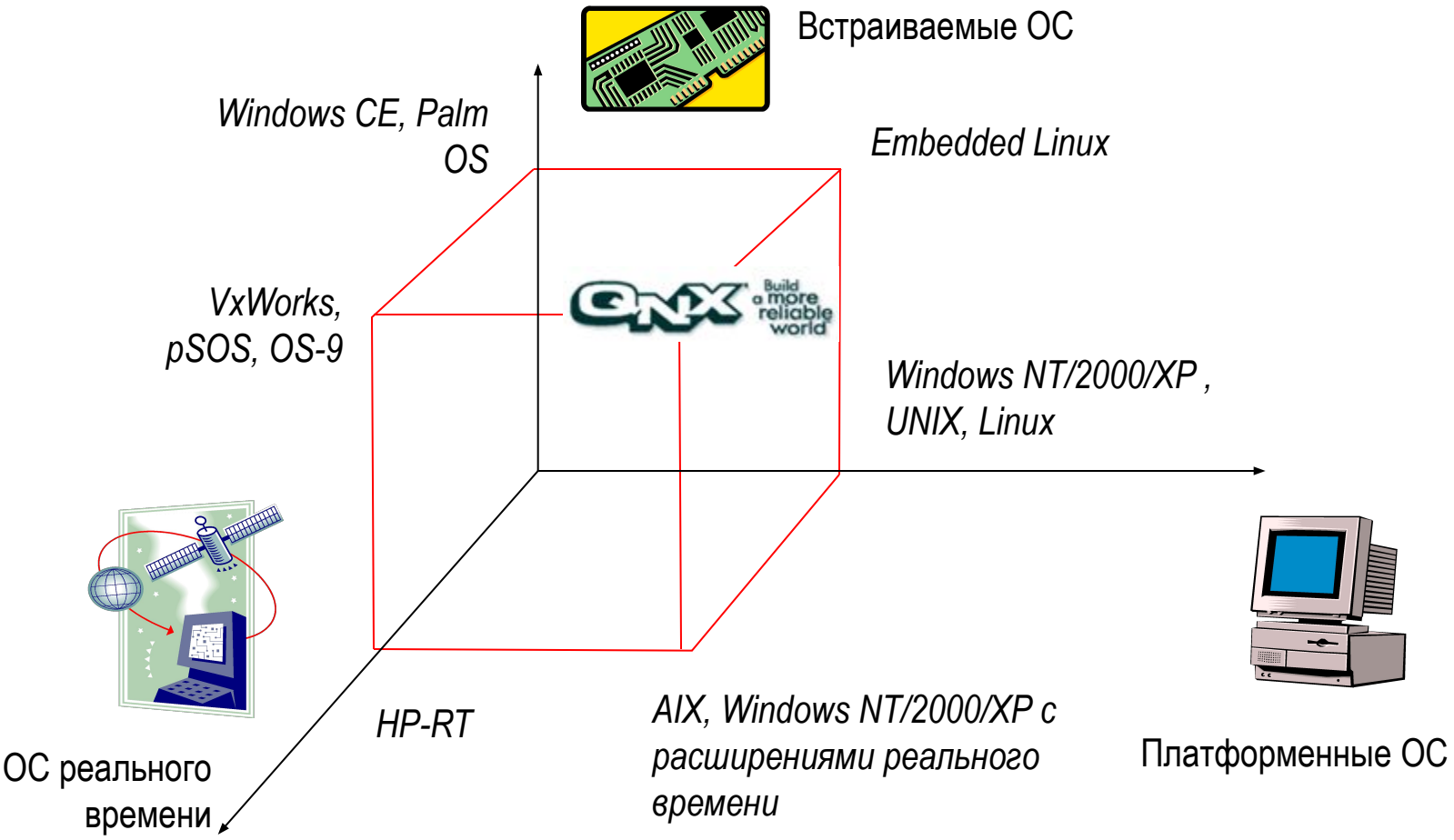
ОС реального времени QNX и интегрированный комплект разработчика QNX Momentics

Александр Трофимов
SWD Software Ltd.

OCPB QNX

Чем QNX отличается от других ОС?

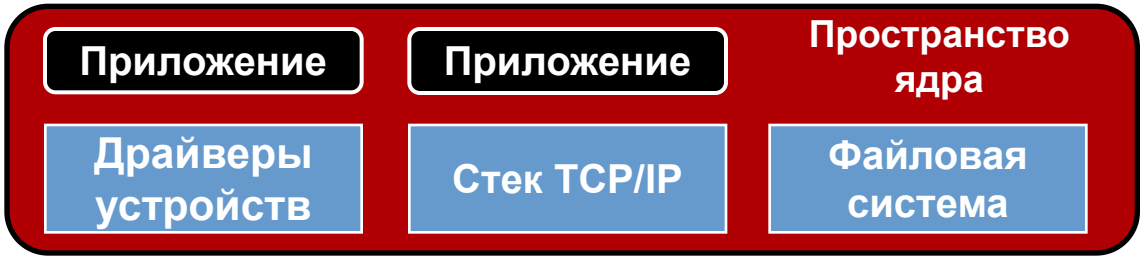


QNX как ОС жесткого реального времени

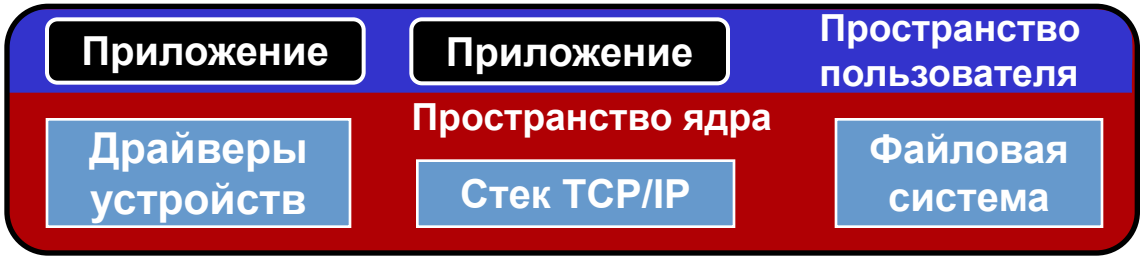
Параметр \ CPU	Pentium II 233 МГц	Pentium II 350 МГц	PowerPC MTX604 300 МГц
Время реакции на прерывание, мкс	2.08	1.20	0.96
Время постановки потока на выполнение, мкс	5.46	4.18	4.65
Время отработки вызова ядра (*sched_yield()*), мкс	1.62	1.30	1.85
Переключение контекста между потоками одного процесса, мкс	2.33	1.87	1.9

Микроядерная архитектура

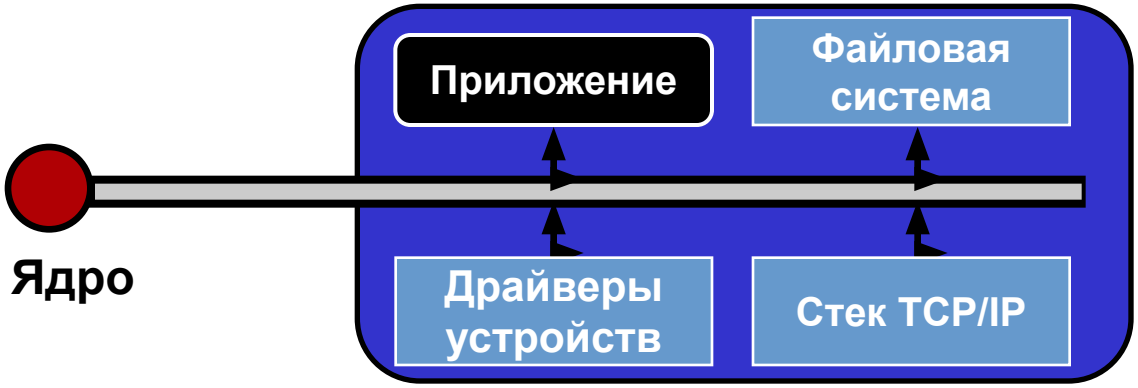
Исполнямый модуль реального времени (напр. VxWorks)



Монолитное ядро (напр. Linux)



Микроядро (напр. QNX Neutrino)



Микроядерная архитектура

Исполнямый модуль реального времени

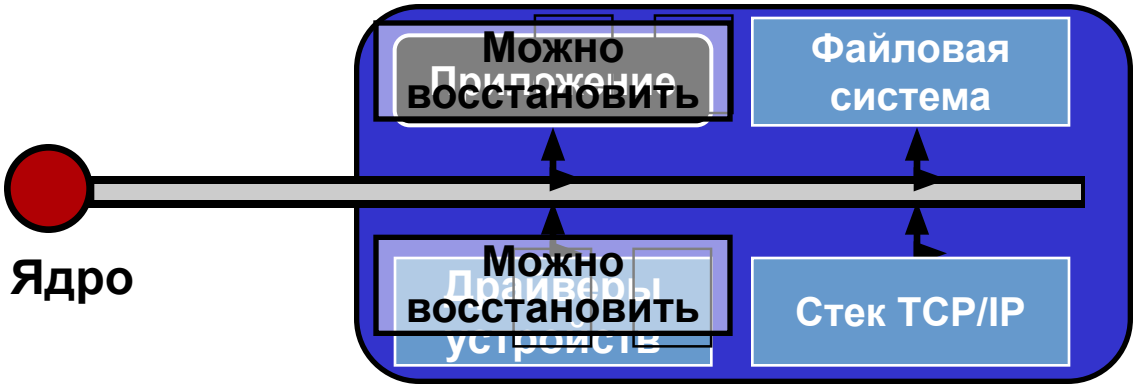
- Защиты памяти нет
- Приложения, драйверы и протоколы "живут" в пространстве ядра

Монолитное ядро (NT / Unix / и т.п.)

- MMU, частичная защита
- Защищены только приложения

Микроядро (QNX Neutrino)

- MMU, полная защита
- Защищены приложения, драйверы и протоколы



Межзадачное взаимодействие (IPC)

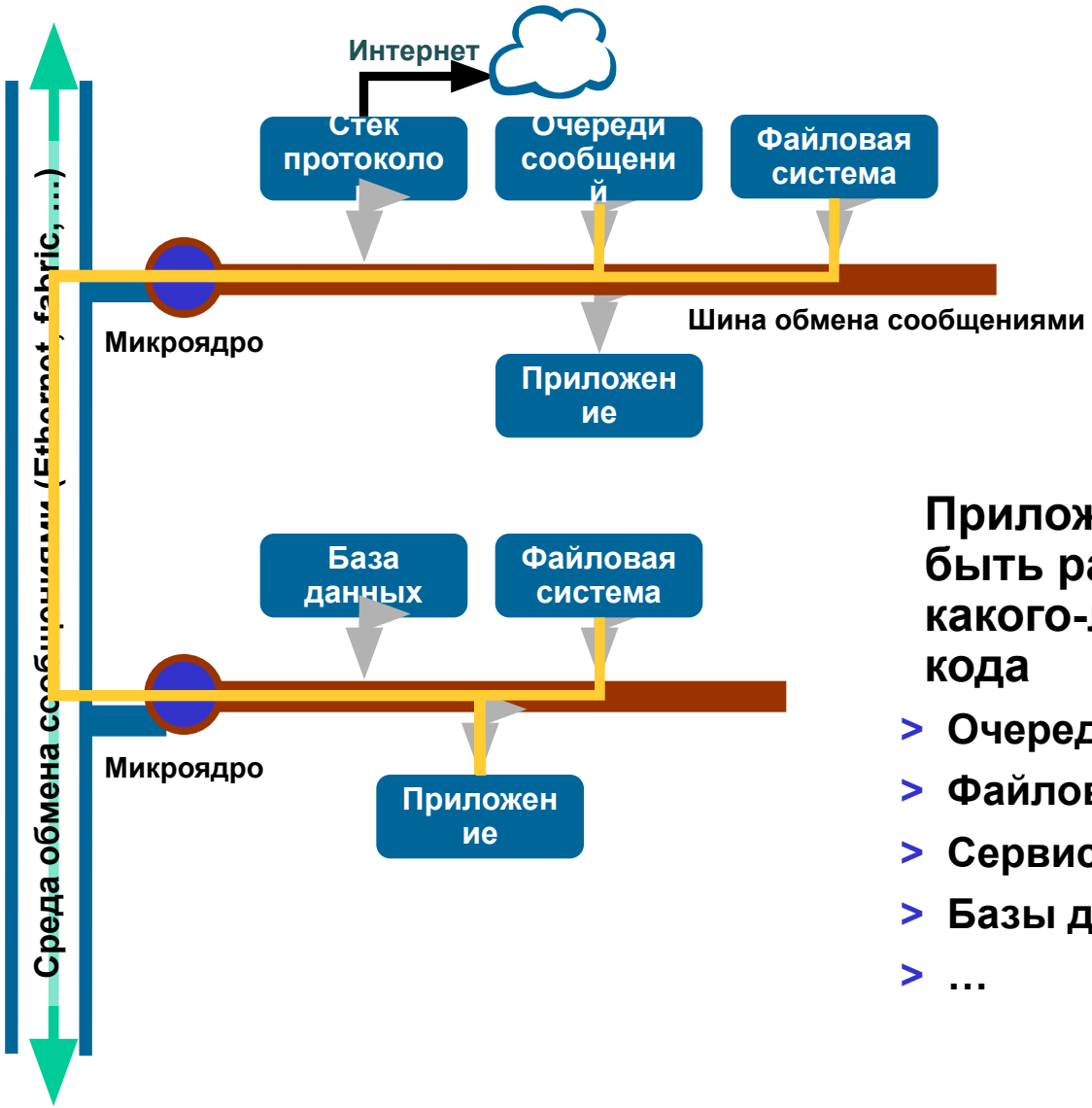
- **Задачи общаются посредством сообщений**

