УДК 656.25+004.451

АДАПТАЦИЯ ЯДРА ОПЕРАЦИОННОЙ СИСТЕМЫ LINUX ДЛЯ МИКРОПРОЦЕССОРНОЙ ЦЕНТРАЛИЗАЦИИ МПЦ-МПК

КОЗЛОВ Аркадий Александрович, научный сотрудник кафедры «Автоматика и телемеханика на железных дорогах»; e-mail: arkadiikozlov@gmail.com

Петербургский государственный университет путей сообщения Императора Александра I, Санкт-Петербург

В статье рассматриваются особенности применения операционной системы Linux в микропроцессорной централизации на базе микроЭВМ и программируемых контроллеров МПЦ-МПК, построенной на основе промышленных контроллеров с процессорами архитектуры x86. Основное внимание уделено обеспечению требуемых временных характеристик программной платформы, необходимых для функционирования дублированной структуры контроллеров логики централизации по принципу «два из двух». Показано, что применение стандартной конфигурации Linux не является оптимальным для специализированной промышленной системы, предъявляющей жесткие требования к времени реакции. Рассмотрены подходы к масштабированию ядра Linux, включению механизма вытеснения ядра CONFIG_PREEMPT, использованию высокоточных таймеров (high-resolution timers) и режимов планирования реального времени SCHED_FIFO для критичных прикладных процессов. Для оценки временных характеристик программной платформы предлагается использовать как стандартный тест cyclictest, так и специализированное тестовое программное обеспечение, выполняющее обмен данными между контроллерами и логику централизации по Ethernet. Приводится подход к экспериментальной оценке времени реакции операционной системы и времени взаимодействия прикладных процессов.

Ключевые слова: Linux, CONFIG_PREEMPT, SCHED_FIFO, высокоточный таймер, система реального времени, время реакции системы, системы железнодорожной автоматики и телемеханики, безопасные системы управления

DOI: 10.20295/2412-9186-2026-12-03-193-201

▼ Введение

Современные системы железнодорожной автоматики и телемеханики (СЖАТ) предъявляют повышенные требования не только к функциональности программного обеспечения, но и к предсказуемости времени его выполнения. Для систем, реализующих функции управления и контроля объектов железнодорожной инфраструктуры, существенное значение имеет способность программно-аппаратной платформы своевременно реагировать на внешние события, обрабатывать поступающие данные и обеспечивать взаимодействие между резервированными вычислительными каналами.

Микропроцессорная централизация на базе микроЭВМ и программируемых контроллеров (МПЦ-МПК) [1] построена на основе стандартных промышленных контроллеров с процессорами семейства Intel x86. Программная платформа контроллеров использует операционную систему (ОС) Linux, под управлением которой функционирует специализированное прикладное программное обеспечение системы микропроцессорной централизации.

Использование ОС Linux в промышленной системе позволяет применять широкий набор готовых программных и аппаратных решений, однако стандартная конфигурация универсальной ОС содержит большое количество компонентов, которые не требуются специализированному контроллеру. Кроме того, обычная конфигурация ОС Linux ориентирована прежде всего на универсальность и высокую производительность, а не на обеспечение минимальной и предсказуемой задержки реакции на события.

Для системы МПЦ-МПК это обстоятельство имеет принципиальное значение. В системе используется дублированная структура, в которой контроллеры логики централизации (КЛЦ) выполняют обмен информацией с контроллерами безопасного сопряжения

с объектами (КБСО) и осуществляют взаимное сравнение результатов. Время, затрачиваемое на выполнение соответствующих операций и обмен данными между КЛЦ, должно находиться в пределах установленных требований. Проектное требование задержки выполнения взаимного сравнения в системе МПЦ-МПК не должно превышать 10 мс.

Таким образом, задача применения ОС Linux в МПЦ-МПК состоит не только в реализации функционирования прикладного программного обеспечения. Необходимо также сформировать такую конфигурацию ОС, при которой ее собственная активность не будет препятствовать выполнению критичных по времени функций прикладного программного обеспечения [2].

Целью настоящей работы является рассмотрение адаптации ядра Linux для МПЦ-МПК, а также исследование влияния конфигурации ядра и параметров планирования на время реакции программной платформы системы микропроцессорной централизации.

