УДК 004.451

ОПЕРАЦИОННАЯ СИСТЕМА РЕАЛЬНОГО ВРЕМЕНИ ДЛЯ ВСТРАИВАЕМЫХ УСТРОЙСТВ

Зайцев С.А., Корниенко С.К.

Целью данной работы является разработка операционной системы реального времени для встраиваемых устройств. Встраиваемым устройством называют программно-аппаратный комплекс, предназначенный для решения определённого и при этом достаточно узкого круга задач. Отсюда следуют и особенности встраиваемых систем. В отличие от настольных систем, где аппаратная часть практически стандартна и программное обеспечение не оптимизируется под аппаратную платформу, во встраиваемых системах выбирается минимально необходимое для решения конкретной задачи аппаратное обеспечение, и чем оптимальнее реализован программный код, тем меньше средств будет затрачено на аппаратное обеспечение. Потому разработка легкой, минимальной и универсальной ОС является актуальной задачей для малых и средних встраиваемых систем. Так, около трети встраиваемых решений используют либо собственные ОС, либо модифицируют существующие ОС с открытыми исходными кодами[1].

В разработанной операционной системе использован относительно новая для встраиваемых систем парадигма — событийно-ориентированное программирование (event-driver programming, “принцип Голливуда” [2]). Согласно проведенным исследованиям, этот подход даёт существенный прирост во временных затратах на разработку, в быстродействии конечного продукта и его надёжности. Так как обработчики событий могут вызываться в пределах одного процесса, они взаимоисключаемы, а это позволяет уменьшить использование примитивов синхронизации. Управление сигналами занимает центральную нишу между пакетным программированием и использованием вложенных прерываний.

Управляемость событиями приводит к усложнению планировщика ОС, и для этого был разработан планировщик, реализующий в частных случаях как кооперативную, так и вытесняющую многозадачность. Планировщик должен обеспечивать равномерное распределение процессорного времени между задачами и к тому же гарантировать своевременную обработку прерываний. В настольных ОС, как и в некоторых встраиваемых ОС, прерывания не являются вложенными, т.е. во время обработки одного прерывания остальные либо сохраняются в

очереди, если это предусмотрено аппаратным обеспечением, либо игнорируются. Как правило, аппаратная очередь прерываний вмещает не более 2-3 запросов. Ставший за долгое время классическим, подход с использованием вложенных прерываний заключается в том, что в контексте одного обработчика прерывания допускается возникновение еще нескольких прерываний, каждое в дочернем контексте [3]. Этот подход трудно реализовать и он достаточно ресурсоёмкий. Многие ОС реального времени намеренно его не используют, тем самым ухудшая время отклика (напр. FreeRTOS, TNKernel).

В разрабатываемой ОС за счет системы событий при возникновении прерывания генерируется событие, адресуемое процессу-обработчику (это занимает малую кванту времени и на время отклика практически не влияет). Сами обработчики прерываний при этом прерываемы, что не влияет на время отклика.

При этом была сохранена простота UNIX-систем — не использовались никакие дополнительные реализации задач, кроме процессов. При этом из-за утраты актуальности на встраиваемых системах была исключена поддержка файловых систем, а также управление учетными записями пользователей.

Механизм использования функций ОС сводится к системным вызовам. В некоторых архитектурах системные вызовы не реализованы. Для этих платформ использовалась эмуляция системных вызовов — непрерываемое выполнение кода в контексте ядра ОС.

В ОС реализованы базовые функции для работы с процессами (порождения и уничтожения процессов, приостанов и ожидание процессов, управления приоритетами, управления сигналами, как средством межпроцессных коммуникаций). ОС требует наличия таймера для отсчета интервалов планировщика, наличия оперативной памяти и наличия последовательного интерфейса для отладки.

В минимальной сборке ОС занимает для процессоров ARM около 3кб и около 4 кб для архитектуры AVR. Для архитектуры ARM реализована поддержка Thumb-режима, что позволяет уменьшить размер машинного кода за счёт некоторых потерь в быстродействии.

Перечень ссылок

1. Keith E. Curtis Embedded Multitasking – 2006. – NEWNES – 416 p.
2. L. Sha, R. Rajkumar, J. Lehoczky Priority Inheritance Protocols: An Approach to Real-Time Synchronization // IEEE Transactions on Computers, Vol.39, No.9, 1990.
3. David Seal ARM Architecture Reference Manual, Second Edition – 1996. – Addison-Wesley – 807 p.