УДК 004.054 DOI: 10.17587/prin.13.168-177

А. Н. Годунов, канд. физ.-мат. наук, зав. отделом, nkag@niisi.ras.ru,
И. И. Хоменков, вед. инж., nkigor@niisi.ras.ru, **В. Г. Щепков**, вед. инж., nkvs@niisi.ras.ru,
Федеральный научный центр Научно-исследовательский институт системных исследований Российской академии наук (ФНЦ НИИСИ РАН), Москва,
А. В. Хорошилов, канд. физ.-мат. наук, вед. науч. сотр., khoroshilov@ispras.ru, Институт системного программирования им. В. П. Иванникова Российской академии наук (ИСП РАН), Москва

Конфигурируемая тестовая система для ОСРВ семейства Багет

Описана тестовая система, предназначенная для проверки работоспособности операционных систем реального времени, которая была разработана и используется в НИИСИ РАН для автоматизации процесса тестирования программного обеспечения. Если большинство тестовых систем проектируются, в первую очередь, с ориентацией на автоматизацию сборки, проведение тестирования, а также сбора, обработки и визуализации полученных результатов, то в рассматриваемой системе помимо этого предусмотрены специальные возможности для упрощения отладки и исправления обнаруживаемых проблем, что позволяет сократить усилия разработчиков операционной системы реального времени на анализ результатов тестирования.

Ключевые слова: *операционная система реального времени (ОСРВ), тестирование программного обеспечения, автоматизация программирования, кросс-платформенная разработка, языки программирования C, C++, POSIX, ARINC-653*

Введение

При разработке операционных систем необходимо проводить тестирование для проверки надежности и соответствия стандартам. И как следовало ожидать, существует специализированное программное обеспечение (ПО) для автоматизации тестирования Unix-подобных операционных систем, например, Avocado [1], LAVA [2], Linux Test Project [3], Linux Distribution Checker [4], Open POSIX Test Suite [5], UnixBench [6] и т. д. Но использование таких готовых программных систем не всегда удобно, так как они либо содержат только узкоспециализированные наборы тестов, либо поддерживают только определенную аппаратуру, либо не содержат гибкой системы конфигурирования. Так или иначе, требуются большие трудозатраты на адаптацию готовой тестовой системы для тестирования нового семейства операционных систем. Универсальные системы автоматизации тестирования, такие как BuildBot [7], Jenkins [8], GitLab CI [9] и др., не учитывают особенности тестирования операционных систем, связанные с необходимостью выполнения тестируемой системы на аппаратных устройствах или их эмуляторах и также требуют существенной доработки. Поэтому в НИИСИ РАН была разработана собственная оригинальная тестовая система (ТС) для проверки работоспособности операционных систем реального времени (ОСРВ) семейства Багет [10], которые разрабатываются в НИИСИ РАН для применения во встраиваемых системах.

1. Назначение и возможности

Тестирование является неотъемлемой частью разработки ПО. Для достижения высокого качества ПО требуется, как правило, выполнение большого числа тестов. Их разработка сравнима по трудоемкости с разработкой собственно тестируемого ПО. Отметим, что тесты приходится запускать многократно в различных конфигурациях на различных устройствах, как в процессе разработки нового ПО, так и при выпуске обновлений уже существующего.

Время выполнения тестов в случае сложного ПО может быть значительным (вплоть до нескольких суток), запуск тестов (указание значений параметров и создание ПО нужной конфигурации) и анализ результатов требуют много времени и высокой квалификации тестировщиков. В силу этого возникает потребность в автоматизации процесса тестирования.

С этой целью в НИИСИ РАН в сотрудничестве с ИСП РАН была разработана ТС, которая уже несколько лет используется при разработке ОСРВ семейства Багет.

При разработке ТС ставились задачи создать удобный инструмент тестирования как для тестировщи-

ков, так и для программистов, снизить трудоемкость и сократить время тестирования. Тестировщики, с одной стороны, нуждаются в удобном средстве для запуска большого числа разнообразных тестов без трудоемкого указания всего набора параметров ОСРВ. С другой стороны, программистам нужно иметь возможность быстро найти тест, завершившийся с ошибкой, и определить условия, в которых он выполнялся. Если такая ошибка была обнаружена ранее и зарегистрирована в системе отслеживания ошибок, нужно найти ее описание, обсуждение и текущее положение дел по ее устранению.

При создании прикладного ПО для ОСРВ используется кросс-платформенная разработка, когда процесс разработки программного кода и исполнение готовых программ происходят на компьютерах с разной архитектурой. Компьютер, на котором ведется разработка, называется инструментальной ЭВМ (ИЭВМ), а компьютер, на котором выполняется ПО, — целевой ЭВМ (ЦЭВМ). После завершения цикла разработки ПО может работать на ЦЭВМ автономно. В нашем случае ИЭВМ — это компьютер с процессором Intel, который работает под управлением операционной системы Linux, а ЦЭВМ — это компьютер с микропроцессором архитектуры MIPS под управлением ОСРВ семейства Багет.

Тестовая система имеет клиент-серверную архитектуру. Сервер ТС (приложение, написанное на языке JAVA) устанавливается на ИЭВМ. В качестве клиентов используются web-браузеры, которые с помощью удобного интерфейса позволяют запускать тесты ОСРВ на разных ЦЭВМ и анализировать результаты тестирования.

Тестовая система обеспечивает автоматизацию следующих рутинных процессов тестирования ОСРВ:

- установку новых версий ОСРВ (а также удаление или переустановку старых версий);
- конфигурирование ОСРВ, т. е. настройку параметров для конкретной ЦЭВМ (сетевые адреса, имена терминалов, каталоги на сервере и т. д.);
- сборку исполняемых образов ОСРВ (с различными тестами, включенными в образ);
- загрузку на ЦЭВМ исполняемых образов ОСРВ и поочередный запуск тестов;
- сбор и хранение на сервере протоколов выполнения тестов и их комплексный анализ;
- представление результатов тестирования в удобной форме, их архивирование на сервере и визуализацию по запросу оператора.

Процесс полного тестирования ОСРВ (от запуска тестов до получения результатов) может быть длительным. Для экономии времени предусмотрены различные сценарии запуска тестов. Возможен запуск не только всех тестов сразу на всем спектре оборудования, но и запуск выбранных групп тестов на выбранных архитектурах, а также запуск одиночных тестов, что удобно в процессе разработки и отладки ОСРВ. Имеются также другие способы управления объемом тестирования, например, можно запустить выполнение только тех тестов, которые не были выполнены или были выполнены с ошибкой ранее. Такой подход к тестированию позволяет экономить много времени и ресурсов за счет автоматизации рутинных операций и более эффективного использования оборудования и рабочего времени. Тестовая система может выполнять тесты одновременно на нескольких ЦЭВМ, максимально загружая оборудование и экономя время.