Архитектура программно-аппаратной платформы системы МПЦ-МПК

Как было ранее указано, система МПЦ-МПК имеет дублированную структуру. В состав вычислительной части входят два КЛЦ, каждый из которых выполняет обработку информации, поступающей от КБСО, а также формирует и передает запросы в эти КБСО. Запросы, формируемые КЛЦ и направляемые в КБСО, взаимно сравниваются между КЛЦ для обеспечения согласованности работы дублированных вычислительных каналов. Упрощенно взаимодействие компонентов может быть представлено следующим образом (рис. 1).

Каждый КЛЦ самостоятельно получает и обрабатывает информацию от КБСО, который, в свою очередь, подключен к реальным объектам системы (стрелки, светофоры, рельсовые цепи и т. д.). Такой подход соответствует принципу «два из двух» [1, 3], при котором функционирование системы предполагает одновременную обработку информации двумя независимыми каналами и сравнение результатов их работы. Для выполнения этих функций прикладное программное обеспечение МПЦ-МПК использует многопроцессную архитектуру [4]. Различные процессы выполняют отдельные функции системы, причем часть процессов является критичной с точки зрения временных характеристик.

Важной особенностью является использование прикладным программным обеспечением высокоточных таймеров ОС Linux — high-resolution timers (hrtimer) [5, 6]. Они позволяют формировать события с высокой временной точностью и использовать их для периодического запуска или синхронизации критичных операций. Однако наличие высокоточного таймера само по себе не означает, что прикладной процесс будет немедленно выполнять требуемую операцию после наступления момента срабатывания таймера. Реальное

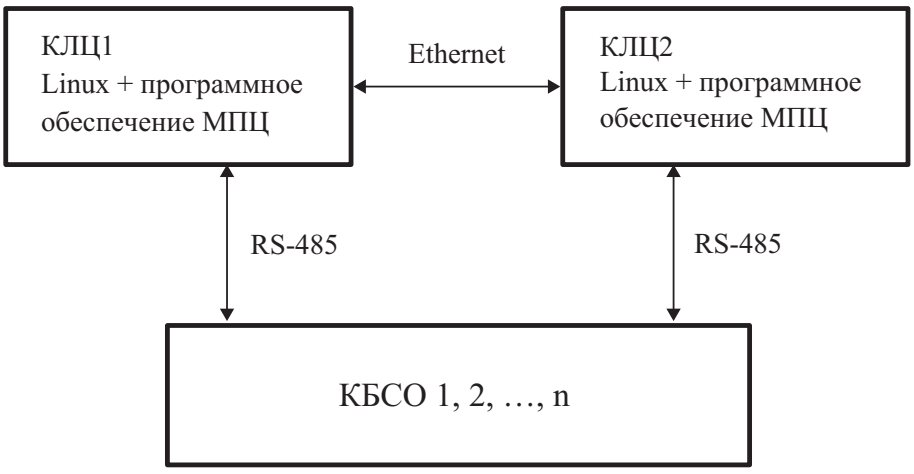

Рис. 1. Упрощенная архитектура программно–аппаратной платформы системы МПЦ–МПК

время начала выполнения процесса зависит от работы планировщика и текущего состояния ядра операционной системы. Именно поэтому параметры ядра Linux и используемые режимы планирования процессов имеют непосредственное значение для временных характеристик МПЦ-МПК.

Масштабирование ядра Linux как способ оптимизации программной платформы

Использование полного универсального дистрибутива ОС Linux в специализированном промышленном контроллере является избыточным для встраиваемых систем реального времени. Такой дистрибутив содержит большое количество драйверов, подсистем и пользовательских компонентов, которые не требуются для выполнения функций конкретного устройства. Поэтому при разработке программной платформы МПЦ-МПК был выбран подход масштабирования Linux. Под масштабированием в данном случае понимается формирование минимально необходимой конфигурации ядра и пользовательского пространства ОС, содержащей только функциональность, необходимую для работы аппаратной платформы, системных компонентов и прикладного программного обеспечения системы МПЦ-МПК.

Основной целью такой минимизации является не только уменьшение размера ядра и экономия оперативной памяти. Исключение неиспользуемой функциональности позволяет также сократить количество механизмов ОС, потенциально способных влиять на выполнение прикладного программного обеспечения.

Для специализированной промышленной системы это имеет несколько преимуществ:

- уменьшение потребления системных ресурсов;
- сокращение количества выполняемого и потенциально активируемого кода;
- уменьшение количества источников фоновой активности;
- повышение предсказуемости поведения ОС;
- упрощение анализа программной платформы;
- уменьшение потенциальной поверхности атаки [7].