- **Использование сообщений органично "развязывает" задачи**
- **Над сообщениями надстроены вызовы POSIX**

  fd = open(“/dev/tcpip”, ,,,)
  read, write, stat, devctl, …
  close

- **… и другие вызовы POSIX**
  - realtime signals
  - pipes and POSIX mqueues
  - mutexs, condvars, semaphores
  - barriers, sleepon
  - reader/writer locks

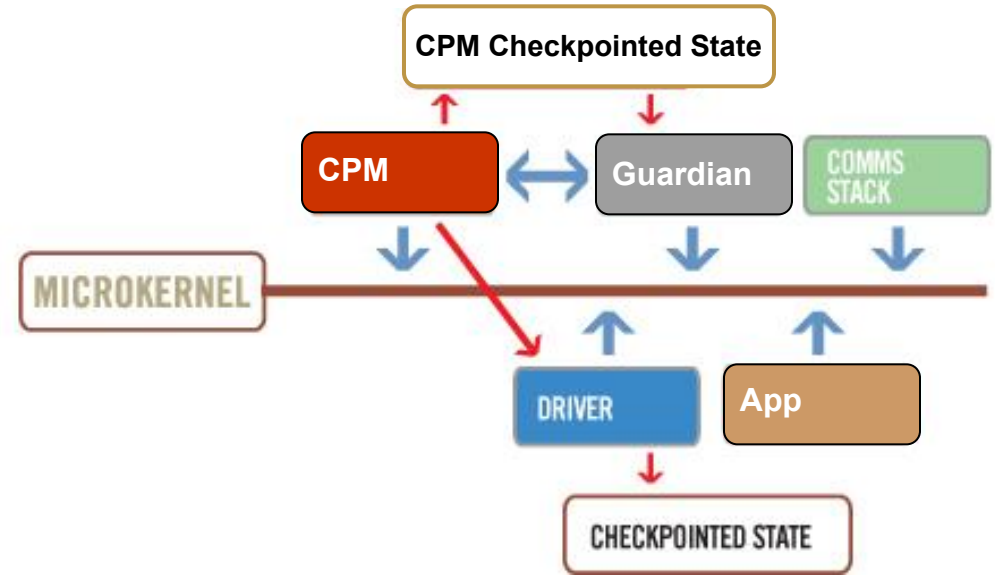
Прозрачные распределенные вычисления



Приложения/серверы могут быть распределенными без какого-либо специального кода

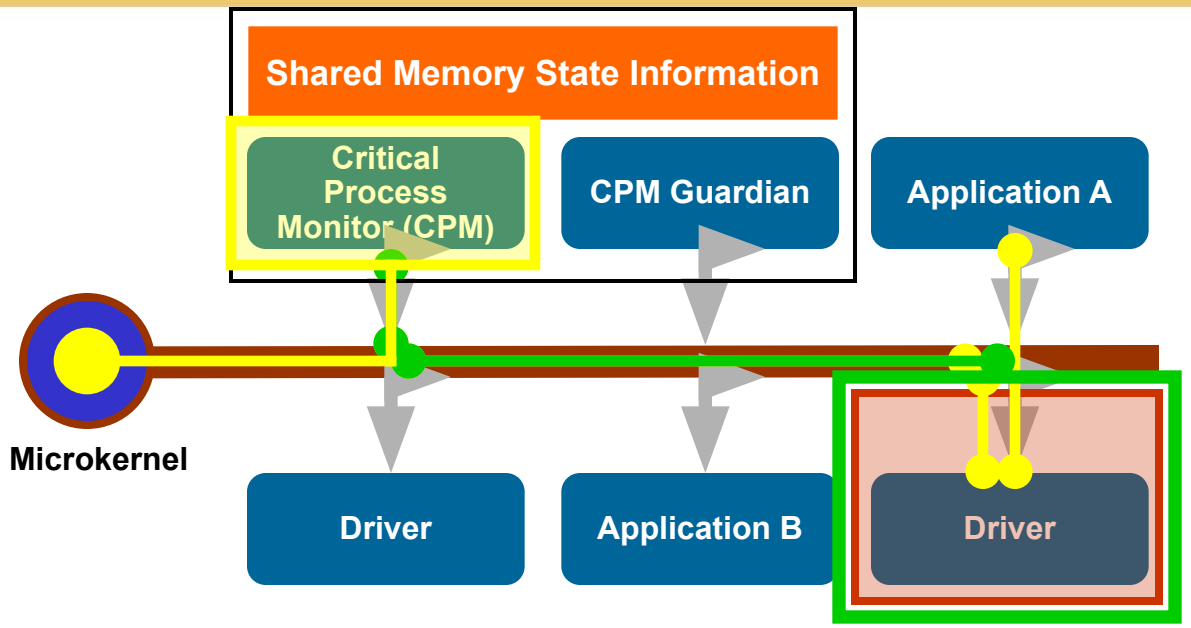
- **Очереди сообщений**
- **Файловые системы**
- **Сервисы**
- **Базы данных**
- **…**

CPM: восстановление после сбоев



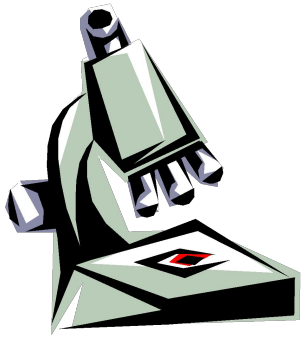
- **Основа системы высокой готовности – Монитор Ключевых Процессов (CPM). Выполняет мониторинг выбранных компонентов и обеспечивает восстановление после сбоев**
- **Процесс-охранник дублирует CPM**
- **Клиентская библиотека позволяет компонентам незамедлительно и прозрачно восстанавливать все соединения**
- **При обнаружении факта сбоя выполняется группа правил, определяющая способ восстановления**
  - **освободить ресурсы**
  - **перезапустить процесс**
  - **…и т.п.**

Critical Process Monitor



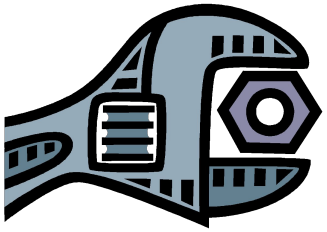
1. Сбой драйвера из-за некорректного обращения с памятью
2. Ядро уведомляет CPM об ошибке
3. Сохраняется отладочная информация о процессе (стандартный core файл)
4. Драйвер выгружается и возвращает ресурсы; уничтожается IPC канал
5. CPM перезапускает новый драйвер
6. Каналы IPC драйвера восстанавливаются клиентской библиотекой CPM
7. Драйвер запрашивает информацию у CPM о своем состоянии в последней контрольной точке и сервис восстанавливается

QNX как встраиваемая и масштабируемая ОС



- **Компактность и неприхотливость**
  - **умещается в 2Мб ОЗУ и 2Мб флэш-памяти**
  - **не требует мощного процессора**

- **Модульная структура**
  - **гибко масштабируется**
  - **сохраняет ключевые свойства даже в минимальных конфигурациях**



- **Простота адаптации к оборудованию**
  - **изящная открытая архитектура драйверов**
  - **множество примеров в исходных текстах**

QNX как платформенная ОС: поддержка POSIX

- **POSIX.1 (IEEE 1003.1) – базовый API операционных систем**
- **POSIX.1a (IEEE 1003.1a) – некоторые расширения API**
- **POSIX.2 (IEEE 1003.2) – набор утилит и командных интерпретаторов**
- **POSIX.4 (IEEE 1003.1b) – расширения для поддержки реального времени**
- **POSIX.4a (IEEE 1003.1c) – интерфейсы потоков, выполняющихся внутри POSIX-процессов**
- **POSIX.1b (IEEE 1003.1d), IEEE 1003.1j – дополнительные расширения реального времени**
- **POSIX.12 (IEEE 1003.1g) – независимый от протокола интерфейс сокетов**

QNX как платформенная ОС: «а что под нее есть?»

QNX и целевая аппаратура

- **Целевые процессоры**
  - **QNX поддерживает x86, PowerPC, MIPS, SH-4, ARM/StrongARM/Xscale и их производные**
- **Пакеты поддержки процессорных плат (BSP)**
  - **BSP в исходных текстах для QNX Momentics**
  - **BSP от производителей оборудования**
- **Стартовые комплекты**
  - **содержат все необходимое, чтобы сразу приступить к делу**

Adaptive Partitioning

Что такое Адаптивная Декомпозиция?

- Новый продукт QNX, расширяющий ОСРВ QNX Neutrino
- Позволяет разработчикам создавать безопасные разделы или «партиции» из множества приложения или потоков
- Разделам гарантируется определенная часть ресурсов CPU на основании простых в использовании бюджетов

Почему Адаптивная?

- Запатентованная технология распределит все ресурсы CPU по разделам исходя из их потребностей – ресурсы CPU используются максимально эффективно
- Обеспечивает максимальную производительность благодаря рациональному использованию ресурсов процессора, особенно в пиковых ситуациях

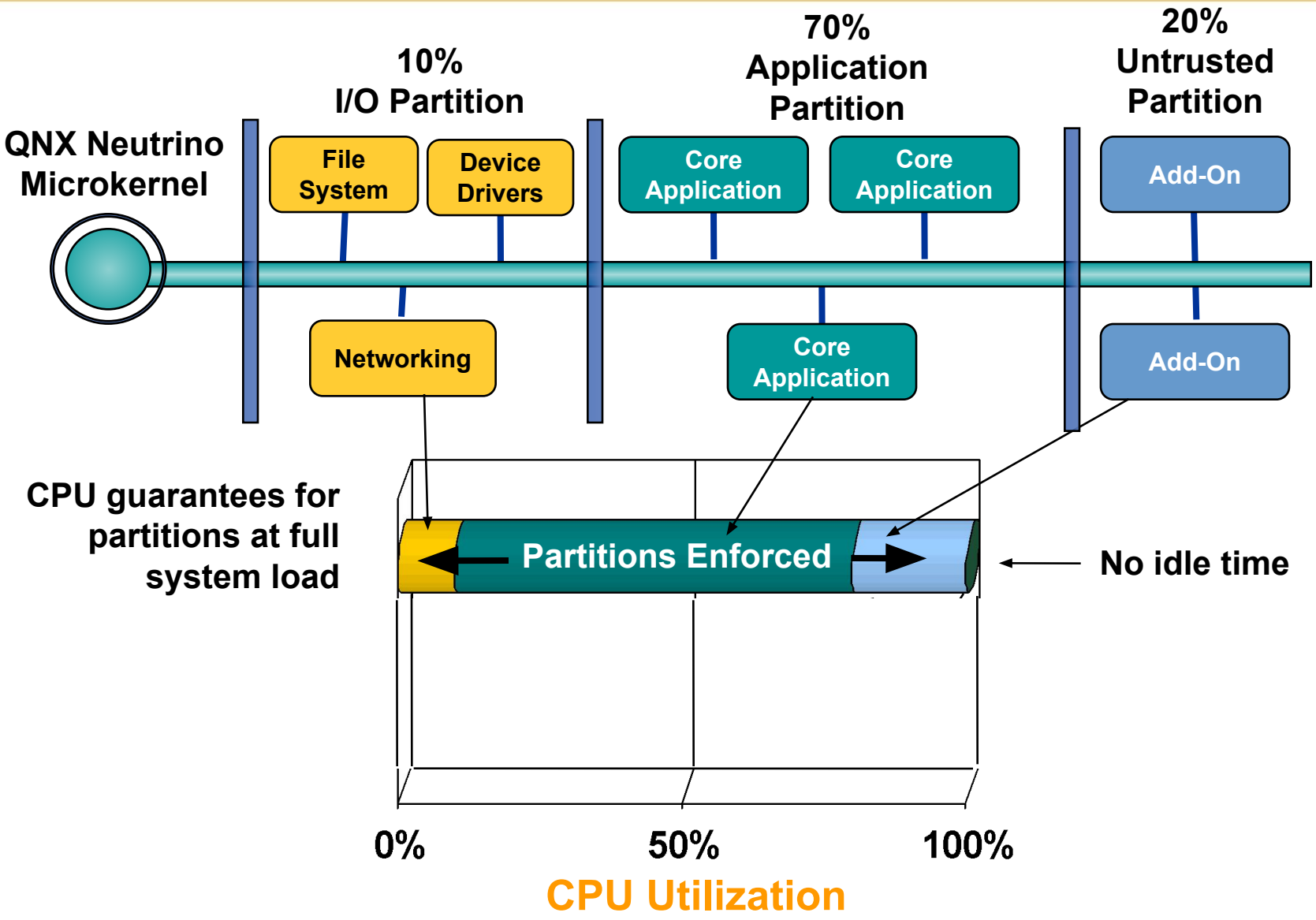
Простота использования

- ***Не требуются изменения*** при проектировании
  - Та ж модель программирования POSIX, знакомый дизайн, техники программирования и отладки
- ***Нет требуются изменения в коде*** для использования адаптивной декомпозиции

Зачем нужна Adaptive Partitioning?

