***В. А. Сартаков**, аспирант НИЯУ «МИФИ», генеральный директор ООО «Ксис Лабс», г. Москва*

***И. О. Атовмян**, докт. техн. наук, профессор, и. о. зав. кафедры Интеллектуальных управляющих систем НИЯУ «МИФИ», г. Москва*

***М. А. Заева**, доцент НИЯУ «МИФИ», г. Москва*

Опыт разработки и тестирования встраиваемой микроядерной операционной системы

Процесс проникновения во все сферы жизни современного человека различных микроэлектронных устройств можно еще более ускорить за счет рациональных подходов к разработке их программного обеспечения.

Введение

Неотъемлемой частью современного мира стали мобильные и встраиваемые устройства, такие как портативные компьютеры, сотовые телефоны, навигаторы, игровые и телевизионные приставки, бытовые и промышленные средства управления процессами и т. д. Используемые в них высокопроизводительные встраиваемые процессоры нуждаются во встраиваемой операционной системе (ОС).

Среди разнообразных архитектур ОС выделяются микроядерная и монолитно-модульная архитектуры [1]. При этом вторая получила большее распространение благодаря развитию ядра Linux. Основное отличие этих архитектур заключается в размере исполняемого кода, работающего в привилегированном режиме. В частности, в монолитно-модульных ОС драйверы устройств находятся в одном адресном пространстве с TCP/IP — стеком, менеджером памяти и другими функциональными частями ядра. Такой подход ускоряет разработку системы на ранних этапах, поскольку не требует разработки дополнительных механизмов передачи данных между составными частями ядра. В то же время этот подход характеризуется низким уровнем безопасности и отказоустойчивости — ошибка в любом драйвере может привести к сбою в работе всего ядра, а возможность получения драйвером доступа к структурам ядра угрожает безопасности пользовательских данных. В противоположность такому подходу в микроядерных ОС часть кода, работающая в привилегированном режиме, невелика, и отдельные функции ядра выполняют специальные модуль-серверы, работающие в пользовательском режиме и способные взаимодействовать между собой, например при помощи сообщений [2]. Возникновение сбоя в драйвере не приведет к отказу всей системы. Таким образом, при создании встраиваемого аппаратно-программного комплекса с высоким уровнем отказоустойчивости и безопасности необходимо использовать встраиваемую микроядерную ОС.

В настоящее время на рынке встраиваемых ОС микроядерных систем немного. Среди коммерческих продуктов лидером является QNX, доступных для применения свободно распространяемых проектов к моменту написания работы не было. Поэтому стояла задача разработки встраиваемой микроядерной ОС с открытым исходным кодом и демонстрации ее работоспособности для применения в сетевых устройствах.

Возможные подходы

Сегодня используется два основных подхода к разработке встраиваемой ОС — разработка либо полностью с нуля, либо на основе использования существующего ядра. Для реализации последнего подхода в настоящее время применяются два способа.

Первый способ заключается в преобразовании больших ОС общего назначения во встраиваемые ОС посредством упрощения функционала и уменьшения кода ядра, примером может служить разработка ОС uCLinux, за основу которой была взята ОС Linux[1].

Второй способ — противоположность первому, он состоит в наращивании мощности простейших ОС контроллеров и функционала ядра.

Подход к разработке ОС на основе существующего ядра называется портированием и сводится к переносу существующего кода с одной аппаратной платформы на другую. В данном случае решено было воспользоваться существующим микроядром общего назначения и перенести его во встраиваемую систему.

Наиболее известным микроядром с открытым кодом является микроядро Mach. Оно разработано в университете *Carnegie Mellon* в начале 1990-х гг. Для этого микроядра разработана и продолжает использоваться микроядерная ОС общего назначения Hurd, доступная только для архитектуры x86.

Другой известный проект в области микроядерных архитектур — проект L4, не являющийся непосредственно реализованной ОС, это концепция, описывающая интерфейс взаимодействия для микроядер (модуль-серверов). Архитектура L4 является более новой, чем Mach, в ней устранены многие недостатки Mach, например такие, как низкая производительность межпроцессного взаимодействия (IPC). На основе идеологии L4 были созданы несколько ОС — L4Ka: Pistachio, L4:Fiasco.OC и другие проекты.