Масштабирование позволяет рассматривать ОС Linux не только как средство уменьшения размера операционной системы, но и как способ формирования контролируемой программной среды для выполнения задач с временными ограничениями.

Вытеснение ядра Linux и время реакции прикладного программного обеспечения

Одним из основных механизмов, использованных при конфигурировании ядра Linux для МПЦ-МПК, является опция вытеснения ядра (CONFIG_PREEMPT) [8, 9]. В стандартной конфигурации ОС выполнение прикладной задачи может быть задержано активностью ядра. Например, при обслуживании системных вызовов, сетевого взаимодействия, обработке различных событий или выполнении других задач ядро может некоторое время продолжать выполнение текущего кода до момента, когда станет возможным переключение на требуемый прикладной процесс. Для универсальной ОС такие задержки обычно не являются критичными, однако в системе МПЦ-МПК ситуация отличается, поскольку часть прикладных процессов должна реагировать на события в пределах ограниченного временного интервала.

Подобная ситуация может возникнуть при удаленном подключении к контроллеру КЛЦ по SSH[1]. В процессе инициализации и обслуживания SSH-сессии в системе появляется дополнительная активность, которая может приводить к влиянию на временные характеристики выполнения прикладных задач. Вытесняемое ядро позволяет уменьшить интервал, в течение которого выполнение прикладной задачи может быть задержано активностью ядра, хотя конкретная величина задержки зависит от конфигурации системы и характера нагрузки. Исторически механизм вытеснения ядра появился в период разработки Linux 2.5. В частности, в Linux 2.5.4 была добавлена соответствующая опция ядра. Для МПЦ-МПК данный механизм используется уже не как

[1] SSH (от англ. Secure Shell) — это сетевой протокол, который позволяет безопасно подключаться к удаленному компьютеру или серверу и управлять им через командную строку.

средство повышения интерактивности универсальной системы, а как один из инструментов обеспечения требуемого времени реакции специализированного прикладного программного обеспечения.

Взаимодействие механизмов CONFIG_PREEMPT, high-resolution timer и SCHED_FIFO в программно-аппаратной среде

Важным аспектом работы программной платформы МПЦ-МПК является совместное использование нескольких механизмов Linux. Прикладное программное обеспечение использует high-resolution timer для формирования событий, требующих выполнения соответствующих процессов. После срабатывания таймера процесс становится готовым к выполнению, однако фактический момент получения процессора определяется планировщиком.

Для процессов, критичных по времени, может использоваться режим планирования реального времени SCHED_FIFO с соответствующим приоритетом [10]. В упрощенном виде последовательность обработки события выглядит следующим образом (рис. 2).

При использовании CONFIG_PREEMPT это позволяет уменьшить задержку между моментом возникновения события и фактическим началом выполнения критичного прикладного процесса. При этом CONFIG_PREEMPT и SCHED_FIFO выполняют разные функции. CONFIG_PREEMPT определяет возможность вытеснения выполняющегося кода ядра в соответствующих условиях, а SCHED_FIFO задает правила приоритетного планирования прикладного процесса.

Важно подчеркнуть, что CONFIG_PREEMPT, high-resolution timer и SCHED_FIFO не являются абсолютной гарантией фиксированного времени реакции. В ядре остаются участки выполнения, обработка прерываний, блокировки и другие механизмы, которые могут влиять на фактическую задержку. Поэтому возможность достижения требуемого времени реакции должна подтверждаться экспериментально.

Дополнительные механизмы настройки ядра ОС Linux

Рассмотренные выше меры не исчерпывают всех возможностей, полученных в результате адаптации ядра Linux для применения в системе МПЦ-МПК. Помимо включения опции вытеснения ядра и использования приоритетных режимов планирования для критичных процессов, на временные характеристики системы оказывают влияние и другие механизмы ОС и аппаратной платформы. К ним относятся, в частности:

- управление группой процессов (autogroup);
- управление состояниями простоя процессора (CPU idle states);
- управление частотой и режимами энергопотребления процессора;
- использование режима производительности (performance);

