уровню означает более высокую степень уверенности в том, что система защиты работает правильно и эффективно. Уровни 5 - 7 применяются для тестирования ПП, созданных с применением специализированных технологий безопасности.

Отметим, что большинство оценок безопасности ПП сосредоточены на уровне *EAL*4 и ниже, что говорит об ограниченном применении формальных методов в этой области. С точки зрения программиста, «*Common*

Criteria …» можно рассматривать как набор библиотек, с помощью которых пишутся задания по безопасности, типовые профили защиты и т.п.

Существуют и другие стандарты, например: ГОСТ Р ИСО/МЭК 51904-2002; ГОСТ Р ИСО/МЭК 12207-99[10,11].

Анализ свойств ОСРВ

Рассмотрены такие свойства характерны для распространенных ОСРВ:

- соответствие стандарту POSIX на интерфейс ПП, обеспечивающий функционирование в режиме РВ, то есть: обладают вытесняемым ядром; имеют вытесняющий динамический механизм планирования, основанный на использовании статических приоритетов задач (реализуются в виде потоков); предоставляют механизмы синхронизации потоков; кроме POSIX имеют свой собственный API;
- являются модульными и масштабируемыми; можно разместить ядро в постоянно запоминающем устройстве (ПЗУ) встраиваемых систем; возможно добавления средств ввода-вывода, управления файлами, сетевого взаимодействия (например, TCP/IP) и т.п.;
- быстродействие и эффективность достигается за счет: построения концепции микроядра; низких накладных расходов на работу планировщика, переключение контекстов, операции блокировки/деблокировки мьютексов и т.п.; быстрой реакции на прерывание (несколько *мкс*); оптимизации критических секций в ядре (участки кода, работающих без вытеснения);
- поддерживают двухэтапную обработку прерываний: сначала исполняется короткая ПП обслуживания прерывания *ISR* (*interrupt service routine*), которая ставит в очередь на выполнение более длинную ПП обработки прерывания *IHR* (*interrupt handling routing*); первая из них выполняет минимальный объем действий, вторая – более емкие действия, например, передачу данных от внешнего устройства. Отметим, что такой механизм обработки прерываний имеется и в ядрах ОСОН. Например, в inux подобный механизм называется *upper*/*bottom half* (верхняя/нижняя половина). «*Bottom*» соответствует понятие *IHR*, «*upper*» – *ISR*. В ОС *W's* имеется механизм отложенного вызова процедур (*DPC*). ПП, вызываемая с помощью этого механизма, аналогична *IHR*.
- обладают более/менее гибкими схемами планирования: уровней приоритетов потоков (≥ 32 согласно *POSIX*, но бывает 128 и 256); для потоков с одинаковым уровнем приоритета применяются схемы планирования *FIFO* и *RR* (*round robin*); как правило, не поддерживают алгоритм *EDF* (сложно реализовать по сравнению со статическими планировщиками);

- имеют высокое разрешение таймеров и счетчиков (номинальное – *нс,* реальное – *мкс*);
- содержатся механизмы подкачки и защиты памяти; задачи и ядро исполняются в едином адресном пространстве;
- соответствуют схеме разработки ПП *Host-Target* или *SelfHosted*.

Self-Hosted – это системы (применяются на ПрК), в которых программист может разрабатывать ПП, работая в самой ОСРВ. Это предполагает, что ОСРВ поддерживает файловую систему, средства ввода-вывода, пользовательский интерфейс, имеются компиляторы, отладчик, редакторы и т.п. Достоинство таких систем – простой и наглядный механизм создания и запуска ПП, которые работают на том же компьютере, что и пользователь. Недостатком является то, что для ПрК в реальной эксплуатации часто вообще не требуется пользовательский интерфейс и возможность запуска тяжеловесных ПП вроде компилятора. Следовательно, большинство из описанных выше возможностей ОСРВ просто не используются и только зря занимают память и другие ресурсы ПрК.

Host/*Target* – это системы, в которых ОС и/или компьютер, на котором разрабатываются ПП (*host*), и ОС и/или компьютер, на котором запускаются ПП (*target*), различны. Связь между компьютерами осуществляется с помощью *COM*-порта, *Ethernet*, общей шины *VME*, *compact PCI* и др. В качестве *host* обычно используется компьютер под управлением *UNIX* /*W'NT*, в качестве *target* – ПрК /встраиваемый компьютер под управлением ОСРВ. Существуют системы, в которых на одном компьютере работает ОСОН и ОСРВ. Достоинство таких систем – использование всех ресурсов ОСОН (графический интерфейс, файловая система, быстрый МП и большой объем оперативной памяти) для создания ПП и уменьшение размеров ОСРВ за счет включения только нужных ПП компонент. Недостаток – относительная сложность программных компонент: кросс-компилятора, удаленного загрузчика и отладчика и т.д.