Основными факторами при выборе ядра в качестве основы разрабатываемой ОС были уровень проработанности системы, удобство разработки и отладки, а также численность разработчиков и их активность. Дело в том, что некоторые архитектурные тонкости отражены в документации недостаточно подробно, и при разработке ОС иногда необходимо консультироваться с представителями разработчиков. К сожалению, проекты на L4 к моменту начала разработки уже не поддерживались, сообщество разработчиков не превышало 10 человек и отсутствовала какая-либо активность в разработке системы. В то же время Mach и созданный на основе этого микроядра проект Hurd активно развивались. Это послужило причиной того, что в качестве основы для ядра был выбран проект Mach.

Аппаратная платформа

В качестве аппаратной платформы портирования ядра Mach была выбрана отладочная плата SK-AT91SAM9XE512 с процессором ARM9 Atmel AT91sam9260. Процессор ARM9 представляет собой реализацию популярной архитектуры встраиваемых процессоров, обеспечивая, таким образом, принципиальную возможность адаптации для него ОС. Область применения ядра ARM9 весьма широка, примерами могут служить контроллеры *GSM*-терминалов, средства преобразования протоколов обмена данными, наладонные компьютеры (*palm PC*), портативные измерительные устройства, карманные устройства сбора данных, устройства управления автомобильными двигателями, смарт-карты и *JPEG*-контроллеры устройств отображения.

Применяемая технология

Разработка встраиваемой ОС осуществлялась в три этапа [3].

[1] В ОС uCLinux в отличие от ОС Linux отсутствует код работы с блоком управления памятью MMU, что позволило использовать ее в системах, в которых ввиду ресурсных ограничений использовать ОС Linux нельзя.

1. Подготовительный этап. Цель состояла в разработке механизмов автоматической сборки и запуска микроядра под разные аппаратные платформы, а также в подготовке инструментов разработки, включающих компилятор, отладчик и другие средства.

2. Перенос микроядра с архитектуры i386 на архитектуру ARM9 и запуск его на аппаратной платформе.

3. Создание драйвера Ethernet с элементами TCP/IP стека в виде самостоятельного модуль-сервера, находящегося в отдельном адресном пространстве и предназначенного для обработки пакетов сети.

Для демонстрации работоспособности ОС, в качестве основы для сетевого устройства, оценивалось время обработки сетевых запросов ICMP Echo-Request, создаваемых при помощи утилиты ping.

Поскольку ОС логически разделяется на несколько частей, а именно, микроядро GNU/Mach, пространство пользователя (набор прикладных программ и стандартная библиотека libc), драйверы, средства разработки и конфигурации, то для переноса ОС необходимо переработать каждую из этих частей с учетом особенностей аппаратной платформы. При проектировании системы в нее также закладывались принципы, дающие возможность в будущем существенно расширить ее функционал:

На уровне микроядра: аппаратно зависимый исходный код ОС строился таким образом, чтобы в него можно было внести изменения без больших временных затрат, например добавить новую аппаратную платформу или архитектуру. Это требование привело к необходимости внести серьезные изменения в базовые средства конфигурирования микроядра GNU/Mach;

На уровне пользователя: исходный код драйверов устройств, пользовательских программ и библиотек разрабатывался с тем расчетом, чтобы его можно было сконфигурировать под любую аппаратную платформу. Драйверы устройств позволяют сделать это не всегда, но пользовательские библиотеки и программы, например TCP/IP стек, возможно сделать не зависящими от платформы.

Обозначенные выше принципы позволили разрабатывать ОС как портируемую с ориентацией на архитектуру ARM, что позволило вести разработку ОС как встраиваемой системы.

На первом этапе были разработаны скрипты, позволяющие автоматически произвести компиляцию системы под заданную платформу (i386 или ARM9), а также осуществить ее запуск. На этом же этапе были разработаны вспомогательные программы для тестирования функционала микроядра и проверены на платформе i386. Для эмуляции аппаратного обеспечения платформы i386 использовалась виртуальная машина QEMU, включающая в себя эмуляцию процессоров x86 и других аппаратных платформ.