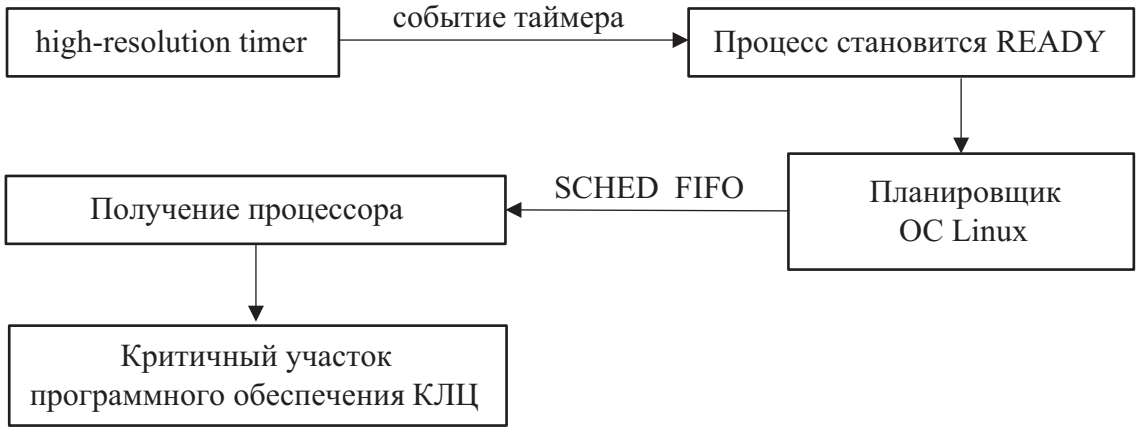

Рис. 2. Диаграмма последовательности обработки события

- конфигурация пользовательского пространства;
- минимизация состава BusyBox[2];
- другие параметры планировщика и ОС.

Механизм autogroup, например, может изменять распределение процессорного времени между группами процессов. Исключение ненужной для специализированной системы функциональности позволяет сделать поведение планировщика более предсказуемым и уменьшить влияние запуска дополнительных процессов на выполнение критичных задач. Аналогично переход процессора в состояние глубокого простоя или изменение его частоты может приводить к дополнительным задержкам, которые для обычной вычислительной системы практически незаметны, но могут иметь значение при анализе временных характеристик системы с требованиями реального времени [8, 11].

Использование режима performance позволяет исключить влияние динамического изменения частоты процессора на временные характеристики выполнения критичных задач.

Пользовательское пространство системы также масштабируется. В МПЦ-МПК используется минимальная конфигурация BusyBox, включающая только необходимые системные утилиты и средства запуска программного обеспечения.

Совокупность подобных настроек позволяет дополнительно снизить влияние фоновой активности операционной системы и аппаратных механизмов управления процессором на выполнение критичных по времени процессов.

Подробное рассмотрение перечисленных механизмов, а также результатов их индивидуального и совместного влияния на временные характеристики системы в рамках ограниченного объема настоящей статьи не проводится. Основное внимание в данной работе уделено средствам, непосредственно связанным с обеспечением требуемого времени реакции ОС и их экспериментальной проверке.

[2] BusyBox — набор UNIX-утилит командной строки, используемый в качестве основного интерфейса во встраиваемых операционных системах.

Методика экспериментального исследования влияния конфигурации ядра Linux на прикладное ПО системы МПЦ-МПК

Для оценки влияния конфигурации Linux на временные характеристики МПЦ-МПК целесообразно использовать два взаимодополняющих метода:

- первый метод основан на применении стандартного тестового средства cyclictest, предназначенного для измерения задержек реакции системы на периодические события таймера;
- второй метод основан на специализированном тестовом программном обеспечении, разработанном на языке C++ и выполняющем сетевой обмен между двумя контроллерами КЛЦ.

Такое сочетание позволяет исследовать систему на двух уровнях. Cyclictest дает возможность оценить системную задержку в условиях заданной тестовой нагрузки, то есть интервал между ожидаемым моментом выполнения периодической задачи и фактическим моментом ее запуска. Специализированное тестовое ПО позволяет оценить уже прикладной уровень, то есть какое время требуется для взаимодействия двух прикладных процессов, выполняющихся на разных КЛЦ.

Такой подход дает возможность отдельно оценить задержки, возникающие непосредственно при планировании выполнения задач ядром Linux, не смешивая их с задержками, обусловленными сетевым стеком, драйверами, прикладным программным обеспечением и другими компонентами системы.