Также существует возможность запускать тесты ОСРВ традиционным способом (в ручном режиме). Такой способ тестирования используется, когда нужно провести серию однотипных официальных испытаний ОСРВ. Для автоматизации этого процесса в ТС было разработано приложение на языке JAVA, которое использует интерфейс командной строки вместо графического web-интерфейса. Оно позволяет выполнять все рутинные процедуры по конфигурированию, сборке, запуску тестов и сборке протоколов тестирования. Но анализ протоколов выполняется вручную (полуавтоматический режим).

Таким образом можно сказать, что ТС является полезным и эффективным инструментом, который позволяет:

- повысить качество конечного продукта (ОСРВ) за счет возможности выполнения большого числа тестов на большой номенклатуре оборудования;
- снизить суммарное время проведения полного тестирования за счет автоматизации всех подготовительных операций, распараллеливания тестирования и возможности непрерывной круглосуточной работы;
- снизить число ошибок при тестировании, связанных с "человеческим фактором" (невнимательность, забывчивость, торопливость и т. д.), и, как следствие, повысить достоверность результатов тестирования и сократить время тестирования;
- уменьшить время на поиск ошибок за счет автоматического анализа протоколов тестирования и возможности быстро повторить тест, завершившийся с ошибкой;
- архивировать результаты тестирования на сервере, хранить их сколь угодно долго и предоставлять программистам по первому требованию со своих рабочих мест;
- использовать web-интерфейс для максимального удобства работы с ТС.

2. Используемая аппаратура

2.1. Состав стенда

На рис. 1 представлены фрагменты реального стенда, на котором происходит тестирование ОСРВ в НИИСИ РАН.

Стенд для тестирования и разработки ОСРВ состоит из одной ИЭВМ (высокопроизводительный выделенный сервер, на котором установлена ТС), совокупности ЦЭВМ разных типов и инфраструктуры, необходимой для их взаимодействия:

- внешняя сеть позволяет нескольким программистам одновременно работать удаленно с ТС и по-

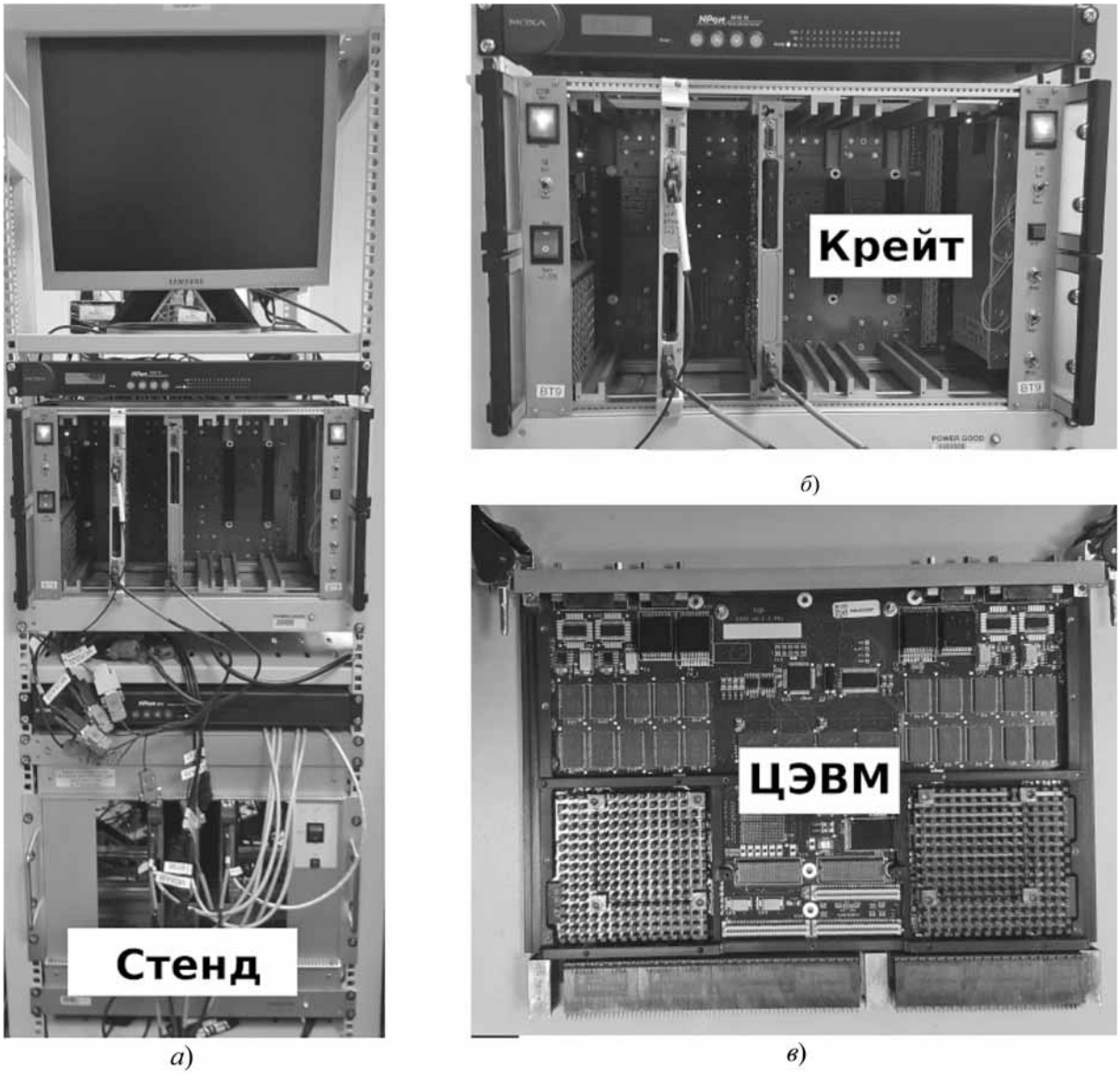

Рис. 1. Оборудование ТС:
а — фрагмент тестового стенда; *б* — крейт; *в* — ЦЭВМ

лучать результаты тестирования со своих рабочих мест;

- внутренняя сеть используется для организации взаимодействия между различными компонентами тестового стенда по сети Ethernet;
- коммутаторы последовательных портов RS-232 используются для подключения к ЦЭВМ через терминальный интерфейс, который обеспечивает реализацию интерфейса управляющей командной строки, а также получение текстовой информации с результатами выполнения тестов; взаимодействие между коммутаторами портов RS-232 и ИЭВМ осуществляется по сети Ethernet;
- генератор аппаратных сигналов RESET на перезагрузку ЦЭВМ; выдачей сигналов на перезагрузку управляет сервер ТС по сети Ethernet;
- графические мониторы, графические манипуляторы и клавиатуры используются в качестве графических терминалов ЦЭВМ, когда разрабатываются приложения для ОСРВ, которые требуют интерактивного взаимодействия с человеком, например, рабочее место оператора.