На втором этапе в сконфигурированном исходном коде работающего под i386 микроядра была произведена замена аппаратно-зависимых частей кода с i386 на ARM9. Не измененные аппаратно-зависимые части ядра заменили заглушками. Был разработан код для операций сохранения, загрузки и переключения контекста и вывода диагностических сообщений, осуществлена базовая инициализация аппаратной платформы, реализованы механизмы работы с оперативной памятью. Разработка каждой из этих частей ОС велась отдельно с помощью средства разработки IAR 5.1. Использование указанной среды обусловлено обязательностью применения внутрисхемного отладчика и средств пошаговой отладки, что необходимо, например, при отладке функций переключения контекста.

Вслед за этим осуществлялись подготовка и запуск модуля-сервера, активация системного таймера и отладка интерфейса между микроядром и модуль-сервером (RPC). Была создана основа для стандартной библиотеки языка C, включающая системные вызовы и базовые функции работы с памятью (копирование, сравнение).

Также на втором этапе разработан обмен данными между модуль-сервером и микроядром.

Третий этап состоял из нескольких шагов:

- создание и отладка драйвера Ethernet в среде IAR 5.1. Драйвер производил инициализацию MAC и PHY модулей, а также осуществлял прием и отправку пакетов в сеть (MAC – Media Access Control — и PHY — Physical layer — являются аппаратными блоками, осуществляющими обмен данными между сетью и микропроцессором);
- создание анализатора пакетов сети Ethernet с минимальными возможностями. Анализатор получал пакеты из драйвера сетевого устройства, обрабатывал заголовки и формировал ответы на ARP и ICMP Echo-Request запросы (ARP, IP, ICMP являются частью стека протоколов TCP/IP [4]). Для создания ICMP Echo-Request запросов была использована утилита ping, предназначенная для проверки соединений в сетях на основе TCP/IP;
- перенос драйвера и стека в отдельный модуль-сервер.

Созданный модуль-сервер, содержащий элементы TCP/IP стека, позволил наглядно продемонстрировать работу ОС. В результате получены параллельно работающие микроядро и модуль-сервер, которые обменивались между собой сообщениями, используя механизм системных вызовов, при этом и микроядро, и модуль-сервер находились в собственных адресных пространствах.

Для тестирования созданной системы использовалась утилита ping, которая отправляет запросы (ICMP Echo-Request) протокола ICMP указанному узлу сети и фиксирует поступающие ответы (ICMP Echo-Reply). Время между отправкой запроса и получением ответа RTT (Round Trip Time) позволяет определять двусторонние задержки по маршруту и частоту потери пакетов и на их основе оценивать загруженность каналов передачи данных и промежуточных устройств. Сравнение величин этого времени, получаемых на разных платформах, обычно используется для оценки производительности TCP/IP стека и механизмов взаимодействия ОС с сетевой подсистемой.

Созданная система успешно прошла тестирование ICMP запросами. При этом не потерялось ни одно сообщение, на консоль непрерывно выводились диагностические сообщения от микроядра, а также сообщения анализа заголовков модуль-сервера. Было отправлено 50 сообщений с интервалом в 1 с.

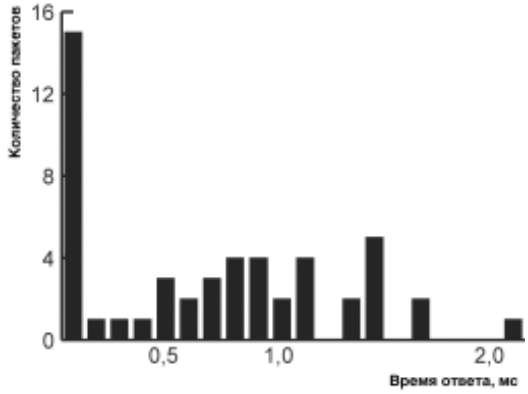

Рис. 1. Гистограмма распределения времени ответов микроядерной ОС

На рисунке 1 видно, что ответ формируется с разными интервалами времени — 15 сообщений со временем ответа в интервале 0,2–0,3 мс, и 20 сообщений в интервале 0,6–1,3 мс. При этом наблюдались и пакеты с большим временем формирования ответа — 2,2 мс. Причиной, по-видимому, является загруженность системы диагностическими сообщениями во время работы модуль-сервера.