Измерение времени реакции прикладного программного обеспечения с помощью cyclictest

Принцип работы тестовой программы cyclictest следующий: в ходе тестирования создается периодическая задача, для которой фиксируются ожидаемый момент пробуждения и фактический момент начала выполнения. Разность между этими моментами характеризует задержку реакции системы. Для рассматриваемой задачи особый интерес представляет именно минимальная задержка [12, 13].

Среднее значение задержки характеризует типичное поведение системы, однако для системы с жесткими временными требованиями критичным может оказаться единичный случай существенно большей задержки. Поэтому при проведении эксперимента целесообразно фиксировать:

- минимальную задержку;
- среднюю задержку;
- максимальную задержку;
- количество измерений;
- продолжительность теста;
- параметры нагрузки;
- конфигурацию ядра.

Предлагается сравнить как минимум две конфигурации ядра: без CONFIG_PREEMPT и с CONFIG_PREEMPT.

Дополнительно тестирование проводится при наличии фоновой активности системы, например при установлении SSH-сессии с КЛЦ. В табл. 1 представлены результаты тестирования.

Полученные данные показывают, насколько изменяется максимальная задержка реакции системы при включении вытеснения ядра, использовании режима планирования SCHED_FIFO и при воздействии фоновой активности. Таким образом, наилучший результат при воздействии фоновой активности достигается при использовании CONFIG_PREEMPT и режима планирования процессов SCHED_FIFO.

Следует учитывать, что cyclictest характеризует задержку реакции ОС в рамках данного теста и не является непосредственным измерением времени синхронизации двух КЛЦ. Поэтому результаты cyclictest должны рассматриваться совместно с результатами прикладного сетевого тестирования.

Измерение времени взаимодействия КЛЦ по Ethernet

Для проверки поведения реального прикладного программного обеспечения МПЦ-МПК была разработана тестовая программа на языке C++, выполняющая обмен данными между двумя контроллерами. Упрощенно схема эксперимента представлена на рис. 3.

Один прикладной процесс на КЛЦ1 передает сообщение другому процессу на КЛЦ2 по Ethernet. После получения сообщения второй процесс формирует ответ. По времени между отправкой исходного сообщения и получением ответа определяется длительность полного цикла обмена. Полученное значение включает задержки выполнения прикладных процессов, планирования, сетевого стека, драйвера и передачи данных по Ethernet.

Таким образом, тест можно рассматривать как своеобразный прикладной вариант измерения времени взаимодействия между двумя процессами, работающими на разных КЛЦ. Для каждого эксперимента выполняется серия обменов, после чего определяются минимальное, среднее и максимальное время прохождения цикла. Условия эксперимента остаются такими же, как и в случае с cyclictest. В табл. 2 представлены результаты тестирования.

Полученные результаты показывают, насколько изменяется максимальная задержка взаимодействия двух процессов на разных

ТАБЛИЦА 1. Задержка времени реакции системы

Конфигурация ядра	Условия	Мин., мкс	Средн., мкс	Макс., мкс
Без CONFIG_PREEMPT	Без нагрузки	16	33	1236
	SSH-сессия	16	33	5292
Без CONFIG_PREEMPT, SCHED_FIFO	Без нагрузки	15	20	81
	SSH-сессия	16	22	165
CONFIG_PREEMPT	Без нагрузки	15	26	1089
	SSH-сессия	13	27	4638
CONFIG_PREEMPT, SCHED_FIFO	Без нагрузки	14	19	43
	SSH-сессия	12	18	90