В состав стенда в настоящее время входят несколько десятков ЦЭВМ с различными типами процессоров, разработанных в НИИСИ РАН, таких как КОМДИВ32, КОМДИВ64, КОМДИВ64-SMP, КОМДИВ64-RIO, КОМДИВ128-RIO [11]. Эти ЦЭВМ (модули) вставляют в крейты (корпус с общей шиной) по несколько штук. Крейты, в свою очередь, монтируются в индустриальные стойки с необходимой вспомогательной инфраструктурой. Стенд для тестирования и разработки ОСРВ состоит из нескольких стоек с оборудованием, которые в значительной степени не зависят друг от друга. Такой подход позволяет легко масштабировать стенд для подключения новых типов ЦЭВМ путем добавления новых стоек или новых крейтов.

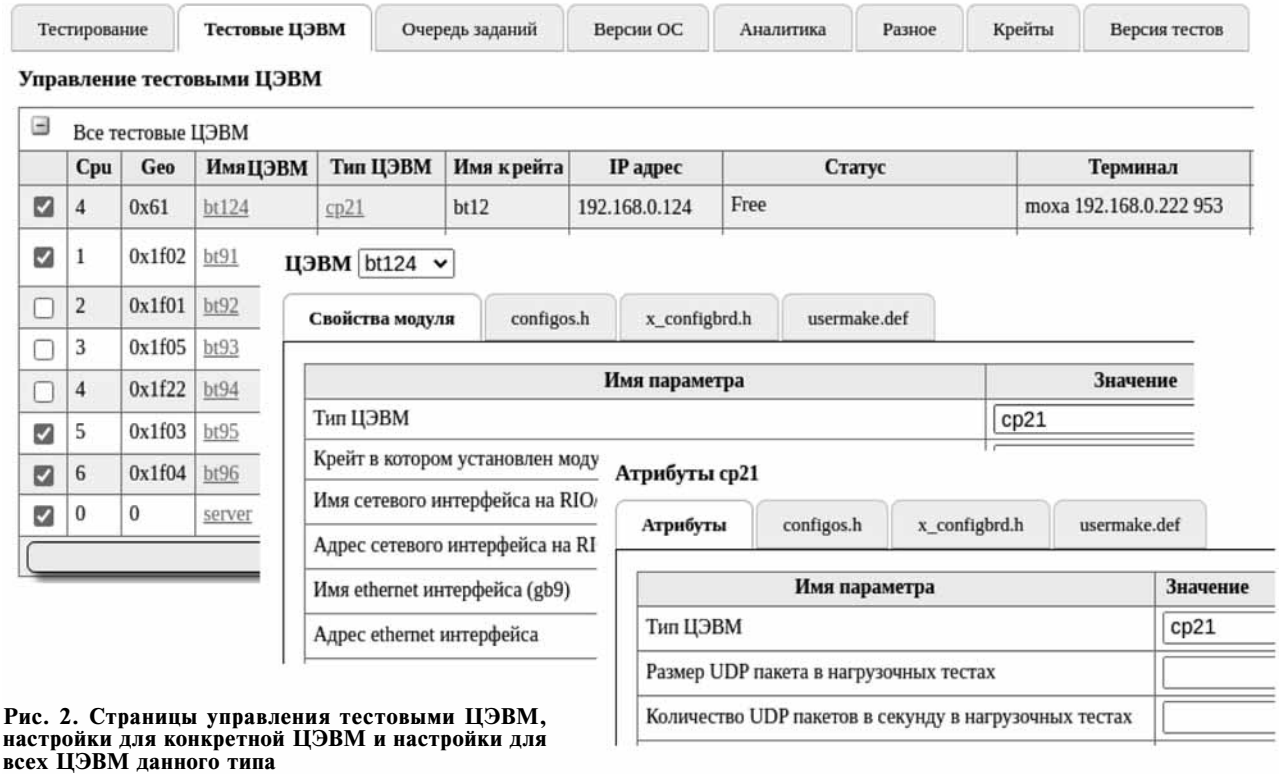

Рис. 2. Страницы управления тестовыми ЦЭВМ, настройки для конкретной ЦЭВМ и настройки для всех ЦЭВМ данного типа

2.2. Конфигурирование ЦЭВМ

Программный сервер ТС работает на ИЭВМ и взаимодействует со всем спектром ЦЭВМ. Для этого он использует формальное описание всех имеющихся на стенде ЦЭВМ. Страница "Управление тестовыми ЦЭВМ" сервера содержит сводную таблицу всех ЦЭВМ, зарегистрированных в ТС (фрагмент таблицы приведен на рис. 2).

Каждая строка в сводной таблице относится к одной ЦЭВМ (или к одному процессору в многопроцессорных модулях), а в полях этой строки перечислены наиболее важные параметры, которые используются ТС для взаимодействия с этой конкретной ЦЭВМ. В этой таблице можно добавлять новые ЦЭВМ (нажав кнопку внизу таблицы) или редактировать настройки уже существующих. Настройки ЦЭВМ делятся на две группы: настройки для конкретной ЦЭВМ и настройки для всех ЦЭВМ данного типа.

При нажатии на ссылку в столбце "Имя" сводной таблицы происходит переход на страницу, где указываются атрибуты (параметры) только для одной конкретной ЦЭВМ (средняя таблица на рис. 2).

Перейдя по ссылке в столбце "Тип ЦЭВМ" сводной таблицы, можно попасть на страницу атрибутов этого типа ЦЭВМ (нижняя таблица на рис. 2). Эти настройки применяются для всех ЦЭВМ такого типа.

3. Программное обеспечение

3.1. Установка и удаление версий ОСРВ в ТС

Дистрибутив ОСРВ содержит базовый набор программ, сценариев и библиотек, необходимых для построения исполняемых образов. Для установки ОСРВ на компьютер используется сценарий, написанный на языке командного интерпретатора Linux. Установленная таким образом ОСРВ содержит все необходимое для поддержки конкретной микропроцессорной архитектуры, но не содержит драйверов периферийных устройств, которые могут различаться для разных типов ЦЭВМ. Поэтому, кроме установки дистрибутива ОСРВ нужно установить один или несколько пакетов поддержки модулей (ППМ), которые содержат драйверы для конкретных типов ЦЭВМ.