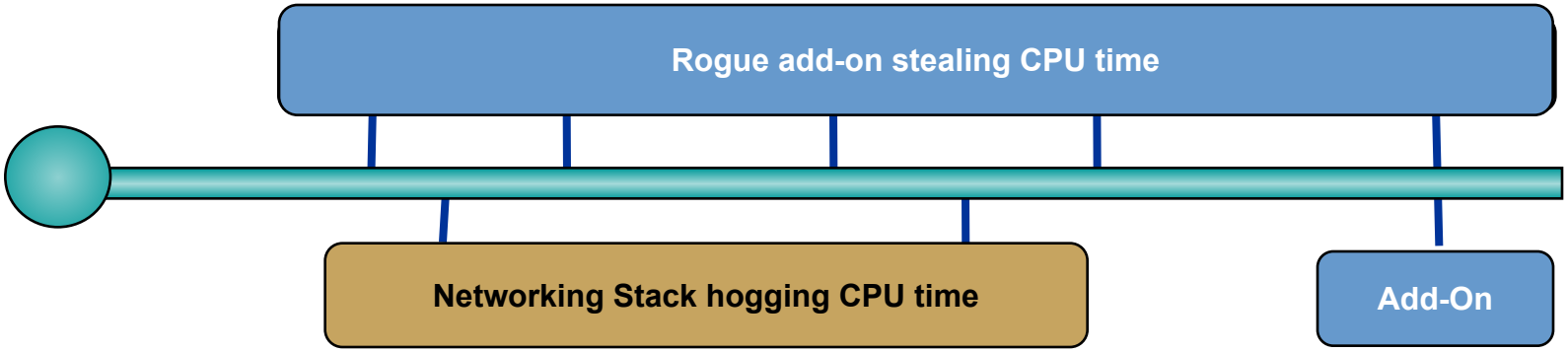
- **Основанное на приоритетах вытесняющее планирование может гарантировать выполнение приоритетных задач в системах реального времени**
  - **Как только задача готова к выполнению, она сразу же получает ресурсы процессора**
  - **С усложнением системы, множество задач борются за ресурсы CPU и становится сложно масштабировать схему приоритетов задач**
- **Планирование на основе приоритетов не гарантирует то, что задача будет выполнена в случае, если готова к выполнению более приоритетная задача**
  - **Без гарантий времени CPU, низкоприоритетные задачи будут находится в состоянии дефицита процессорного времени**
  - **Это может привести к деградации и даже к общему сбою системы**
- **Адаптивная декомпозиция гарантирует минимальное время CPU партициям для обеспечения корректного функционирования системы в периоды сильной загрузки CPU**

Максимизация производительности



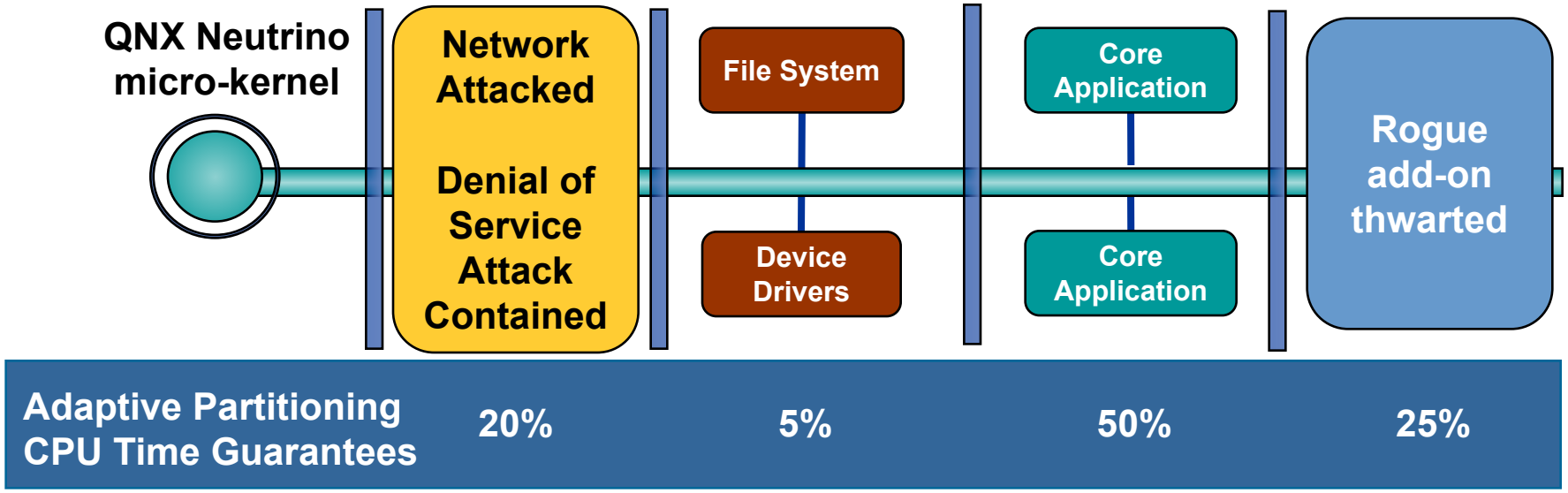
Безопасность

- **Почти все встраиваемые устройства подключены к сети**
  - Ненадежные сетевые интерфейсы и угрозы
  - Недоверенное add-on программное обеспечение
- **Если превентивные меры не включены в проект, доступность и безопасность устройств может быть скомпрометирована**
  - Возможны атаки отказа в обслуживании (DOS), что отнимет у приложений ресурсы процессора
  - Нет необходимости проверять недоверенные приложения на предмет возможных атак
- **Распределенная DOS атака может загрузить систему обработкой сетевых событий**



Безопасность

- **Создайте разделы для защиты критичных системных ресурсов**
  - Гарантия ресурсов CPU для базовых функций
  - Наследование партиций гарантирует время CPU всем сервисам ОС (драйвера, файловые системы, сетевая система)
- **Защита основные приложений от угроз**
  - Минимизация влияние вредоносного ПО
  - Защита от DOS атак

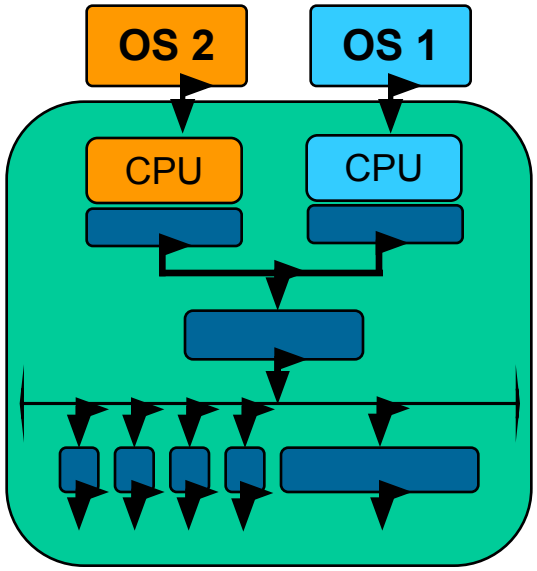


QNX Neutrino Multi-Core TDK

- **QNX Neutrino Multi-core Technology Development Kit единственная в отрасли полнофункциональная платформа для нового поколения multi-core кристаллов**
- **При помощи QNX Neutrino Multi-Core Technology Development Kit вы сможете:**
  - **Быстро перенести приложения для однопроцессорной архитектуры на любую многопроцессорную архитектуру, значительно сократив при этом время выхода на рынок ваших изделий**
  - **Быстро разработать надежные, высокопроизводительные продукты для последнего поколения multi-core процессоров**
  - **Сразу же создавать проекты с возможностью их дальнейшего расширения с dual-core на multi-core кристаллы**

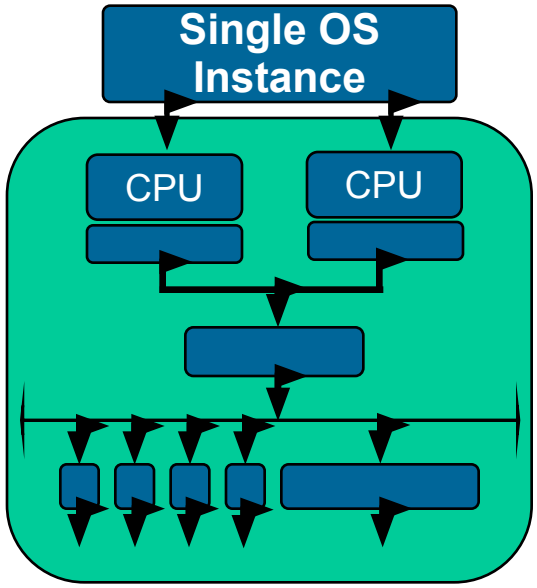
Многопроцессорная модель

Asymmetric



- 2 ядра, 2 ОС
- Одна и та же или разные ОС
- QNX, Linux, VxWorks, OSE, Integrity

Symmetric



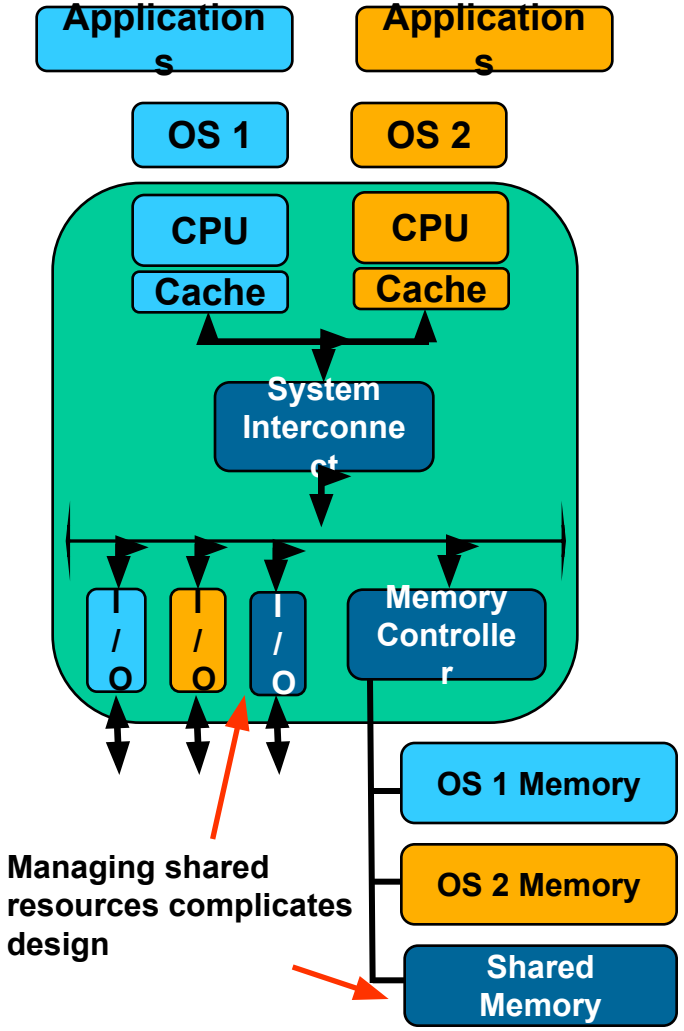
- 2 ядра, одна ОС
- QNX, Linux

Asymmetric Model – «ЗА»:

- Единственный возможный вариант запускать различные ОС
- CPU может быть назначен на обработку какой-либо задачи
- Единственный вариант для задач, которые нельзя распараллелить

Asymmetric Model – «ПРОТИВ»:

- Разработчикам необходимо реализовывать распределение и арбитраж ресурсов
- Никакая из ОС не управляет всеми ресурсами – память, ввод/вывод, прерывания общие
- Синхронизация между ядрами реализуется сообщениями программного уровня – влияние на производительность
- Добавление новых ядер может потребовать существенного изменения проекта

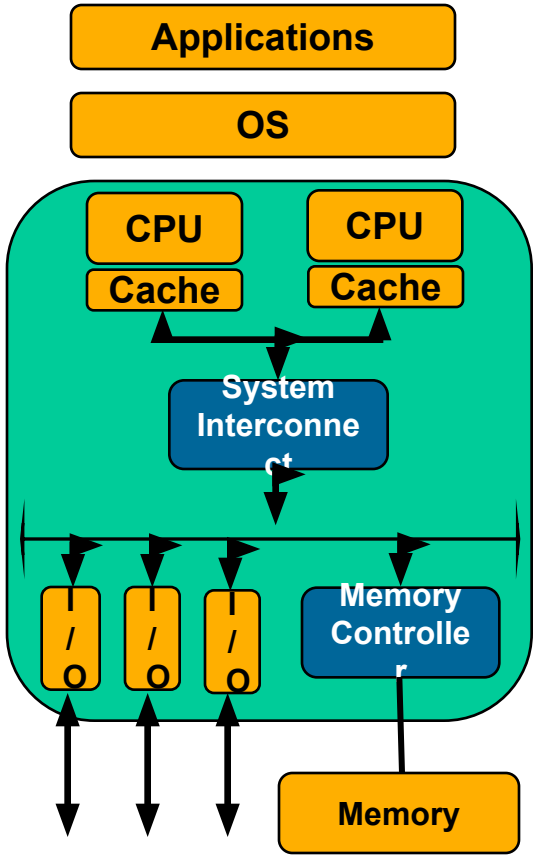


Symmetric Model – «ЗА»:

- Хорошо масштабируется. Безшовная поддержка многоядерности *<u>без модификации кода</u>*
- Одна ОС владеет и управляет всеми ресурсами, их совместным использованием и арбитражем
- Динамическая балансировка контролируется механизмом планированием потоков ОС
- Высокая производительность взаимодействия ядер и потоков с использованием примитивов POSIX
- Сбор статистики и информации на уровне всей системы с последующей оптимизацией производительности, отладкой и т.д.