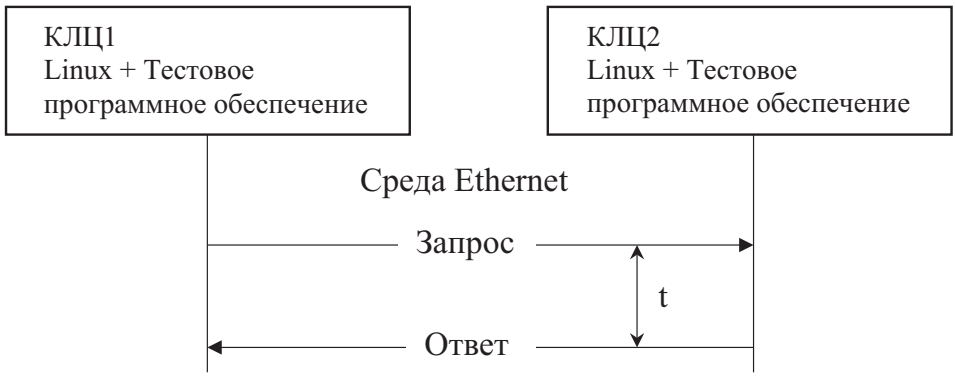

Рис. 3. Упрощенная схема эксперимента

ТАБЛИЦА 2. Длительность цикла обмена

Конфигурация ядра	Условия	Мин., мкс	Средн., мкс	Макс., мкс
Без CONFIG_PREEMPT	Без нагрузки	265	276	311
	SSH-сессия	286	313	12 155
Без CONFIG_PREEMPT, SCHED_FIFO	Без нагрузки	274	293	331
	SSH-сессия	251	287	3025
CONFIG_PREEMPT	Без нагрузки	241	269	275
	SSH-сессия	264	296	7265
CONFIG_PREEMPT, SCHED_FIFO	Без нагрузки	242	253	265
	SSH-сессия	214	269	1531

КЛЦ при включении опции вытеснения ядра CONFIG_PREEMPT, использовании режима планирования SCHED_FIFO и при воздействии фоновой активности. Таким образом, наилучший результат при воздействии фоновой активности достигается с использованием CONFIG_PREEMPT и режима планирования процессов SCHED_FIFO.

Заключение

На основании проведенных испытаний можно оценить влияние конфигурации Linux на временные характеристики программной платформы МПЦ-МПК.

Включение опции CONFIG_PREEMPT уменьшает задержки, связанные с выполнением кода ядра, а использование SCHED_FIFO для критичных процессов позволяет задавать их приоритет относительно обычных процессов. Однако по приведенным данным нельзя отделить вклад каждого механизма полностью, поскольку наиболее выраженное улучшение наблюдается при их совместном использовании.

Использование high-resolution timers создает механизм формирования событий с высокой временной точностью, однако фактическое время запуска процесса определяется не только таймером, но и планированием, а также текущим состоянием системы. При этом cyclictest и специализированное тестовое ПО решают разные задачи. Cyclictest показывает системную задержку в рамках локального теста, тогда как сетевой тест показывает результат взаимодействия нескольких уровней программной платформы: планировщика, ядра, сетевого стека, драйвера Ethernet и прикладных процессов. Поэтому согласованность выводов двух типов экспериментов является более убедительным подтверждением пригодности выбранной конфигурации ядра ОС Linux для системы МПЦ-МПК, чем использование только одного метода измерения.

Рассмотренная конфигурация Linux является результатом развития программной платформы системы МПЦ-МПК и не ограничивает возможности дальнейшего совершенствования

системы. В настоящее время проводятся исследования возможности применения современных версий ядра Linux, в том числе Linux 5.15 и Linux 6.18. Обе версии являются ветками с длительной поддержкой.

Одним из существенных изменений современных ядер является развитие планировщика процессов. В Linux 4.14, использовавшемся в рассматриваемой программной платформе, основным планировщиком для обычных задач является CFS (Completely Fair Scheduler) [14]. Начиная с Linux 6.6 ядро начало переход от CFS к EEVDF (Earliest Eligible Virtual Deadline First) [15]. В EEVDF при выборе следующей задачи учитывается ее виртуальный срок выполнения, при этом механизм допускает более благоприятное планирование задач, чувствительных к задержкам. Для систем, предъявляющих требования к времени реакции, представляет интерес также развитие механизма PREEMPT_RT. В современных версиях Linux поддержка real-time-возможностей ядра существенно расширена по сравнению с обычным CONFIG_PREEMPT.

Применение современных механизмов планирования и real-time-возможностей потенциально позволяет дополнительно снизить и сделать более предсказуемыми задержки выполнения критичных по времени процессов.