Тестовая система имеет доступ к каталогам, куда разработчики помещают дистрибутивы версий ОСРВ, и периодически проверяет содержимое этих каталогов. Страница ТС "Версии ОС" позволяет устанавливать и удалять версии ОСРВ и ППМ с помощью следующих вкладок:

- **Установленные версии** — таблица со списком уже установленных в ТС версий ОСРВ, а также версий ППМ к ним;
- **Неустановленные версии** — таблица версий ОСРВ, которые были созданы, но еще не были установлены в ТС;
- **Архивные версии** — дополнительный раздел, который аналогичен разделу "Установленные версии"; здесь перечислены версии ОСРВ, разработка которых была завершена.

После успешной установки информация о версии перемещается из раздела "Неустановленные версии" в раздел "Установленные версии". Для того чтобы версия ОСРВ попала список архивных версий, нужно присвоить ей соответствующий статус.

При установке ОСРВ создаются один или несколько каталогов, предназначенных для работы программиста, которые называют целевыми. Они содержат все необходимые компоненты для построения исполняемого образа ОСРВ для конкретного типа ЦЭВМ (по одному каталогу на каждый тип ЦЭВМ). В целевом каталоге размещаются исходные тексты программ на языках С или С++. Построение исполняемого образа начинается с запуска конфигуратора ОСРВ. Конфигуратор позволяет включить тексты программ в создаваемый исполняемый образ ОСРВ, а также задать конфигурационные параметры ОСРВ (размер памяти, сетевые адреса и т. д.). По завершению конфигурирования приводится компиляция и сборка исполняемого образа ОСРВ в виде файла, который можно загрузить и выполнить на ЦЭВМ.

3.2. Структура тестов ОСРВ

Все тесты ОСРВ сгруппированы в подкаталоги по принципу проверки какой-нибудь одной подсистемы ОСРВ. Тестовым набором авторы называют все тесты одного такого подкаталога. Каждый тестовый набор состоит из отдельных тестов, которые оформлены в виде файлов на языке С (С++). Обычно все тесты тестового набора помещаются в один исполняемый образ ОСРВ и выполняются последовательно, хотя есть и исключения из этого правила. Иногда случаются ситуации, когда ошибка, возникшая при выполнении одного теста, может повлиять на выполнение последующих тестов (например, ошибочное затирание памяти). Если необходимо исключить возможное влияние одного теста на другой, то можно построить исполняемый образ ОСРВ только с одним тестом из тестового набора и выполнить только его (такой режим тестирования требует существенно больше времени).

Перечислим наиболее важные тестовые наборы:

- тесты для проверки ОСРВ на соответствие стандарту ARINC-653;
- тесты для проверки ОСРВ на соответствие стандарту POSIX;
- тесты для проверки математических функций (sin, cos и т. д.) на точность вычисления результата с учетом различных режимов округления чисел с плавающей точкой;
- тесты для проверки ОСРВ (семейства ОС4000) на соответствие стандарту языка С++;
- тесты производительности, измеряющие скорость выполнения различных функций ОСРВ (при этом не проверяется корректность выполнения этих функций).

Последний пункт требует дополнительных пояснений. Тесты производительности (*benchmarks*) не являются корректными тестами, но позволяют оценивать временные характеристики ОСРВ и сравнивать между собой быстродействие различных микропроцессорных модулей и различных версий ОСРВ (это можно сделать на странице "Аналитика", описанной ниже). Дополнительно они являются средством обнаружения "скрытых" проблем в ПО. Например, если тест вычисления какой-либо математической функции дает верный результат (что проверяется математическими тестами), но время ее выполнения сильно увеличивается по сравнению с предыдущими версиями ОСРВ (что проверяют тесты производительности), можно предположить, что в ходе выполнения этой функции происходят исключительные ситуации (*exceptions*). Эти исключительные ситуации корректно обрабатываются соответствующими обработчиками, но эти обработчики требуют дополнительного процессорного времени, что может быть критично в системах реального времени. Тесты производительности могут также являться средством профилирования ПО (сбором статистики о времени выполнения какой-либо функции и о числе вызовов этой функции) и помогать оптимизировать программный код.

3.3. Управление тестовыми наборами

Перед использованием тестов соответствующие тестовые наборы должны быть предварительно зарегистрированы в ТС. Для этих целей служит страница "Управление тестами", представленная на рис. 3.

Эта страница содержит таблицу, где перечислены все тестовые наборы. Она также позволяет управлять ими: добавлять, удалять, редактировать и т. д. При нажатии на ссылку в столбце "Имя тестового набора" происходит переход на страницу конфигурирования для данного тестового набора. Там указывается, на каких целевых платформах и с какими параметрами этот тестовый набор может запускаться. Кроме того, на странице содержится полный список тестов в этом наборе. Это позволяет сравнить список реально выполненных тестов с эталонным списком и обнаружить невыполненные или пропущенные по каким-то причинам тесты.

3.4. Построение исполняемых образов ОСРВ

Важным этапом при построении исполняемого образа ОСРВ является процесс конфигурирования. Здесь задаются конкретные параметры и структура будущего образа, например, сетевые адреса, каталоги с файлами, пользовательские программы и т. д. Все эти настройки сохраняются в виде конфигурационных файлов.

Для автоматизации процесса конфигурирования в ТС заранее заготовлены фрагменты конфигурационных файлов, относящиеся к различным режимам работы ОСРВ, ЦЭВМ и тестовых наборов. Эти фрагменты конфигурационных файлов хранятся в отдельном каталоге ТС, который имеет сложную систему подкаталогов и названий файлов, позволяющую быстро получить доступ к нужным фрагментам конфигурации ОСРВ.