Symmetric Model – «ПРОТИВ»:

- Невозможность специально выделить определенный процессор задаче из-за динамической балансировки
- Приложения с плохой синхронизацией потоков могут некорректно работать в многопроцессорной системе



Лучшее из обоих моделей

- ОС работает в симметричном режиме с возможностью «привязать» приложения к конкретному ядру
- Одна ОС имеет полный контроль
  - Ресурсы распределятся ОС, что облегчает проектирование
  - Сбора информации и статистики на уровне всей системы для оптимизации производительности и отладки
  - Высокая производительности
    - Синхронизация приложения между ядрами с использованием примитивов POSIX
    - Высокая скорость обмена сообщениями сообщениями в пределах одного ядра
- Легко расширяется для варианта с более чем двумя ядрами

Лучшее из обоих моделей

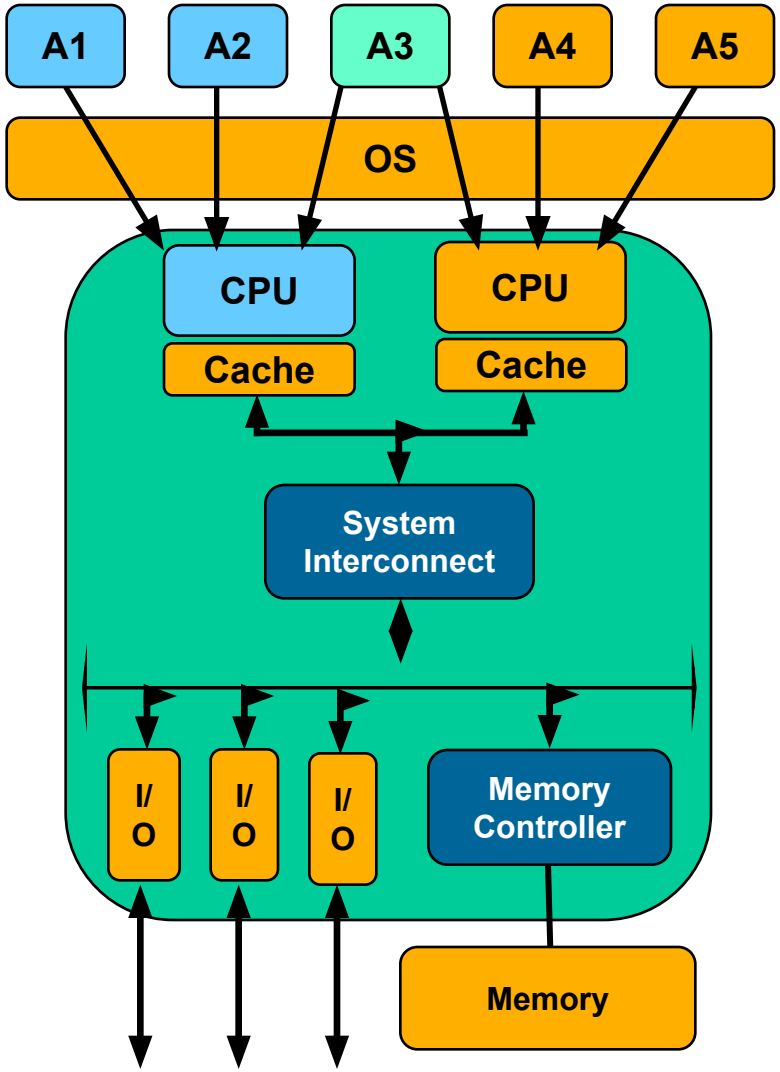
- **Простота перехода на multi-core**
  - Обычное приложение будет работать в multi-core варианте без каких-либо модификаций
  - Нет необходимости проверять или переделывать существующий код для обработки вопросов параллельности
  - Приложения могут работать как полностью симметричном режиме так и в bound режиме на одной системе

- **Контроль разработчиков над приложениями**
  - Разработчик имеет полный контроль над тем, на каком ядре будет исполняться тот или иной поток или приложение
    - Можно привязать к определенному ядру на уровне дизайна
    - Можно на программном уровне переводить любое приложение или поток с одного ядра на другое без остановки приложения
    - Динамическая балансировка нагрузки без перезапуска приожения

QNX Bound Multiprocessing

Лучшее из обоих моделей

- *Bound Multiprocessing* собирает в себе лучшее из симметричной и ассиметричной моделей
- Поддерживает существующие и оптимизированные для multi-core приложения
- Разработчик имеет полный контроль над приложениями
- Балансировка нагрзки
  - Как автоматическая на уровне ОС, так и настраиваемая разработчиком
  - Инструментарий для оптимизации балансировки нагрузки
- Высокая производительность
  - Обмен сообщениями и синхронизация потоков на уровне ядра ОС



Комплект разработчика QNX Momentics

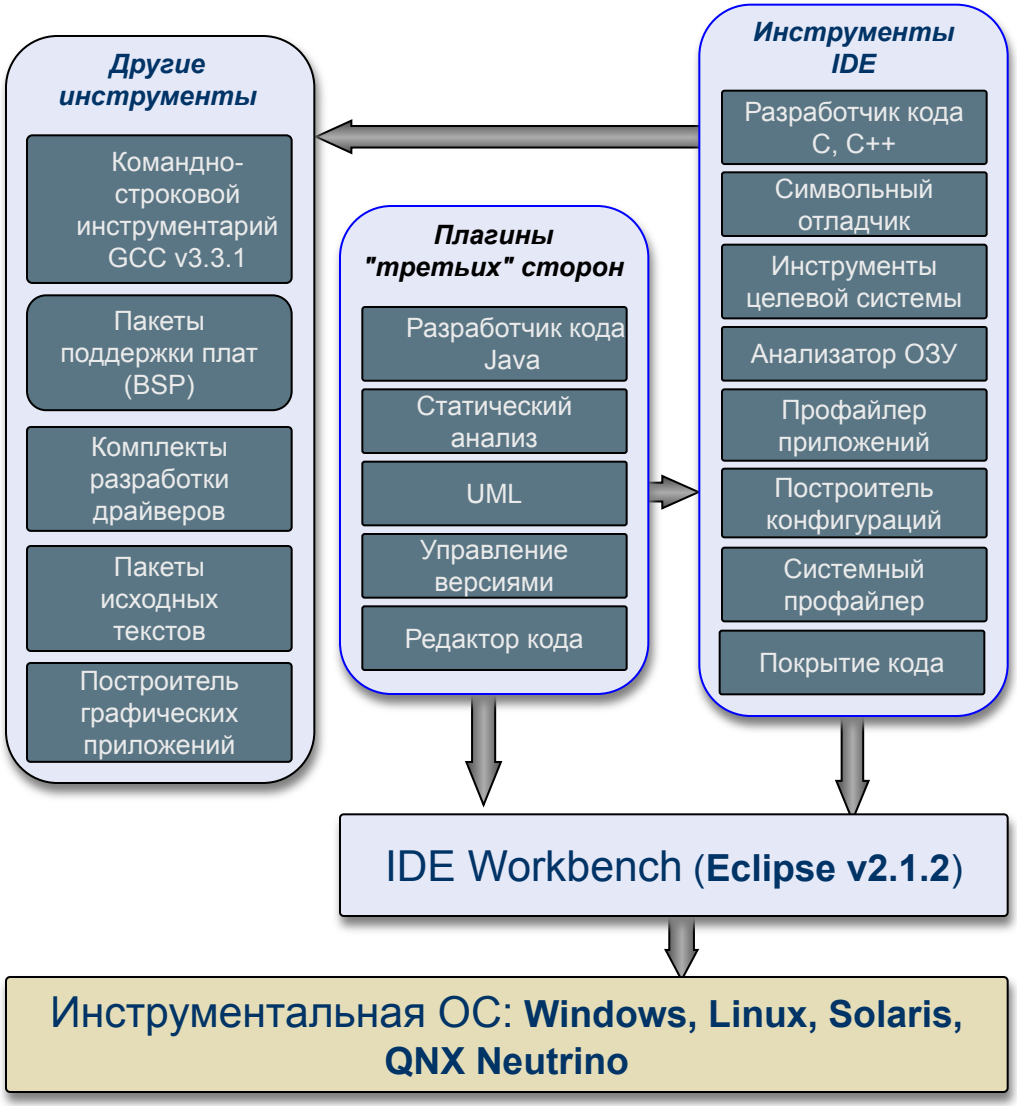
QNX - это мощная инструментальная платформа

Цикл разработки

Написание кода	Трансляция	Отладка	Оптимизация
Визуальный редактор	Компилятор	Отладчик	Профайлер
Генератор кода	Компоновщик	Средства диагностики	Средства верификации
Текстовый редактор	Виртуальная машина		
Система управления версиями	Интерпретатор		
Менеджер исходных текстов			

Комплект разработчика QNX Momentics

Вкратце о QN X Momentics



Инструментальные ОС
Windows, Linux, Solaris, QNX Neutrino

Целевые процессоры
ARM, MIPS, PPC, SH4, XScale, x86

Языки программирования
C, C++, Java, UML, SDL

Богатый выбор BSP
BSP для многих популярных плат и прототипов

Командная строка или графическая IDE
IDE работает с командно-строковым инструментарием

Доп. инструменты
Растущее сообщество Eclipse, объединяющее производителей ПО

Плагины Momentics

- **Разработчик кода**
  - C, C++, Java
  - Удобные "мастера"
  - Подсветка синтаксиса, шаблоны кода
- **Символьный отладчик**
  - Параллельная отладка нескольких приложений на C, C++, Java
- **Монитор целевых систем**
  - Детальная информация о процессах и потоках
- **Профайлер приложений**
  - Статистическое прфилирование
  - Подсчет вызовов и отслеживание пар вызовов с графическим представлением
  - Поддерживает разделяемые библиотеки
- **Анализатор ОЗУ**
  - Обнаружение двойного освобождения, использование нераспределенных блоков, переполнения и утечки памяти
  - Уничтожение/блокирование/отладка/ игнорирование в случае ошибки
- **Системный профайлер**
  - Программный "логический анализатор"
  - Анализ событий, полученных от диагностической версии ядра
- **Построитель встраиваемых конфигураций**
  - Определение зависимости модулей
  - Сокращение размеров библиотек
- **Photon Application Builder**
  - Quickly create Photon apps
  - Drag and drop widgets

Компилятор GCC v3.3.1

- QNX поддерживает оптимизированный компилятор GCC v3.3.1, обеспечивая совместимость с последними разработками сообщества GNU
- Двойная реализация дает разработчикам дополнительную возможность выбора
  - 2.95.3 (по умолчанию): обратная совместимость (в т.ч. C++)
  - 3.3.1: все преимущества новых функций
- Совместимость с промышленными стандартами
  - Поддержка стандарта C99 (препроцессирование, проверка формата)
  - Поддержка стандарта ABI (Application Binary Interface) для C++
- Улучшенные механизмы генерации кода
  - Улучшенные внутренние механизмы правления памятью
  - Оптимизация на основе профилей
  - Улучшенная производительность препроцессора
    - В среднем на 6-8% быстрее чем у v.2.95.3
- Оптимизация для процессоров: PowerPC, ARM, SH4, x86, Pentium