Для сравнения подобное тестирование было проведено без использования ОС. В этом случае отсутствуют затраты времени как на переключения контекста между микроядром и модуль-сервером, так и на формирование диагностических сообщений микроядра. На рисунке 2 хорошо видны две группы сообщений — короткие со временем ответа 0,1–0,2 мс и длительные со временем ответа 0,8–0,9 мс. Отсутствие переключений

контекста и диагностики лишь очистило распределение от промежуточных значений, что говорит о проектных недостатках, присущих механизмам работы TCP/IP стека.

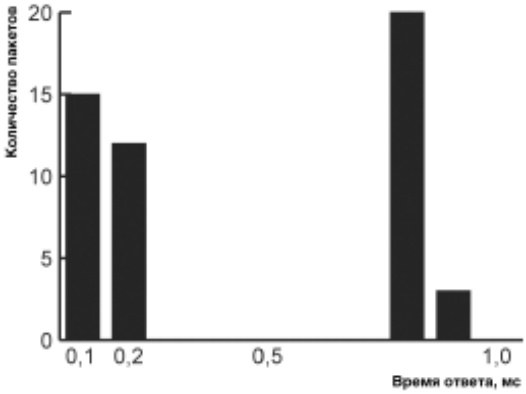

Рис. 2. Гистограмма распределения времени ответов системы без ОС

В качестве эталона можно использовать распределение времени формирования ответов для ОС Linux (рис. 3). Большая часть пакетов попадает в интервал 0,20 – 0,23 мс.

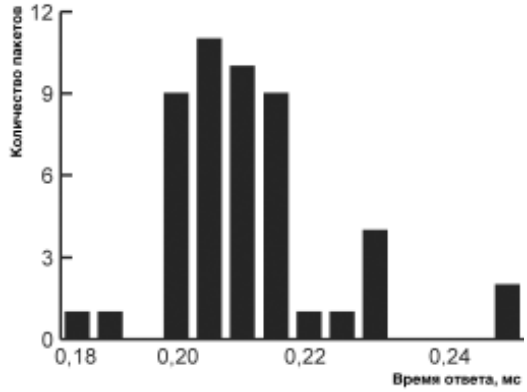

Рис. 3. Гистограмма распределения времени ответов для ОС Linux

Анализ результатов позволяет сделать вывод о полной работоспособности созданной модели микроядерной ОС, состоящей из микроядра и модуль-сервера.

Заключение

Разработанный программный комплекс, который включает параллельно работающие микроядро и модуль-сервер, представляет собой лишь модель ОС. В ней нашли отражение основные механизмы ОС, но реализованный в ней функционал не полон. В частности, в модели присутствует модуль-сервер, передающий сообщения в микроядро при помощи системных вызовов, но он только один. Более того, перенос модуль-сервера страничного обмена (пейджера) требует серьезной доработки библиотеки libc, а также части микроядра.

Важно отметить, что разработанный алгоритм и созданные в процессе разработки ОС программные объекты допускают простой перенос микроядра GNU/Mach на другие популярные платформы, например MIPS и PowerPC.

Полученный результат, таким образом, пока еще нельзя рассматривать как окончательный и представляющий собой полноценную ОС. Считая своей очередной целью создание системы, позволяющей компилировать уже существующие модули-серверы ОС Hurd, авторы планируют осуществить необходимые для достижения этой цели доработки библиотеки libc, самого микроядра и средств конфигурации.

Исходный код разработанной ОС, средства разработки и отладки размещены в сети Интернет по адресу http://www.ksyslabs.org/.

Список литературы

1. *Таненбаум Э. С.* Современные операционные системы. СПб.: Питер, 2007. — 1040 с.
2. *Олифер В. Г., Олифер Н. А.* Сетевые операционные системы. СПб.: Питер, 2005. — 669 с.
3. *Сартаков В. А.* Разработка встраиваемой операционной системы на основе микроядерной архитектуры GNU/MACH // Естественно-научные, гуманитарные и социально-экономические науки: сб. материалов 2-й заочной научн.-практ. конф. 25 ноября. 2009. Челябинск, изд-во ЮУрГУ, 2009.
4. ISO/IEC Information technology — Open Systems Interconnection — Basic Reference Model: The Basic Model 1994 ISO/IEC 7498 – 1:1994 (E).