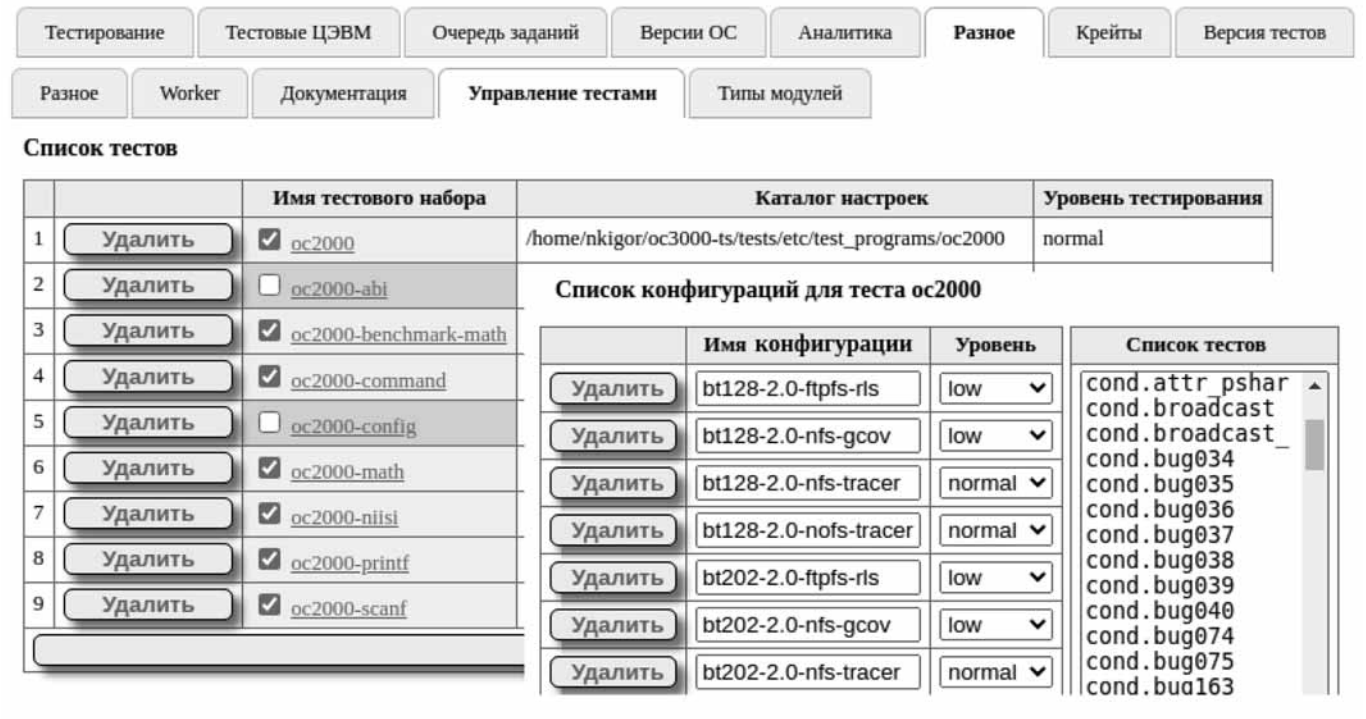

Рис. 3. Страницы управления и конфигурирования тестов

Для того чтобы создать конкретную конфигурацию для тестирования ОСРВ, программист (при первоначальной настройке ТС) задает ее имя следующим образом: имя конфигурации должно состоять из цепочки ключевых слов, соединенных между собой дефисами. Ключевые слова в имени — это названия отдельных файлов с фрагментами конфигураций, которые находятся в подкаталогах, упомянутых выше. Эти файлы будут добавляться к конфигурационным файлам ОСРВ в процессе автоматического конфигурирования.

Имя конфигурации можно увидеть в первом столбце таблицы тестирования (рис. 4, см. вторую сторону обложки). Например, на рис. 4 открыта страница тестирования в ТС, выбраны версия ОСРВ oc3000-3.52.215 и тестовый набор oc3000-math в конфигурации bt206-3.0-nfs-dbg. Для определенности укажем процессорный модуль bt61 в качестве ЦЭВМ. Запустить процесс тестирования ОСРВ в данной конфигурации можно, нажав кнопку "Выполнить тесты" в сводной таблице тестирования (рис. 4, см. вторую сторону обложки) в строке, которая начинается с названия нашей конфигурации и относится к выбранному тестовому набору. При этом ТС будет знать, что нужно использовать следующую конфигурационную информацию для построения исполняемого образа ОСРВ:

- версия ОСРВ — oc3000-3.52.215 (название каталога с установленной ОСРВ);
- тестовый набор — oc3000-math (название каталога с тестовым набором для тестирования математических функций);
- микропроцессорная архитектура ЦЭВМ — bt206 (название целевого каталога для ЦЭВМ типа bt206 с микропроцессором RM7000);
- семейство ОСРВ — 3.0;
- тип файловой системы — nfs (подключить сетевую файловую систему для чтения файлов эталонных данных с ИЭВМ);
- версия библиотек ОСРВ — dbg (использовать отладочный набор библиотек и ключей компилятора);
- конкретная ЦЭВМ — bt61 (использовать сетевой адрес 192.168.0.61 и т. д.).

Конфигурирование и сборка исполняемого образа ОСРВ проводятся следующим образом. Сначала отдельно копируется целевой каталог ОСРВ, затем формируются конфигурационные файлы ОСРВ путем копирования фрагментов конфигурации из разных файлов. После окончания конфигурирования будет выполнена процедура сборки исполняемого образа ОСРВ (компиляция всех исходных текстов проекта и создание исполняемого образа системы).

4. Тестирование

4.1. Процесс тестирования ОСРВ

Настройка и конфигурирование самой ТС проводятся после ее установки на сервер, а также по мере необходимости (например, в случае подключения нового оборудования к стенду).

Использование ТС предполагает выполнение следующих действий. На сервер устанавливаются

версия ОСРВ для тестирования и набор соответствующих ППМ для разных модулей. Затем проводятся конфигурирование и сборка исполняемых образов ОСРВ. На следующем этапе происходят загрузка исполняемых образов ОСРВ на ЦЭВМ и запуск тестов. В заключение ТС проводит анализ протоколов исполнения тестов и представление результатов в удобной для просмотра форме. Дальше тестировщик должен сам проанализировать полученные ТС данные и сделать выводы о качестве ПО и его дальнейшей судьбе.

Результаты тестирования конкретной версии ОСРВ хранятся на сервере и доступны в любое время, пока пользователь не удалит их явно.

В ходе выполнения тесты выводят на консоль протокол тестирования, который начинается с названия теста и заканчивается окончательным вердиктом:

- PASS — тест прошел успешно;
- FAIL — была обнаружена ошибка или возникли какие-то трудности при выполнении теста;
- UNTESTED — тест не был запущен в ходе тестирования в связи с невозможностью его выполнения на данной программно-аппаратной платформе.

Вердикт UNTESTED не является ошибкой тестирования. Он является следствием использования единого набора тестов для всего набора оборудования и всех конфигураций ОСРВ в целях упрощения поддержки ТС в актуальном состоянии и повышения ее надежности. Поэтому некоторые тесты запускаются только в том случае, когда для этого существуют определенные условия, например, наличие соответствующего оборудования.

При тестировании ОСРВ окончательный вердикт является совокупностью вердиктов выполнения отдельных тестовых наборов в различных режимах и конфигурациях.