QNX IDE: разработчик кода C/C++

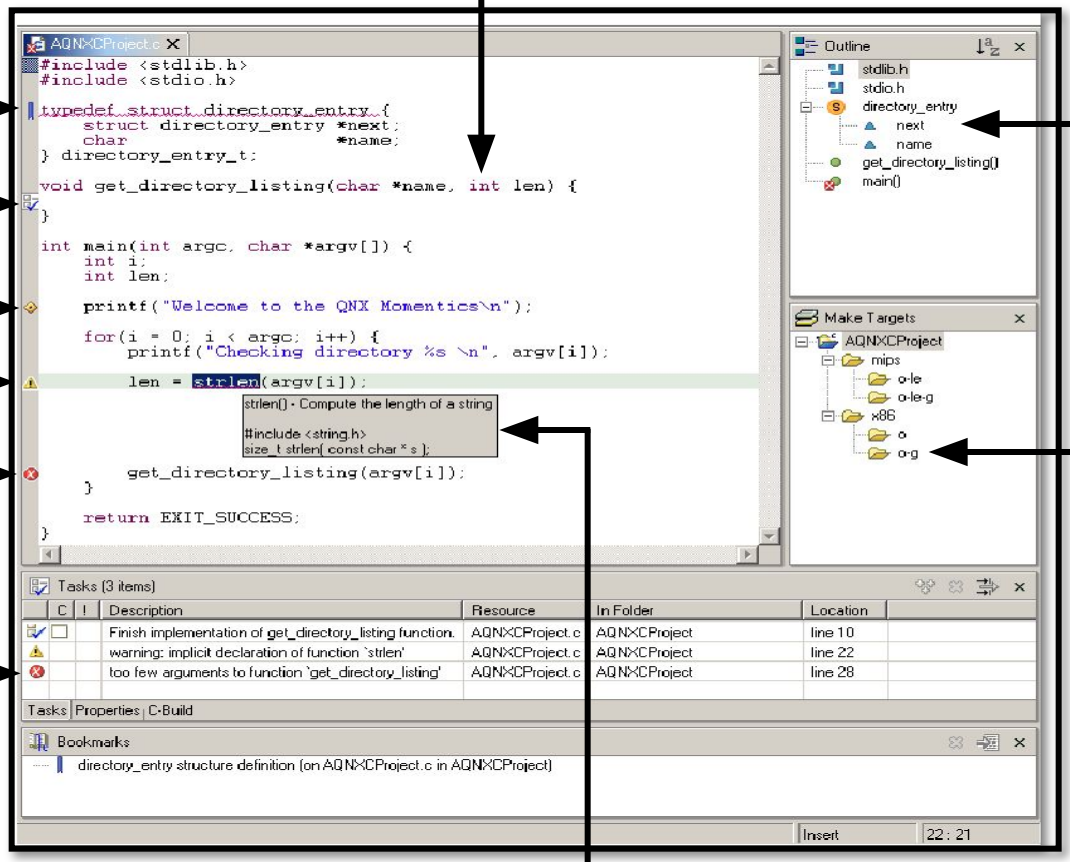
Идентификация ключевых слов, синтаксиса и парных скобок с первого взгляда

Закладки и задачи

Задание точек останова перед компиляцией

Идентификация ошибок и предупреждений компилятора с первого взгляда

Отслеживание всех ошибок и задач из единого списка



Список идентификаторов позволяет перейти к любой точке в исходном тексте

Щелкните два раза, чтобы построить проект для любой платформы

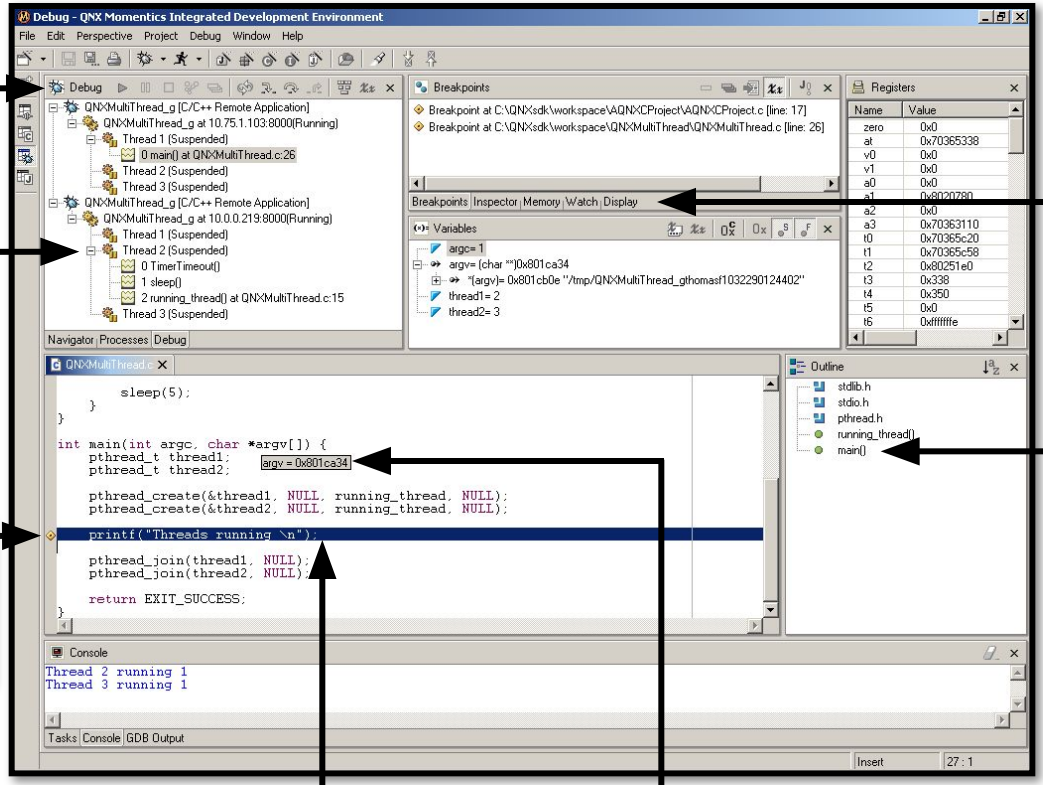
Наведите указатель мыши на функцию, чтобы посмотреть ее аргументы и нужные заголовки, или на имя переменной, чтобы увидеть ее тип

QNX IDE: символьный отладчик + анализатор ОЗУ

Используйте панель инструментов, чтобы запустить/остановить процесс или задать точку останова

Отслеживайте каждый поток независимо, или наблюдайте передачу управления между потоками

Щелкните два раза, чтобы задать точку останова



Щелкните здесь, чтобы посмотреть точки останова, переменные, память, регистры, и т.п.

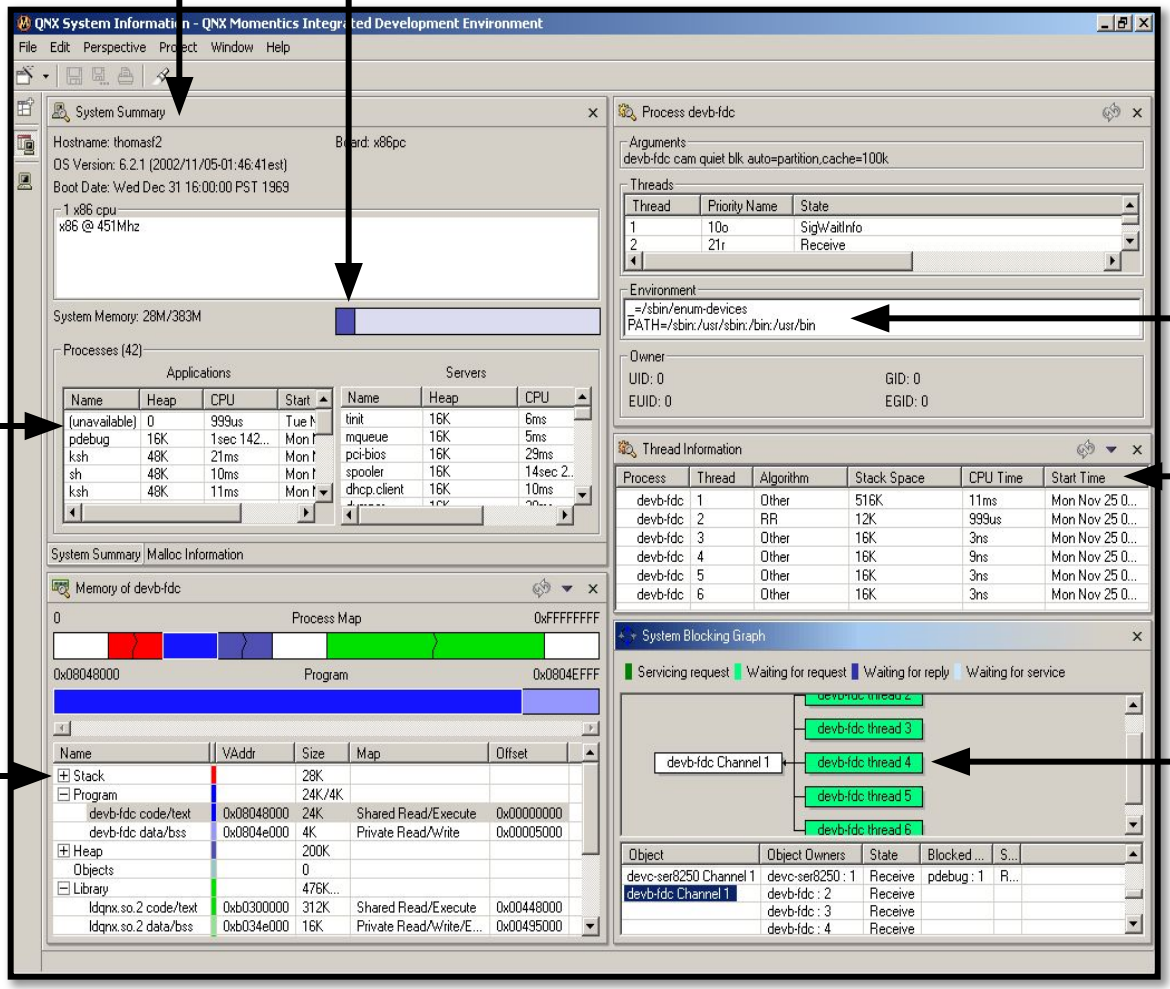
Перейдите к любой точке исходного текста

Редактируйте исходный текст прямо из отладчика

Наведите указатель мыши на имя переменной, чтобы увидеть ее значение

QNX IDE: монитор целевых систем

Системная информация и использование памяти



Использо-вание процессора и "кучи" процессами

Использо-вание памяти конкретным процессом

Просмотр окружения для каждого процесса

Сортировка и анализ потоков по различным атрибутам

Просмотр отношений блокирования

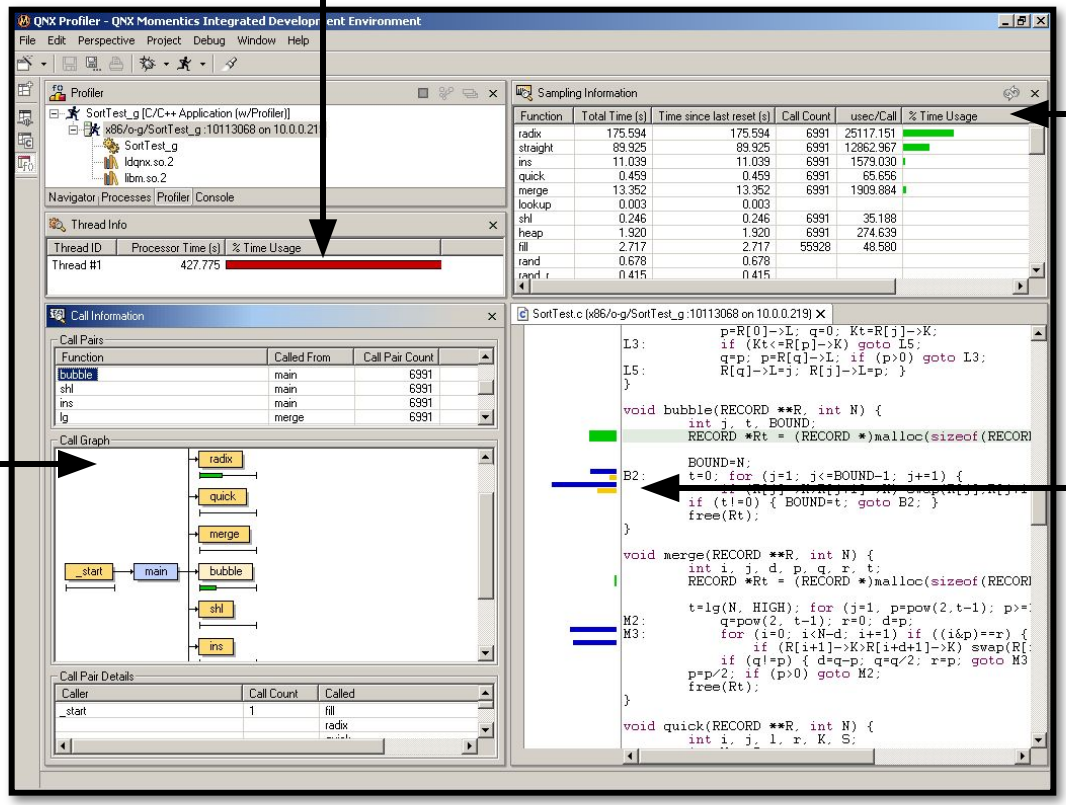
QNX IDE: удобный профилировщик кода

Определение наиболее загруженных потоков

Сортировка результатов по общему времени, процентной доле от общего времени, числу вызовов и т.п.