Отметим, что, с одной стороны, рост мощности ПрК позволяет использовать *self-hosted* на большем числе вычислительных систем. С другой стороны, увеличивающееся распространение встраиваемых систем расширяет сферу применения *host*/*target* систем, где стоимость ПП является определяющим фактором.

Анализ ОСРВ представлен в формат: название, основные характеристики: • тип; • архитектура; • стандарт; • свойства как ОСРВ; • ОС разработки (*host*); • процессоры (*target*); линии связи *host-target*; • минимальный размер; • средства синхронизации и взаимодействия; • средства разработки. Рассмотрены такие ОСРВ: *CHORUS*; *LynxOS*; *OS*-9; *pSOS*; *RTC*; *VRTX*; *VxWorks*; *QNX* и др.

Например: *LynxOS* (*Lynx RTS*): • *self-hosted*; • микроядро; • *POSIX* 1003; • многозадачность – POSIX 1003; многопроцессорность; уровней приоритетов – 255; планирование – *FIFO, round robin, Quantum*; *preemptive*

ядро; • нет; • *Intel* 80x86, *Motorola* 68xxx, *SPARC*, *PowerPC*; • полной /усеченной, только ядро – 256 /124/ 33 *Kb*; • *POSIX* 1003 (семафоры, *mutex, condvar*); • комплект разработчика (компилятор *C*/*C*++, отладчик, анализатор); *X W's*/*Motif* для *Lyux*; *Total View* (многопроцессный отладчик; ядро – 28 *Kb*, обеспечивающее диспетчеризацию прерываний, планирование задач и механизмы их синхронизации; допускает добавления *KPI* (*kernel plug-ins*), и эту ОСРВ можно использовать как ОСОН типа *UNIX* для разработки ПП; поддерживает механизмы защиты памяти и подкачки.

Специализированные ОСРВ проектируются под конкретную модель МП/задачу и имеют такие преимущества: высокая производительность, оптимальный состав оборудования, наибольшая компактность. Недостатки: большое время разработки, высокая стоимость, непереносимость. Примерами таких систем являются ОСРВ компаний *Sony*, *Sagem* и др., и системы, разработанные под конкретную задачу, например, управление железными дорогами *TGV* во Франции.

Проблемы адаптации ОСОН к требованиям РВ

Адаптация *Linux* к требованиям РВ выполняется по следующим направлениям:

- Поддержка стандарта *POSIX* 1003.xx: 1*c* – работа с задачами (*thread*); 1*b* (расширения РВ) – реализует механизмы управления памятью и планирования задач, но не реализует механизмы работы с таймерами, сигналами, семафорами и очередями сообщений.
- Поддержка специального оборудования, например, шины *VME*; моста *VMEPCI*, выполнения *Linux* из ПЗУ, повышение разрешения таймера системы.
- Реализация механизма *preemption* для ядра (превращается в СРВ), что является сложным для *Linux*, так как все *UNIX* системы, надолго запрещают прерывания при входе в ядро, которое практически невытесняемое (*nonpreemptive*) [12].

Существуют два пути реализации *premption* для ядра *Linux*:

- переписать ядро, что позволит достичь самых качественных результатов, но на данный момент видимых успехов в этом пока нет, так как требуется выполнить большой объем работы, связанный с большим объемом ядра, и быстро изменятся ядро без учета интересов РВ;
- механизм *preemption* реализовать как микроядро, отвечающего за диспетчеризацию прерываний и задач, где ядро *Linux* работает как задача с низким приоритетом, а само ядро незначительно изменяется для предотвращения блокирования им аппаратных прерываний; например, *RT-Linux* (без *X-W's* $\leq$2,7Мб).

Адаптация *W'NT* к задачам РВ означала не что иное, как попытку применить господствующую программную технологию для создания при-

ложений РВ. Огромный набор ПП под *W's*, мощный программный интерфейс *WIN*32, наличие специалистов – естественное стремление получить в СРВ все эти возможности.

Отметим, что *W'NT* создавалась как сетевая ОС, и в нее основу были заложены элементы РВ, а именно – двухуровневая система обработки прерываний (*ISR* и *DPC*), классы РВ (процессы с приоритетами 16 – 32 планируются в соответствии с правилами РВ). Поверхностный анализ *W'NT* показывает, что она не годится для построения жестких СРВ (непредсказуема – время выполнения системных вызовов и реакции на прерывания зависит от ее загрузки; объемна; нет защиты от зависаний, др.), и даже в мягких СРВ *W'NT* можно использовать только при выполнении целого ряда ограничений.