4.2. Запуск тестов ОСРВ

Запустить тесты можно на странице "Тестирование". Здесь все тесты сгруппированы в виде сводной таблицы. Вся таблица может быть очень длинной и поэтому имеет два представления — развернутый и компактный (рис. 4, см. вторую сторону обложки).

Перед запуском тестов выбирается семейство ОСРВ Багет: ос2000, ос3000, ос4000 или ос4000.64 (64-разрядная версия ОСРВ ОС4000). Затем выбирается номер конкретной версии ОСРВ (например, ос3000-3.52.215). Если строчки этой таблицы содержат только названия тестовых наборов и окрашены в белый цвет, то тестирование этой версии ОСРВ еще не проводилось. Если некоторые строчки таблицы окрашены в какой-то цвет и содержат какую-либо информацию, то соответствующие тесты уже были запущены и можно посмотреть отчет о результатах тестирования.

Таблица содержит управляющие кнопки, которые позволяют запускать тесты в нужном порядке и в нужном объеме. Можно запустить на выполнение:

- все тесты на всех целевых платформах (будет заполнена вся тестовая таблица);
- все тесты на выбранной целевой платформе (будет заполнен раздел тестовой таблицы);
- группу тестов на выбранной целевой платформе в различных конфигурациях ОСРВ (будут заполнены несколько строк тестовой таблицы);
- группу тестов на выбранной целевой платформе в выбранной конфигурации ОСРВ (будет заполнена одна строка тестовой таблицы);
- только один тест ОСРВ на выбранной целевой платформе в выбранной конфигурации ОСРВ.

Для перехода в режим выполнения одиночного теста нужно нажать на ссылку в первой колонке таблицы. После этого осуществится переход на страницу выбора, где данный тестовый набор представлен в виде управляющих кнопок с названиями одиночных тестов или кнопками с названием групп тестов из данного тестового набора (рис. 5, см. вторую сторону обложки). Теперь можно запустить только один тест или небольшую группу выбранных тестов. Это удобно для поиска ошибок при отладке.

Отметим, что в ТС тесты запускаются не сразу. Вначале они попадают в очередь заданий на тестирование. Очередь формируется и движется автоматически по мере запуска тестов и освобождения оборудования для тестирования. Это позволяет запускать большое число тестов на длительный срок, например, на ночь, когда тестовый стенд свободен. На странице "Очередь заданий" можно видеть текущее состояние очереди (какие тесты сейчас выполняются и на каком оборудовании). Здесь можно остановить выполнение всех тестов и очистить очередь заданий (нажав кнопку "Остановить все тесты"), но нельзя управлять самой очередью, изменяя число и очередность заданий. Добавить или удалить отдельные тестовые наборы из очереди можно на странице "Тестирование", нажав соответствующие кнопки. Алгоритм работы очереди заданий на тестирование не предусматривает никаких приоритетов в выполнении тестов (очередь движется линейно), но при этом может запускать параллельное выполнение одного задания сразу на нескольких модулях одновременно. Например, если запустили один тестовый набор с математическими тестами и есть шесть свободных, подходящих по типу процессорных модулей, то ТС запустит сразу шесть тестов из этого тестового набора (один процессор будет выполнять тестирование функции sin, а другой — функции cos и т. д.). По мере окончания тестирования очередной математической функции каждый из шести процессорных модулей будет получать новую, таким образом сильно ускоряется процесс тестирования. В логику работы очереди заданий также включено слежение за временем выполнения тестов. Если тест тратит больше отведенного ему времени, то его выполнение прекращается, тест считается неудачным, а ТС переходит к выполнению следующего задания в очереди. Таким образом решается проблема "зависания" тестов.

По завершению тестирования (возможно, через несколько часов) можно получить все результаты тестов. Это таблица (рис. 4, см. вторую сторону обложки), где строки содержат названия тестовых наборов и результаты тестирования. Строчки итоговой таблицы раскрашены в разные цвета. Цвет позволяет быстро оценить результаты тестирования. Если строчка таблицы окрашена в зеленый цвет, то результаты всех тестов соответствующей группы положительные (в идеале вся таблица должна быть зеленого цвета), а если строчка красная, то результат хотя бы одного теста отрицательный. Возможен также фиолетовый цвет, когда тесты ОСРВ закончились с ошибками, но все эти ошибки уже известны, зарегистрированы и находятся в стадии устранения. Желтый цвет означает, что все результаты тестов этой группы были положительные (не было отрицательных), но число положительных результатов не совпадает с общим ожидаемым числом тестов этой группы. Другими словами, какие-то тесты не выполнились (нет названия теста в протоколе тестирования, нет окончательного вердикта за отведенное время, поэтому ТС не может автоматически квалифицировать ситуацию) и программисту следует разобраться с этим тестом самостоятельно. Такое возможно, например, если при старте теста возникла исключительная ситуация (*exception*), которая не может быть корректно обработана, и операционная система останавливает выполнение этого теста, т. е. получен отрицательный результат для этого теста, но косвенным образом, как отсутствие какой-либо информации о нем. Розовый цвет говорит, что ошибки возникли еще до запуска тестов на этапе компиляции или тестирование ОСРВ прервалось по неизвестной причине. Например, какой-то крейт был выключен до завершения тестов. Напомним, что белый цвет строк в таблице говорит о том, что запуск соответствующей группы тестов еще не проводился.

Надписи в этих строках, также как и цвет сообщают о результатах тестирования, например, "Нет проблем" означает, что тесты прошли успешно, а "Ошибок *N*" означает, что в ходе тестирования было обнаружено *N* ошибок.

В таблицу тестирования внесены лишь самые краткие сведения о результатах выполнения тестов: сколько тестов было запущено, сколько выполнено и сколько было ошибок. Чтобы получить подробные протоколы о результатах выполнения группы тестов нужно нажать на ссылку в строке таблицы с количеством ошибок (например, "Ошибок: 1") — будет выведена страница с подробным отчетом, которая построена на базе анализа результатов, полученных с ЦЭВМ. Эта страница является результатом обработки всех протоколов выполнения данного тестового набора. Она содержит структурированную информацию о запуске тестов и результатах их выполнения. В начале страницы расположена информация о конфигурации образа ОСРВ, времени начала и окончания тестирования, краткий список тестов с результатами выполнения (тесты с ошибками выделяются красным цветом). Затем идет подробный отчет по выполнению каждого теста в отдельности. Это дает возможность получать детальную информацию об ошибках максимально быстро без утомительного просматривания множества протоколов. Если какая-то ошибка была обнаружена раннее, то можно найти ссылку на запись в базе данных отслеживания ошибок (*Mantis Bug Tracker*) и узнать текущее состояние дел по ее устранению.