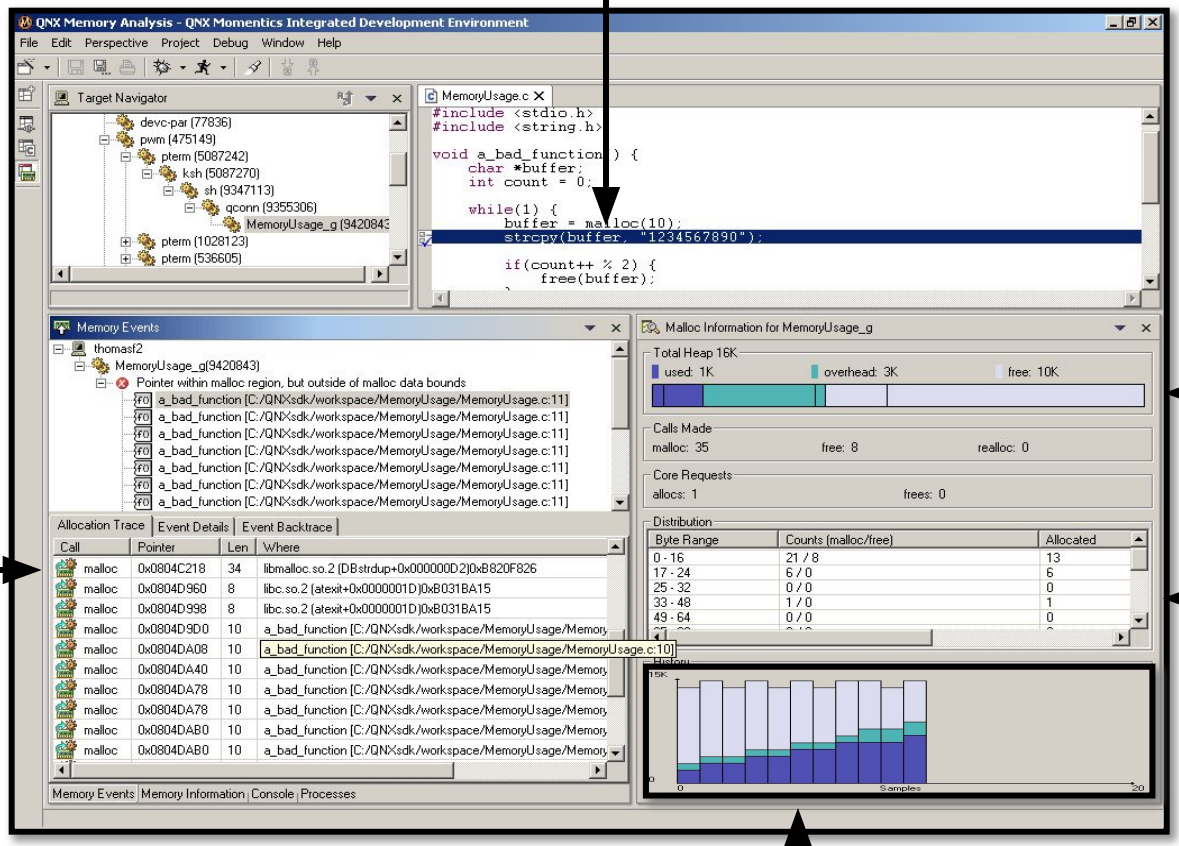
Дерево вызовов помогает сразу же оценить динамическую структуру выполнения приложений

Определение строк кода, потребляющих наибольшее количество процессорноговремени



QNX IDE: анализатор ОЗУ

Локализация переполнения буферов и запуск отладчика



Отслежива-ние операций распределе-ния памяти

Просмотр объема свободной и исполь-зуемой памяти – как в общем, так и для конкретных диапазонов

Просмотр динамики изменений в использовании памяти

QNX IDE: анализатор покрытия кода

- **Определяет используемые ветви кода**
  - **Указывает разработчикам, каким участкам кода уделять внимание для отладки и анализа производительности**
- **Упрощает контроль качества, оптимизацию, исправление ошибок и обслуживание**
  - **Методология контроля качества в военных, автомобильных и медицинских приложениях**
  - **Инструмент оптимизации в сетевых приложениях**
  - **Удобно для подразделений обслуживания и исправления ошибок, не участвовавших в разработке кода**
- **Интегрированный поддерживаемый компонент, в отличие от компонентов "третьих" сторон, дает клиентам уверенность**
  - **QSS – единственный производитель, в IDE которого интегрирован инструментарий анализа покрытия кода**

QNX IDE: анализатор покрытия кода

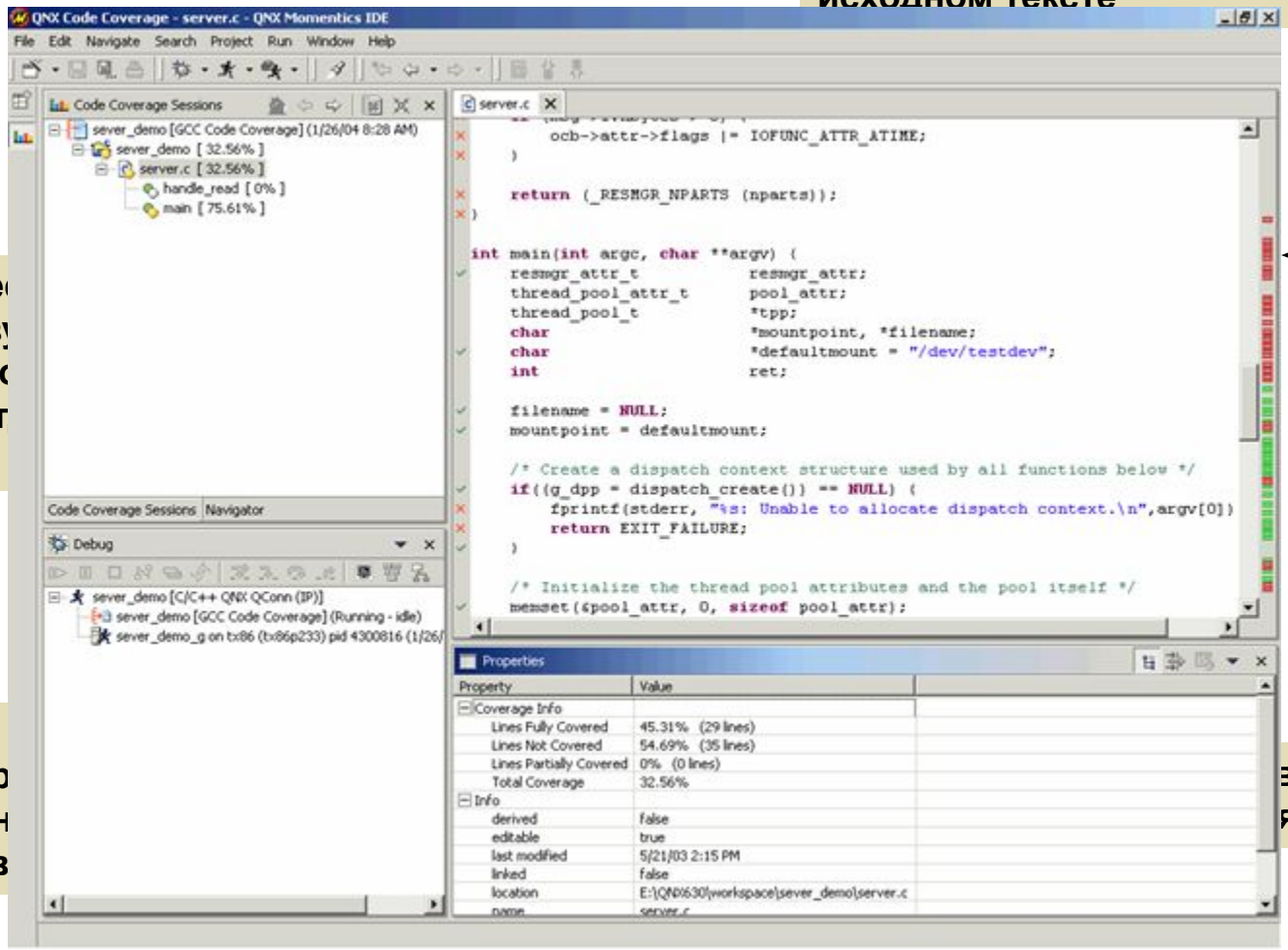
Интегрирован в IDE

Интегрирован с редактором кода: графическое представление покрытия непосредственно в исходном тексте