Перспективным направлением дальнейших исследований является применение современных версий ядра Linux, в частности 6.18, развитие которого сопровождается совершенствованием планировщика процессов и механизмов поддержки систем реального времени. Исследование этих возможностей позволит определить, насколько современные механизмы ОС Linux способны обеспечить дополнительные преимущества при построении программной платформы не только для системы МПЦ-МПК, но и для аналогичных систем управления технологическими процессами в промышленности. ▲

Список источников

1. Сапожников В. В., Никитин А. Б. Микропроцессорная система электрической централизации МПЦ-МПК // Наука и транспорт. 2009. № S. С. 18–21. EDN VBNHVV
2. Дауд А., Квашнин В. М. Linux в системах реального времени: подходы, возможности и анализ производительности // Вестник науки. 2024. Т. 4, № 12 (81). С. 1580–1591. EDN DKXHYC
3. Якушев А. Я., Плакс А. В., Опарина Е. В. Повышение безопасности микропроцессорной подсистемы управления аппаратами электроподвижного состава // Известия Петербургского университета путей сообщения. 2015. № 4 (45). С. 85–92. EDN VJKBYT
4. Волосатова Т. М., Киселев И. А., Князева С. В. Многопоточная обработка в POSIX-стандарте // Восточно-Европейский научный журнал. 2021. № 12–3 (76). С. 37–39. DOI: 10.31618/ESSA.2782-1994.2021.3.76.214. EDN CDWIMS
5. Linux Kernel Documentation. High-Resolution Timers and Dynamic Ticks Design Notes. URL: https://docs.kernel.org/timers/highres.html (дата обращения: 25.05.2026).
6. Linux Kernel Documentation. Hrtimers — Subsystem for High-Resolution Kernel Timers. URL: https://docs.kernel.org/timers/hrtimers.html (дата обращения: 25.05.2026).
7. Девянин П. Н., Тележников В. Ю., Хорошилов А. В. Формирование методологии разработки безопасного системного программного обеспечения на примере операционных систем // Труды Института системного программирования РАН. 2021. Т. 33, № 5. С. 25–40. DOI: 10.15514/ISPRAS-2021-33(5)-2. EDN WBXBTQ
8. Linux Kernel Documentation. Real-Time Preemption (PREEMPT_RT). URL: https://docs.kernel.org/core-api/real-time/ (дата обращения: 25.05.2026).
9. Linux Kernel Documentation. How Realtime Kernels Differ. URL: https://www.kernel.org/doc/html/latest/core-api/real-time/differences.html (дата обращения: 25.05.2026).
10. Overview of CPU Scheduling. Linux Man-Pages. URL: https://man7.org/linux/man-pages/man7/sched.7.html (дата обращения: 25.05.2026).
11. Linux Kernel Documentation. Clock Sources, Clock Events, Sched_Clock() and Delay Timers. URL: https://docs.kernel.org/timers/timekeeping.html (дата обращения: 25.05.2026).
12. Cyclictest. High-Resolution Test Program. URL: https://github.com/jlelli/rt-tests/blob/master/src/cyclictest/cyclictest.8 (дата обращения: 25.05.2026).
13. Real-Time Field Bus Systems with Linux. URL: https://pub.tik.ee.ethz.ch/students/2015-HS/SA-2015-21.pdf (дата обращения: 25.05.2026).
14. Overview of CPU Scheduling. Linux Man-Pages. URL: https://man7.org/linux/man-pages/man7/sched.7.html (дата обращения: 25.05.2026).
15. An EEVDF CPU Scheduler for Linux. URL: https://lwn.net/Articles/925371/ (дата обращения: 25.05.2026).

TRANSPORT AUTOMATION RESEARCH. 2026. Vol. 12, no. 3, pp. 193–201
DOI: 10.20295/2412-9186-2026-12-03-193-201

Adaptation of the Linux Operating System Kernel for Microprocessor-Based Interlocking System

Information about authors:
KOZLOV Arkadij Aleksandrovich, Researcher of the "Automation and Remote Control on Railways" Department; e-mail: arkadiikozlov@gmail.com

Emperor Alexander I St. Petersburg State Transport University, Saint Petersburg

Abstract: the article examines the specific aspects of using the Linux operating system in a microprocessor-based interlocking system built on microcomputers and programmable controllers using industrial controllers with x86 architecture processors. Particular attention is paid to ensuring the required timing characteristics of the software platform necessary for the operation of a duplicated central logic controller structure based on the 2oo2 architecture. It is shown that the use of a standard Linux configuration is not optimal for a specialized industrial system with stringent response-time requirements. The approaches to scaling the Linux kernel, enabling kernel preemption using CONFIG_PREEMPT, using high-resolution timers, and applying the real-time scheduling policy SCHED_FIFO for time-critical application processes are considered. To evaluate the timing characteristics of the software platform, both the standard cyclictest benchmark and specialized test software performing data exchange between central logic controllers via Ethernet are proposed. An approach to the experimental evaluation of operating system response time and the interaction time between application processes is presented.