4.3. Сравнение результатов

Вкладка "Аналитика" позволяет сравнивать результаты тестирования версий ОСРВ по некоторому выбранному критерию.

Сравнивать можно время выполнения тестов на разных версиях ОСРВ. Это позволяет находить ошибки в ОСРВ если, например, производительность операционной системы сильно ухудшается. Также можно сравнивать время выполнения тестов одной и той же версии ОСРВ, но на разных микропроцессорных модулях. Это дает возможность понять, насколько один микропроцессорный модуль производительнее другого, а также находить ошибки в ОСРВ, если время выполнения меняется непропорционально на разных микропроцессорных архитектурах.

Для сравнения версий реализовано несколько алгоритмов, которые представлены в следующих закладках:

- **Сравнение типов** — в этом разделе можно получить таблицу, где сравнивается время выполнения некоторых специально выбранных функций одной и той же версии ОСРВ, но для разных микропроцессорных модулей;
- **Математические функции** — этот раздел аналогичен предыдущему, но для сравнения скорости выполнения функций одной и той же версии ОСРВ на различных микропроцессорных архитектурах выбираются только математические функции стандарта POSIX;
- **ARINC-функции** — этот раздел аналогичен предыдущим, но для сравнения скорости выполнения функций одной и той же версии ОСРВ на различных микропроцессорных архитектурах выбираются только функции стандарта ARINC;
- **Производительность по версиям** — здесь можно сравнить, насколько одна версия ОСРВ быстрее или медленнее, чем другая на одной и той же микропроцессорной архитектуре;
- **Сравнение версий** — этот раздел отличатся от предыдущих тем, что сравниваются не временные характеристики разных версий ОСРВ, а то число ошибок, которое было выявлено при тестировании; таким образом можно отслеживать, как исправляются обнаруженные в ОСРВ ошибки и увеличивается стабильность версий ОСРВ в ходе их разработки.

Результатом сравнения является текстовый файл с расширением .csv (*comma-separated values*), который можно импортировать в программу электронных таблиц, а дальше обработать информацию в нужном виде с помощью встроенных функций (например, построить графики).

4.4. Дополнительные возможности

В разделе "Разное" есть список зарегистрированных проблем в ОСРВ. База данных ошибок формируется на основании регистрации сообщений об ошибках (*ticket*) в системе регистрации и отслеживания ошибок Mantis Bug Tracker [12]. Разработчики ОСРВ регистрируют сообщения об ошибках ОСРВ в системе Mantis в ручном режиме. Для этого нужно сначала создать сообщение об ошибке (*ticket*), а система Mantis присвоит этому сообщению уникальный номер. В этом сообщении нужно описать проблему, присвоить ей статус и назначить ответственного за ее решение. В дальнейшем статус этой проблемы может меняться (в процессе работ по устранению ошибки): новая, назначенная, решенная, закрытая и т. д. Система Mantis не является частью ТС, но предоставляет возможность узнать удаленно (по сети) текущий статус сообщения об ошибке по его уникальному номеру. Эту особенность использует ТС для автоматического слежения за ошибками, возникающими в ходе тестирования ОСРВ. Для этого ТС ведет свою собственную базу данных ошибок на своем сервере и предоставляет тестировщику возможность связать конкретную ошибку, выявленную при тестировании ОСРВ, и номер зарегистрированного сообщения об этой ошибке в Mantis, нажав соответствующую кнопку при просмотре отчетов о тестировании. Впоследствии это дает возможность ТС вставлять ссылки на зарегистрированные в Mantis сообщения, относящиеся к данной проблеме, и менять вид сообщения об ошибках, базируясь на статусе сообщения об ошибке в системе Mantis. Это очень удобно, так как можно понять текущее состояние этой проблемы и что делается для ее разрешения на текущий момент. Отметим дополнительно, что если при тестировании ОСРВ были обнаружены ошибки и все эти ошибки были зарегистрированы в системе Mantis, то в сводной таблице тестирования строка с этой группой тестов закрашивается не в красный цвет, а в фиолетовый — это значит, что ошибки есть, все они известны, зарегистрированы в Mantis и ждут своего исправления.

Заключение

В работе представлена конфигурируемая ТС, которая решает следующие основные задачи:

- управление распределенной и разнородной инфраструктурой тестового стенда;
- распределение процессорных модулей целевых ЭВМ тестового стенда между задачами тестирования с учетом необходимости обеспечивать оперативную передачу этих модулей в пользование разработчиков ОСРВ;
- автоматизацию рутинных операций, связанных с подготовкой тестов к выполнению, загрузкой образов ОСРВ в процессорные модули целевых ЭВМ, сбором журналов тестирования;
- предоставление удобного интерфейса для настройки системы, постановки задач тестирования, мониторинга состояния тестового стенда, отладки и исправления обнаруживаемых проблем.

Многолетний опыт использования ТС для ОСРВ в НИИСИ РАН показал эффективность ее применения для повышения качества программных продуктов, сокращения временных затрат, ускорения процесса разработки новых версий ПО, а также упрощения процесса отладки, нахождения и исправления ошибок разработчиками ПО.

Публикация выполнена в рамках государственного задания по проведению фундаментальных исследований по теме «Исследование и реализация программной платформы для перспективных многоядерных процессоров» (FNEF-2022-002).

Список литературы

1. **Система** автоматизации тестирования Avacado Framework. URL: https://avocado-framework.github.io/

2. **Система** управления тестированием операционных систем LAVA. URL: https://www.lavasoftware.org/

3. **Система** тестирования Linux Test Project. URL: http://linux-test-project.github.io/

4. **Герлиц Е. А., Кулямин В. В., Максимов А. В., Петренко А. К., Хорошилов А. В., Цыварев А. В.** Тестирование операционных систем // Труды Института системного программирования РАН. 2014. Т. 26, № 1. С. 73—108.

5. **Тестовый** набор Open POSIX Test Suite. URL: http://posixtest.sourceforge.net/

6. **Система** тестирования UnixBench. URL: https://github.com/kdlucas/byte-unixbench

7. **Система** непрерывной интеграции программного обеспечения BuildBot. URL: https://buildbot.net/

8. **Leszko R.** Continuous Delivery with Docker and Jenkins. Delivering software at scale. Packt Publishing, 2017. 332 p.

9. **van Baarsen J.** GitLab Cookbook, Packt Publishing, 2014. 172 p.

10. **Годунов А. Н., Солдатов В. А.** Операционные системы семейства Багет (сходства, отличия и перспективы) // Программирование. 2014. № 5. С. 68—76.

11. **Аряшев С. И., Зубковский П. С.** 64-разрядные системы на кристалле с архитектурой "КОМДИВ" // Наноиндустрия. 2019. № 89. С. 78—79.