Просмотр се "живые" резу бинарного по вплоть до от функций

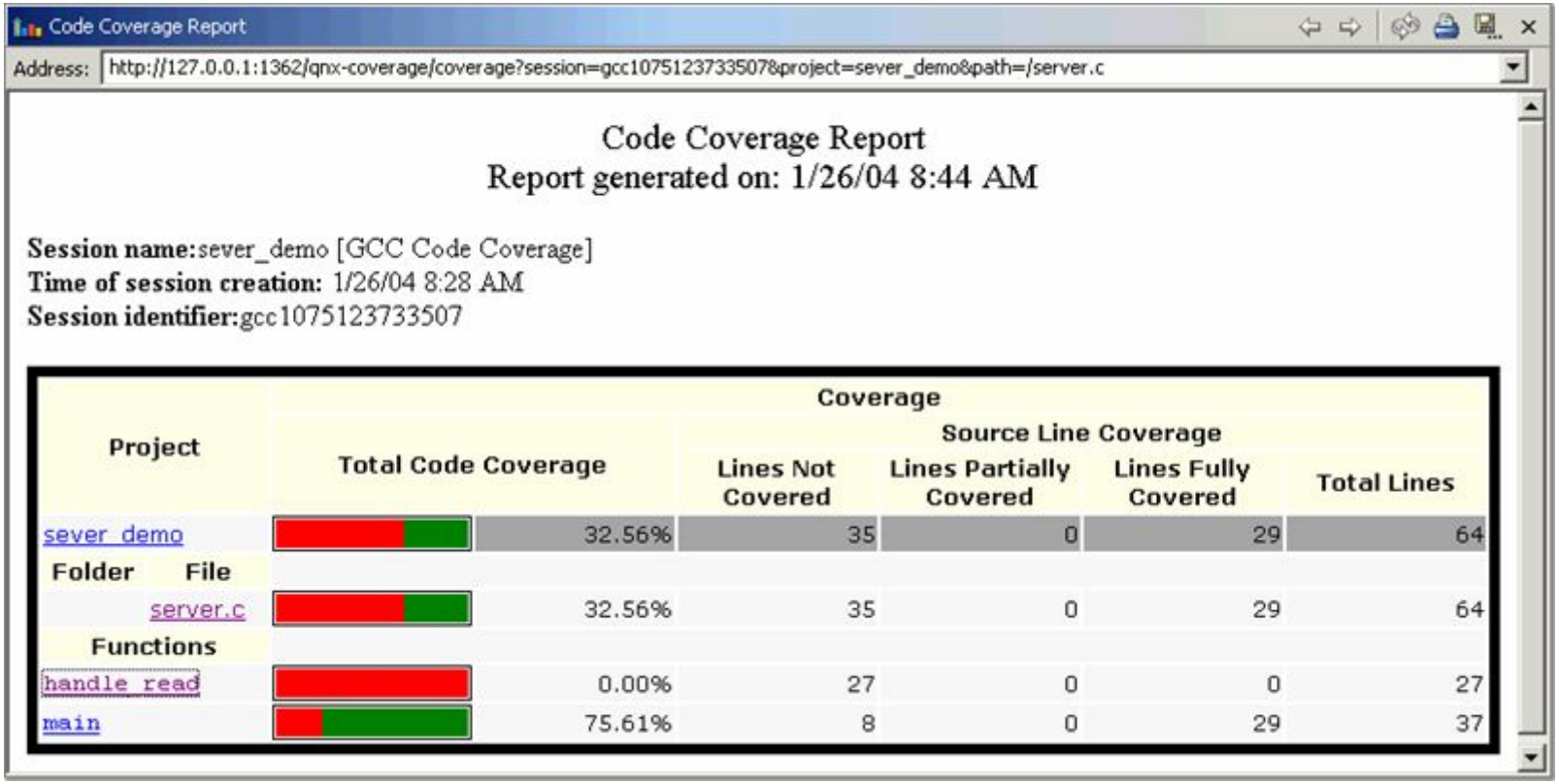
Отладка: просмотр запущенн прцессов

за сессии: я оценка



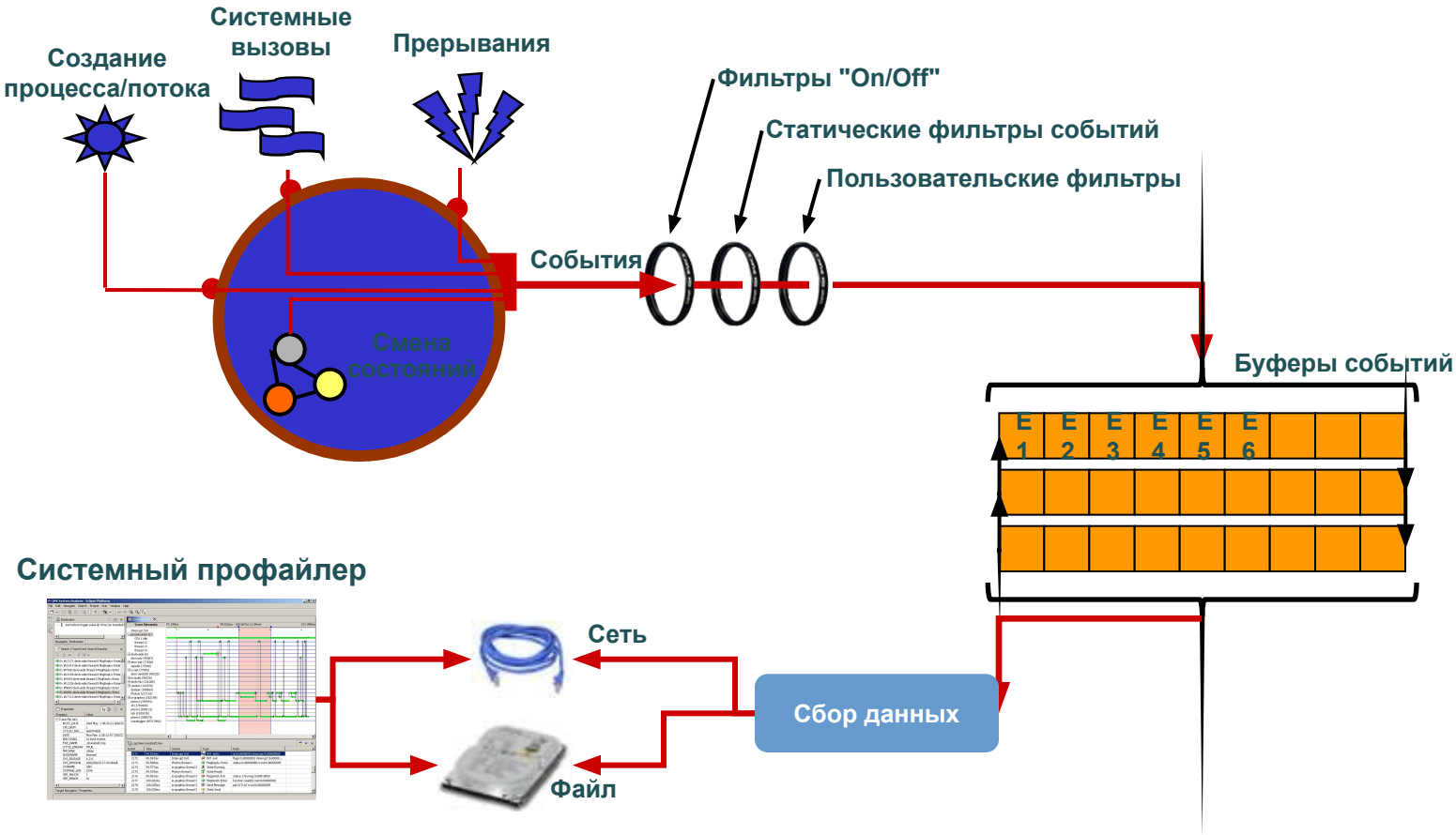
QNX IDE: отчет о покрытии кода

Генератор отчетов:
- **Отчеты в формате HTML для дальнейшего анализа, по каждой сессии**
- **Статистика для контроля качества**

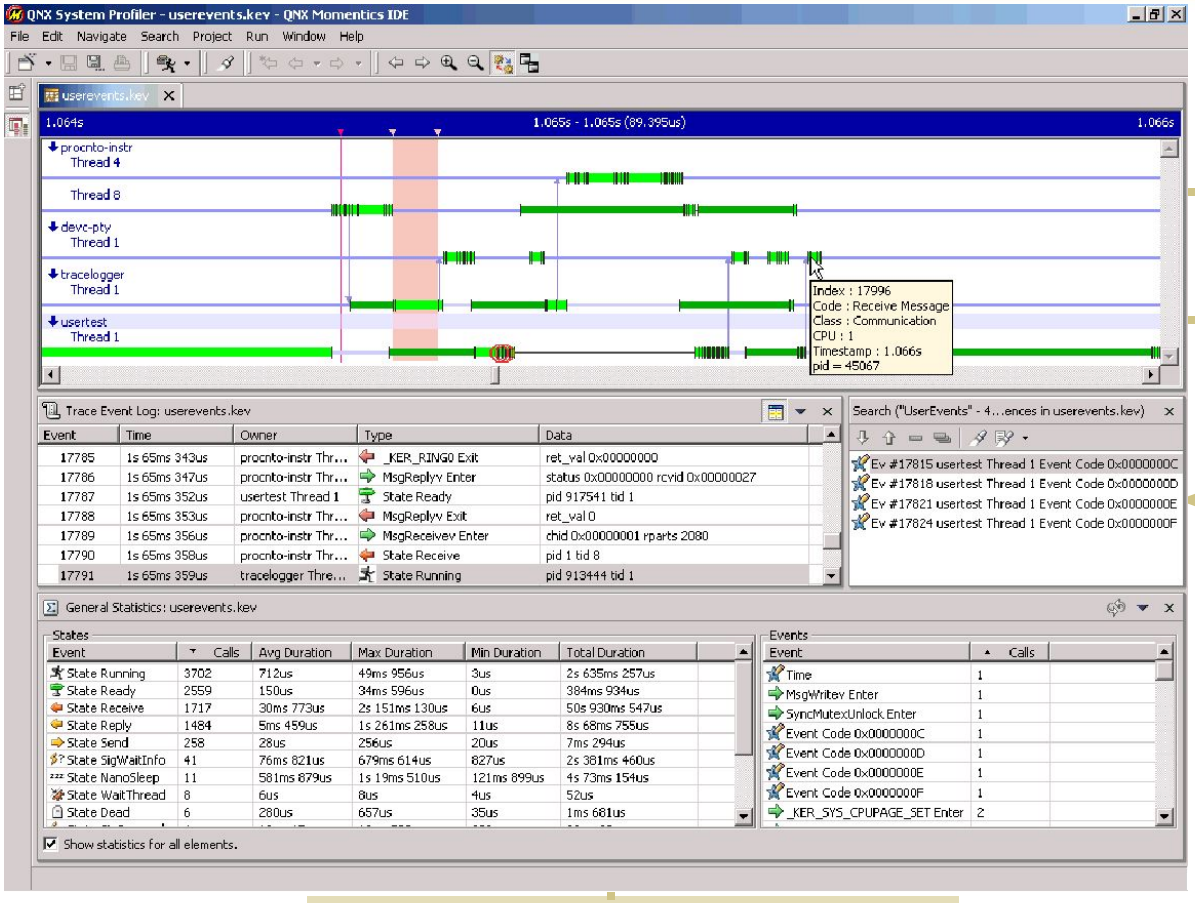


Диагностическая версия ядра

Диагностическая версия ядра ведет журнал событий, фильтрует их и сохраняет в буферах, содержимое которых можно сохранять и анализировать



QNX IDE: системный профайлер



Изменение временного масштаба, выбор нужных процессов, создание нестандартных представлений

Улучшенный интерфейс упрощает навигацию

- Более прозрачен
- Меньше элементов
- Поддержка разбиения окон и прокрутки

Всплывающие подсказки

- Дополнительные сведения (процессор, PID)
- Текстовые пояснения

Новое окно статистики

- Табличное представление
- Статистические выборки
- Активность владельцев событий

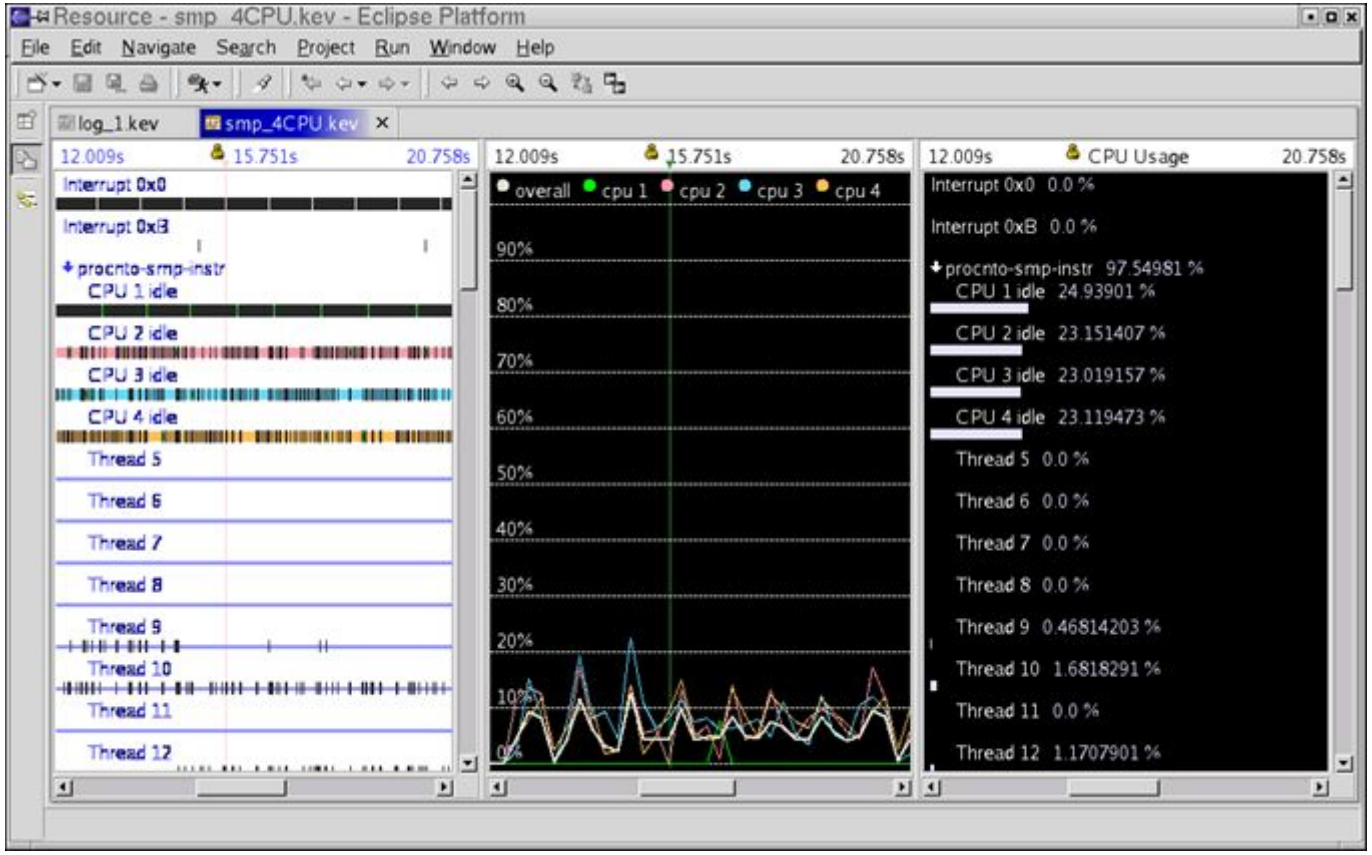
А также…

Фильтры пост-обработки переработаны с учетом расширяемости

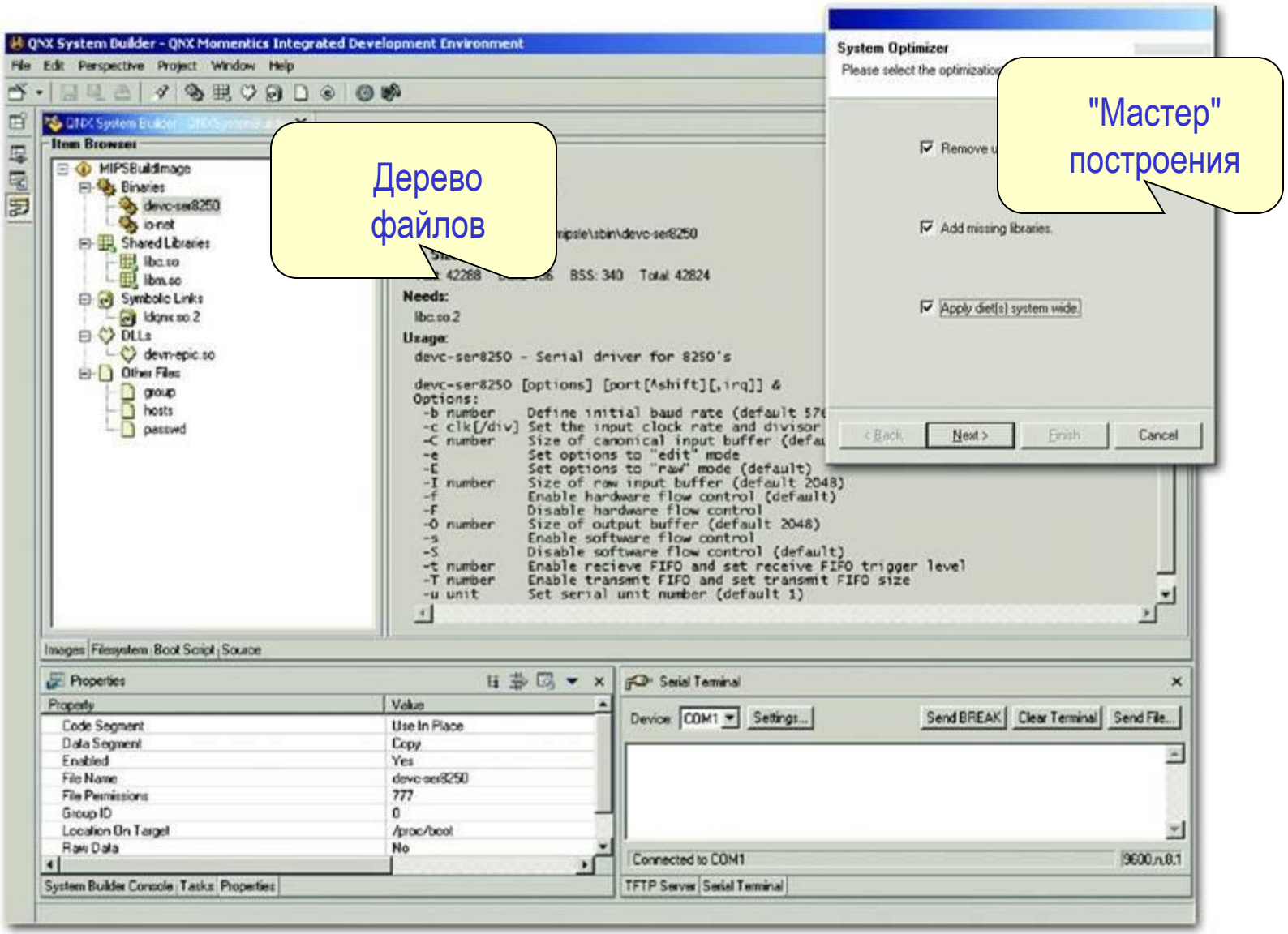
Системный профайлер: представление "Активность CPU"