Keywords: Linux, CONFIG_PREEMPT, SCHED_FIFO, high-resolution timer, real-time system, system response time, railway automation and remote control systems, safety-critical control systems, train control system

References

1. Sapozhnikov V. V., Nikitin A. B. Mikroprocessornaya sistema elektricheskoj centralizacii MPTS-MPK [Microprocessor-Based Electrical Centralization System MPC-MPC], *Nauka i transport [Science and Transport]*, 2009, no. S, pp. 18–21. EDN VBNHVV (In Russian)
2. Daud A., Kvashnin V. M. Linux v sistemah real'nogo vremeni: podhody, vozmozhnosti i analiz proizvoditel'nosti [Linux in Real Time Systems: Approaches, Capabilities and Performance Analysis], *Vestnik nauki [Bulletin of Science]*, 2024, vol. 4, no. 12 (81), pp. 1580–1591. EDN DKXHYC (In Russian)
3. Yakushev A. Ya., Plaks A. V., Oparina E. V. Povyshenie bezopasnosti mikroprocessornoj podsistemy upravleniya apparatami elektropodvizhnogo sostava [Improving the Safety of the Microprocessor Control Subsystem for Electric Rolling Stock Equipment], *Izvestiya Peterburgskogo universiteta putej soobshcheniya [Proceedings of the Petersburg Transport University]*, 2015, no. 4 (45), pp. 85–92. EDN VJKBYT (In Russian)
4. Volosatova T. M., Kiselev I. A., Knyazeva S. V. Mnogopotochnaya obrabotka v POSIX standarte [Multithreaded Processing in the POSIX Standard], Vostochno-Evropejskij nauchnyj zhurnal [East European Scientific Journal], 2021, no. 12-3 (76), pp. 37–39. DOI: 10.31618/ESSA.2782-1994.2021.3.76.214. EDN CDWIMS (In Russian)
5. Linux Kernel Documentation. High Resolution Timers and Dynamic Ticks Design Notes. URL: https://docs.kernel.org/timers/highres.html (accessed: May 25, 2026).
6. Linux Kernel Documentation. Hrtimers — Subsystem for High Resolution Kernel Timers. URL: https://docs.kernel.org/timers/hrtimers.html (accessed: May 25, 2026).
7. Devyanin P. N., Telezhnikov V. Yu., Khoroshilov A. V. Formirovanie metodologii razrabotki bezopasnogo sistemnogo programmnogo obespecheniya na primere operacionnyh sistem [Developing a Methodology for Secure System Software Development Using Operating Systems as an Example], *Trudy Instituta sistemnogo programmirovaniya RAN [Proceedings of the Institute for System Programming of the RAS]*, 2021, vol. 33, no. 5, pp. 25–40. DOI: 10.15514/ISPRAS-2021-33(5)-2. EDN WBXBTQ (In Russian)
8. Linux Kernel Documentation. Real Time Preemption (PREEMPT_RT). URL: https://docs.kernel.org/core-api/real-time/ (accessed: May 25, 2026).
9. Linux Kernel Documentation. How Realtime Kernels Differ. URL: https://www.kernel.org/doc/html/latest/core-api/real-time/differences.html (accessed: May 25, 2026).
10. Overview of CPU Scheduling. Linux Man Pages. URL: https://man7.org/linux/man-pages/man7/sched.7.html (accessed: May 25, 2026).
11. Linux Kernel Documentation. Clock Sources, Clock Events, Sched_Clock() and Delay Timers. URL: https://docs.kernel.org/timers/timekeeping.html (accessed: May 25, 2026).
12. Cyclictest. High Resolution Test Program. URL: https://github.com/jlelli/rt-tests/blob/master/src/cyclictest/cyclictest.8 (accessed: May 25, 2026).
13. Real Time Field Bus Systems with Linux. URL: https://pub.tik.ee.ethz.ch/students/2015-HS/SA-2015-21.pdf (accessed: May 25, 2026).
14. Overview of CPU Scheduling. Linux Man Pages. URL: https://man7.org/linux/man-pages/man7/sched.7.html (accessed: May 25, 2026).
15. An EEVDF CPU Scheduler for Linux. URL: https://lwn.net/Articles/925371/ (accessed: May 25, 2026).