Разработка расширений выполнялась в двух направлениях

1. Использование ядер классических ОСРВ как дополнения к ядру *W'NT* для реализации функций для связи ПП РВ и *W'NT*. Такое решения реализовано фирмами «*LP Eleknroniks*» и «*Radisys*», где параллельно с *W'NT* работает ОС *VxWorks* и *InTime* соответственно. Для «*LP Elektroniks*» это выглядит так: вначале загружается *W'NT*, затем ОС *VxWorks* и распределяется необходимая память. Далее компьютер управляется *VxWorks*, отдавая МП ядру *W'NT* тогда, когда в нем нет надобности для приложений *VxWorks*. Синхронизацию и обмен данными между ОС выполняет псевдодрайвер *TCP*/*IP*. Технология использования двух ОС на одном компьютере состоит в том, что работу с объектом выполняет ПП РВ, передавая затем результаты *W'NT* для обработки, передачи в сеть, архивирования и т.п.
2. Интегрирование РВ в *W'NT* путем исключения задержек и зависаний с помощью расширений РВ. Например, расширение РВ фирмы «*VenturCom*» *RTX* 4.2 базируются на модификациях уровня аппаратных абстракций *W'NT* (*HAL – Hardware Abstraction Layer*) – программный слой, через который драйверы взаимодействуют с аппаратурой. Модифицированный *HAL* и дополнительные функции (*RTAPI*) обеспечивают также стабильность и надежность системы, путем отслеживания краха *W'NT*, зависания ПП и блокировку прерываний. В состав *RTX* включена подсистема РВ *RTSS*, которая расширяет *W'NT* дополнительным набором объектов (аналог стандартным, но с атрибутами РВ). Эти новые объект/ нити (потоки, процессы) РВ, которые управляются специальным планировщиком РВ (256 приоритетов, алгоритм – приоритетный с вытеснением). *RTX* позволяет создавать ПП управления устройствами, используя функции *RTAPI* работы с портами ввода-вывода и физической памятью. Можно конфигурировать *W'NT* и создавать встроенные конфигурации, в т.ч. без дисков, клавиатуры и монитора.

Выводы

Применение в авиационно-космической отрасли цифровых технологий вызывает необходимость в подготовке специалистов, что, в свою очередь, требует наличия учебных пособий, преподавательских кадров и технической базы обучения. Накопленный в Украине опыт применения СРВ при решении задач автоматизации экспериментальных исследований [13 - 15] позволяет организовать обучение для студентов по изучению принципов построения, взаимодействия и реализации компонентов СРВ.

Список использованной литературы

1. Amitava Gupta, Anil Kumar Chandra Real-Time and Distributed Real-Time Systems: Theory and Applications. CRC Press,– 2016. – p. 167.
2. POSIX.1-2008 is simultaneously IEEE Std 1003.1™-2008 and The Open Group Technical Standard Base Specifications, Issue 7. http://pubs.opengroup.org/onlinepubs/9699919799/
3. Ernst-Rudiger Olderog, Henning Dierks Real-Time Systems: Formal Specification and Automatic Verification, Cambridge University Press,– 2008. – p. 344.
4. *Khan, M. A., Saeed, S., Darwish, A., Abraham, A.* Embedded and Real Time System Development: A Software Engineering Perspective, Springer,– 2014. – p. 305.
5. Sławomir Samolej ARINC Specification 653 Based Real-Time Software Engineering. http://www.scarlettproject.eu/publications/technical/ABS653RTSE_SS1-2011.pdf
6. Joshua Billert Automotive Operating Systems: OSEK/VDX. https://es.cs.uni-kl.de /publications /data/ Bill14.pdf
7. SYSTEMES TEMPS REELS http://perso.numericable.fr/pdeiberale69/patrick.deiber/pub/ os-tr.pdf
8. *Зинченко С. В.* Исследование характеристик знание ориентированных интеллектуальных систем мягкого реального времени// Комп'ютерні засоби, мережі та системи. 2012, №11. – С. 13–22.
9. *Египко В. М.* Метод и инструментальные средства обработки данных эксперимента в реальном времени / В. М. Египко, В. П. Зинченко // К., 1995. – 24 с. (Препр./АН Украины Ин-т кибернетики им. В. М. Глушкова; 95–20).
10. ГОСТ Р ИСО/МЭК 51904-2002 Программное обеспечение встроенных систем. Общие требования к разработке и документированию (Докипедия: ГОСТ Р 51904-2002 Программное обеспечение встроенных систем. Общие требования к разработке и документированию) – http://dokipedia.ru/document/5150996.

11. ГОСТ Р ИСО/МЭК 12207-99 Процессы жизненного цикла программных средств http:// www.swrit.ru/ doc/iso/12207-99.pdf.
12. *Стивенс, У.* UNIX: взаимодействие процессов. – СПб.: Питер, 2003. – 576 с.
13. *Зинченко В. П.* Алгоритмы и базовые программные модули для управления технологическими модулями Prometheus / В. П. Зинченко, С. В. Зинченко // УСиМ, 2007. – № 5. – С. 52–60.
14. *Зинченко В. П.* Исследование и реализация алгоритма адаптивного управления экспериментом / В. П. Зинченко // Проблемы управления и информатики, 2001. – № 3. – С. 58 – 69.
15. *Ли Вэй* Алгоритм управления потоком в аэродинамической трубе /Ли Вэй, С. В. Зинченко, В. П. Зинченко // УСиМ, № 3, 2014. – С. 69-78.