% активности CPU от общего времени

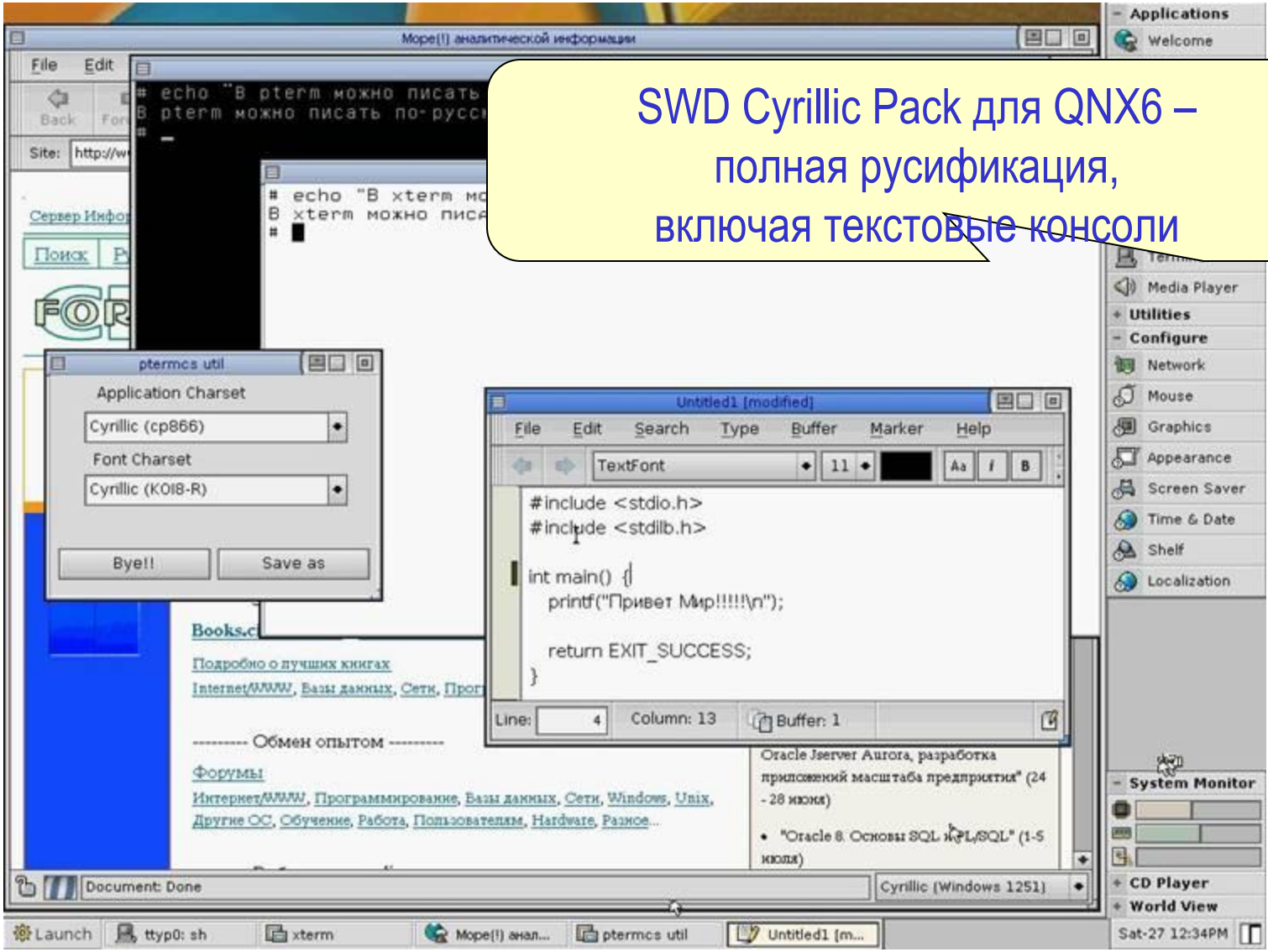
Разбиение активности CPU по элементам трассировки



QNX IDE: построитель встраиваемых конфигураций



Поддержка русского языка в QNX



QNX помогает экономить

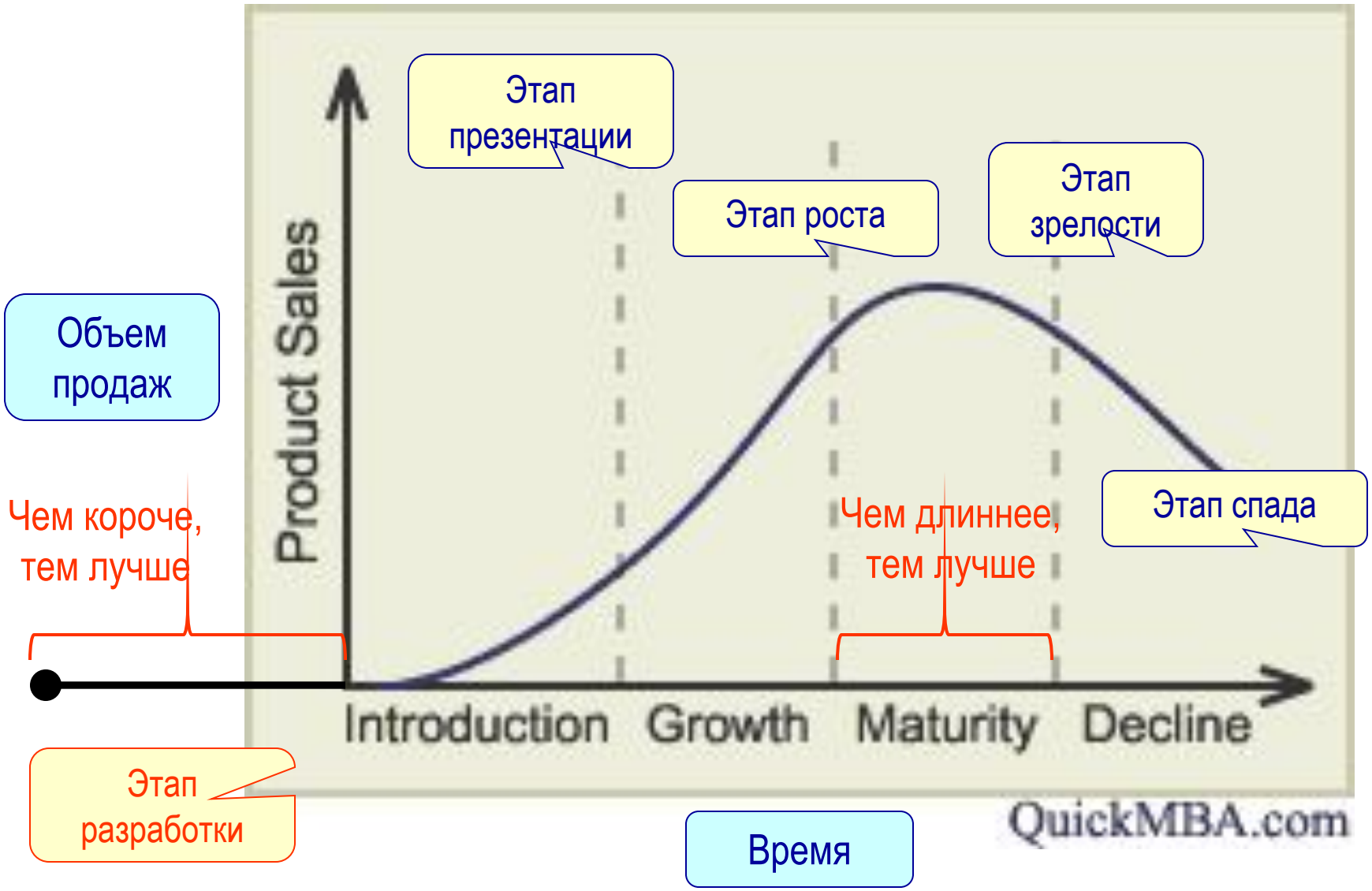
- **QNX как средство сокращения срока разработки и времени выведения на рынок (TTM) нового продукта**

TTM = Time-To-Market,
"время выхода на рынок"

- **QNX и сокращение суммарной стоимости владения (TCO)**

TCO = Total Cost of Ownership,
"суммарная стоимость владения"

Жизненный цикл продукта



Как ОС может сократить TTM?

QNX как средство сокращения TTM

Эффективный инструментарий	▪ Интегрированный комплект QNX Momentics ▪ Инструментарий "третьих" фирм ▪ Стартовые комплекты
Прозрачная методология разработки	▪ Изящная архитектура ▪ Модульная организация ▪ Встроенная распределенная сеть
Простота адаптации к оборудованию	▪ Открытая унифицированная модель драйвера ▪ DDK с примерами ▪ Доступность исходных текстов
Возможность переноса кода из других проектов, в т.ч. из других ОС	▪ Полная поддержка POSIX API ▪ Модель "клиент/сервер" для сервисов ОС
Доступность готовых решений	▪ Партнерская сеть QNX Partner Network
Простота встраивания	▪ Компактность ▪ Модульность
Профессиональные сервисы	▪ SWD Software Ltd. – поддержка, консалтинг, сертифицированное обучение, заказные разработки

Из чего складывается ТСО

Разработчик

- Стоимость инструментария
- Стоимость обучения
- Ресурсы на разработку (время, персонал, рабочие места, поддержка, консалтинг и т.п.)
- Стоимость комплектующих и сборки
- Стоимость сопровождения

Конечный пользователь

- Цена продукта
- Стоимость внедрения (монтаж, пусконаладка, обучение персонала)
- Стоимость отказов (потери на простоях, ликвидация последствий и т.п.)
- Стоимость обслуживания (поддержка, профилактика, ремонт, модернизация)

QNX как средство сокращения TCO

Дешевая аппаратура	▪ Неприхотливость к ресурсам
Стоимость программных компонентов	▪ Модульное лицензирование
Надежность	▪ Надежная архитектура на основе микроядра ▪ Все системные модули выполняются вне пределов ядра в защищенных адресных пространствах ▪ Использование аппаратного диспетчера памяти
Живучесть	▪ Поддержка автоматического самовосстановления на уровне отдельных компонентов ▪ Поддержка режима высокой готовности (коэффициент готовности 99.999% и выше)
Дешевизна в обслуживании, диагностике и модернизации	▪ Возможность обновления и перезапуска любого программного модуля "на лету" без перезагрузки ОС ▪ Поддержка удаленного обновления с использованием защищенных сетевых протоколов ▪ Масштабируемость и расширяемость ▪ Полностью русифицирована ▪ Поддержка удаленного интерфейса пользователя

Небольшое резюме

	Исследования/ разработка	Тестирование/ интеграция	Внедрение/ поддержка
Защита памяти	Не надо "пересобирать" ядро	Ранняя диагностика ошибок	Динамические апгрейды
SMP	"Встроенные" возможности расширения	Решение проблем производительности	Масштабируемые системы для ресурсоемких вычислений
Модульность	Параллельная разработка	Инкрементное тестирование	Апгрейд сервисов без прерывания работы системы
Распределенные вычисления	Удобная модель разработки	Корректное поведение ПО как в локальных, так и в сетевых конфигурациях	Доступ ко всем ресурсам системы с одного узла
POSIX-совместимость	Перенос готового кода	Тестирование на недорогом оборудовании	Общий интерфейс для настройки и поддержки
	Архитекторы Разработчики	Тестеры Контроль качества	Сервисные инженеры Инженеры поддержки

Генеральный директор / Владелец продукта

QNX и рынок специалистов

Знания/навыки	Общедоступно в POSIX-среде	Другие ОСРВ?	QNX?
Навык пользователя ОС	FreeBSD, Solaris, Linux	Нет	Да
Навык администратора ОС	FreeBSD, Solaris, Linux	Нет	Да
Знание языков программирования	C/C++	Да	Да
Навык пользователя средств разработки	GNU и др. (с открытым исходным текстом)	Нет	Да
Навык пользователя прикладных программ	GNU и др. (с открытым исходным текстом)	Нет	Да
Навык программирования задач реального времени	RTLinux?	Нет	Нет
Навык использования специфичного инструментария	–	Нет	Нет

Вопросы?

SWD Software Ltd.
Официальный дистрибьютор QNX

196135, Санкт-Петербург,
пр. Юрия Гагарина 23
тел.: (812) 102-0833
тел.: (812) 373-0260
факс: (812) 373-0497
web: http://www.swd.ru/
e-mail: qnx@swd.ru