12. **Система** управления сообщениями об ошибках Mantis. URL: https://www.mantisbt.org/

Configurable Test System for RTOS of BAGET Family

A. N. Godunov, nkag@niisi.ras.ru, **I. I. Khomenkov**, nkigor@niisi.ras.ru,
V. G. Shchepkov, nkvs@niisi.ras.ru, Federal State Institution "Scientific Research Institute for System Analysis of the Russian Academy of Sciences" (SRISA RAS), Moscow, 117218, Russian Federation,
A. V. Khoroshilov, khoroshilov@ispras.ru, Ivannikov Institute for System Programming of the Russian Academy of Sciences, Moscow, 109004, Russian Federation

Corresponding author:

Shchepkov Vladimir G., Leading Engineer, Federal State Institution "Scientific Research Institute for System Analysis of the Russian Academy of Sciences" (SRISA RAS), Moscow, 117218, Russian Federation
E-mail: nkvs@niisi.ras.ru

Received on January 20, 2022
Accepted on February 24, 2022

The article describes a test system designed for verification of the real-time operating system BAGET Family for embedded systems, developed and used at the Scientific Research Institute for System Analysis of the Russian Academy of Sciences (SRISA RAS). This Unix-like operating system is based on the POSIX and ARINC-653 programming standards. Of course, there is specialized software for automating testing of Unix-like operating systems, for example, Avocado, LAVA, Linux Test Project, Linux Distribution Checker, Open POSIX Test Suite, UnixBench, etc. But the use of such ready-made software systems is not always convenient since they either contain only highly specialized test suites, support only specific hardware, or do not contain a flexible configuration system. Therefore, In SRISA RAS, its own original test system was developed. The task was to create a convenient testing tool for both software testers and programmers. Many years of experience in utilizing the test system has shown the effectiveness of its use to improve the quality of software products, reduction of time spent on testing and analysis of results, maximum automation of software testing process, speed up the process of developing new software versions, as well as simplify the process of debugging, finding and fixing errors by software developers.

Keywords: *real-time operating system, RTOS, software testing, automation of program developing, cross-platform software developing, program language C, C++, POSIX, ARINC-653*

For citation:

Godunov A. N., **Khomenkov I. I.**, **Shchepkov V. G.**, **Khoroshilov A. V.** Configurable Test System for RTOS of BAGET Family, *Programmnaya Ingeneria*, 2022, vol. 13, no. 4. 168—177.

DOI: 10.17587/prin.13.168-177

References

1. **Automated** Testing Framework — Avocado, available at: https://avocado-framework.github.io/

2. **Linaro** Automated Validation Architecture — LAVA, available at: https://www.lavasoftware.org/

3. **Linux** Test Project, available at: http://linux-test-project.github.io/

4. **Gerlits E. A., Kuliamin V. V., Maksimov A. V., Petrenko A. K., Khoroshilov A. V., Tsyvarev A. V.** Testing of Operating Systems, *Proceedings of the Institute for System Programming of the RAS (Proceedings of ISP RAS),* 2014, vol. 26, no. 1, pp. 73—108 (in Russian).

5. **Open** POSIX Test Suite, available at: http://posixtest.sourceforge.net/

6. **UNIX** benchmark suite — UnixBench, available at: https://github.com/kdlucas/byte-unixbench

7. **Automated** Build, Test, and Release Framework — BuildBot, available at: https://buildbot.net/

8. **Leszko R.** *Continuous Delivery with Docker and Jenkins. Delivering software at scale*, Packt Publishing, 2017, 332 p.

9. **van Baarsen J.** *GitLab Cookbook,* Packt Publishing, 2014, 172 p.

10. **Godunov A. N., Soldatov V. A.** OSRT BAGET Family, *Programming,* 2014, no. 5, pp. 68—76 (in Russian).

11. **Aryashev S. I., Zubkovsky P. S.** 64-bit SOC with architecture "COMDIV", *Nanoindustry,* Moscow, 2019, vol. 89, pp. 78—79 (in Russian).

12. **Mantis** Bug Tracker, available at: https://www.mantisbt.org/

Рисунки к статье

А. Н. Годунова, И. И Хоменкова, В. Г. Щепкова, А. В. Хорошилова

«КОНФИГУРИРУЕМАЯ ТЕСТОВАЯ СИСТЕМА ДЛЯ ОСРВ СЕМЕЙСТВА БАГЕТ»

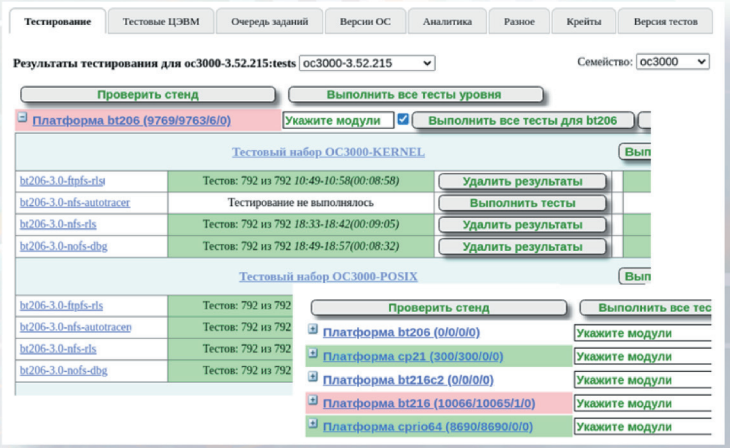

Рис. 4. Главная страница ТС в развернутом и компактном видах

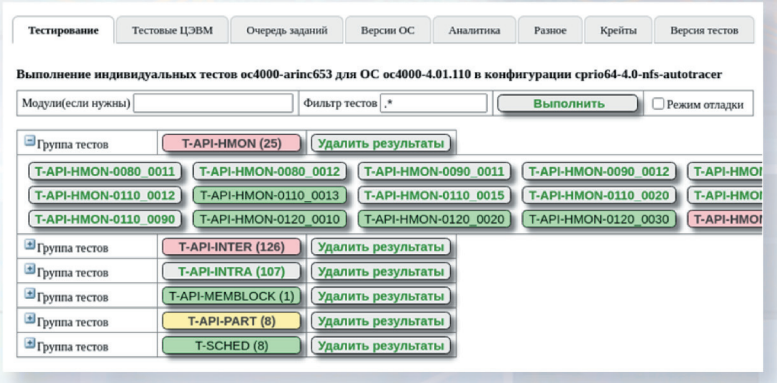

Рис. 5. Страница выбора одиночного теста или группы тестов из тестового набора