



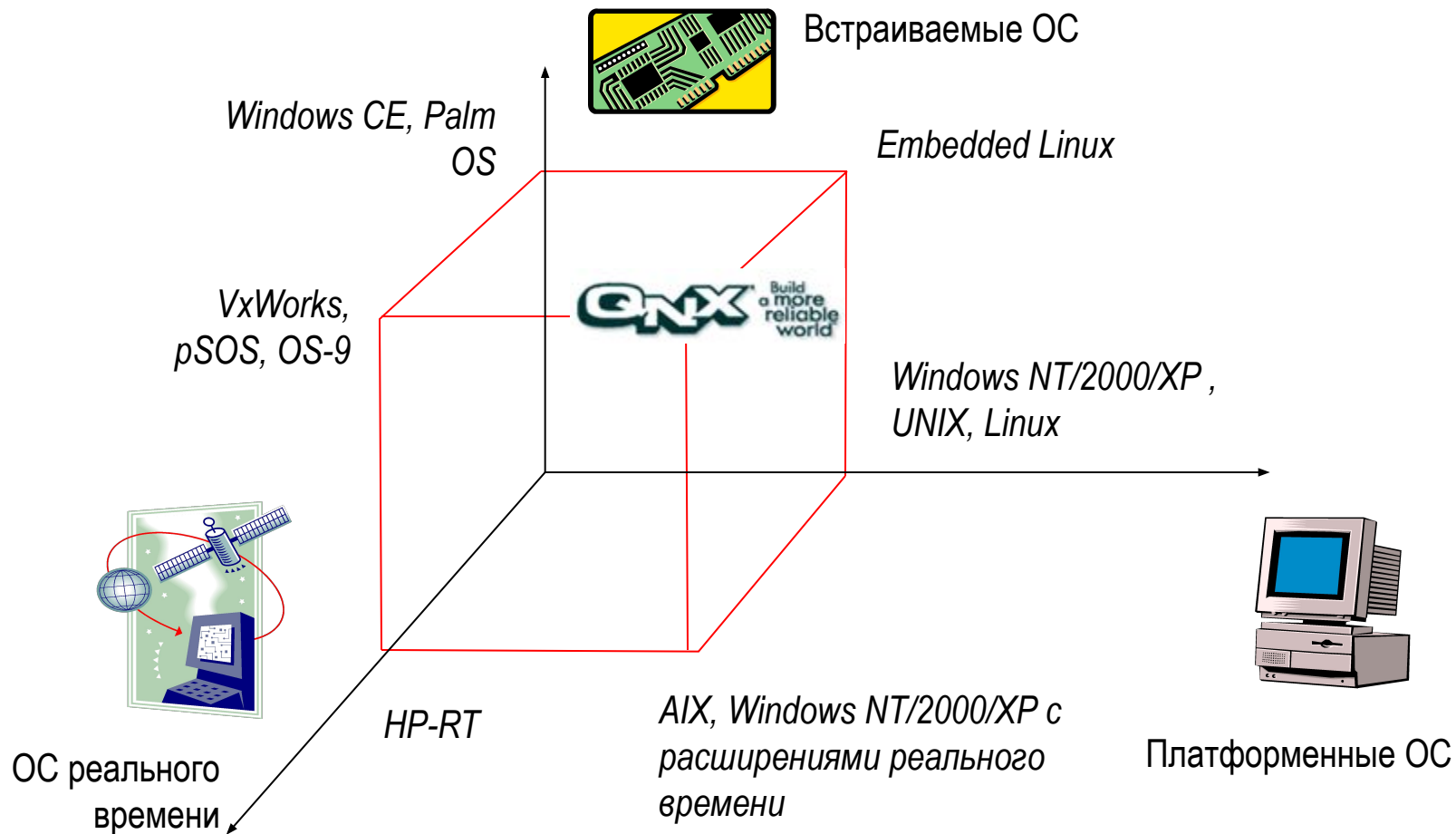
ОС реального времени QNX и интегрированный комплект разработчика QNX Momentics

Александр Трофимов
SWD Software Ltd.



OCPB QNX

Чем QNX отличается от других ОС?

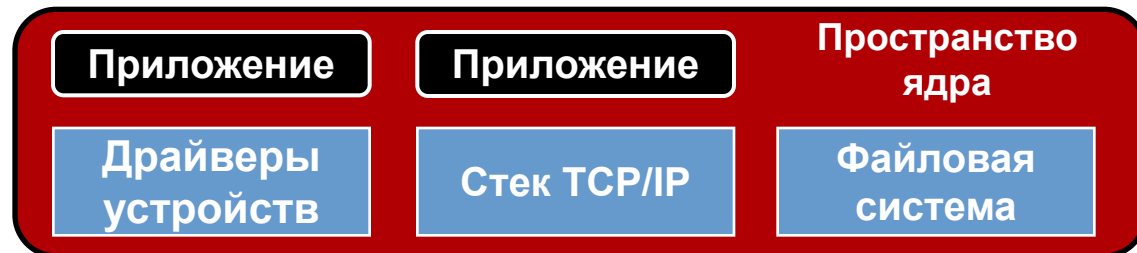




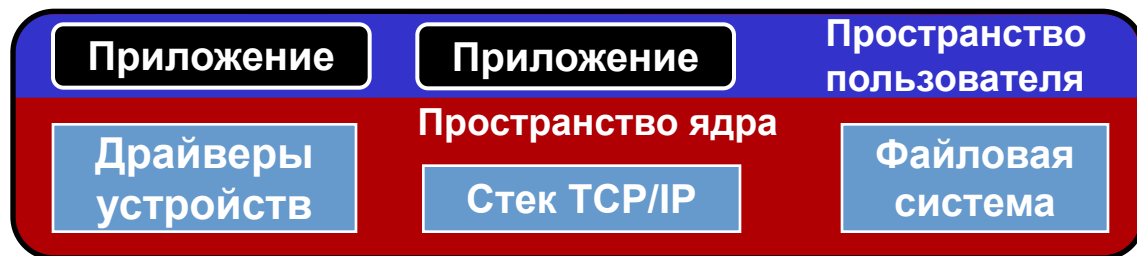
QNX как ОС жесткого реального времени

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 (<i>sched_yield()</i>), мкс	1.62	1.30	1.85
Переключение контекста между потоками одного процесса, мкс	2.33	1.87	1.9

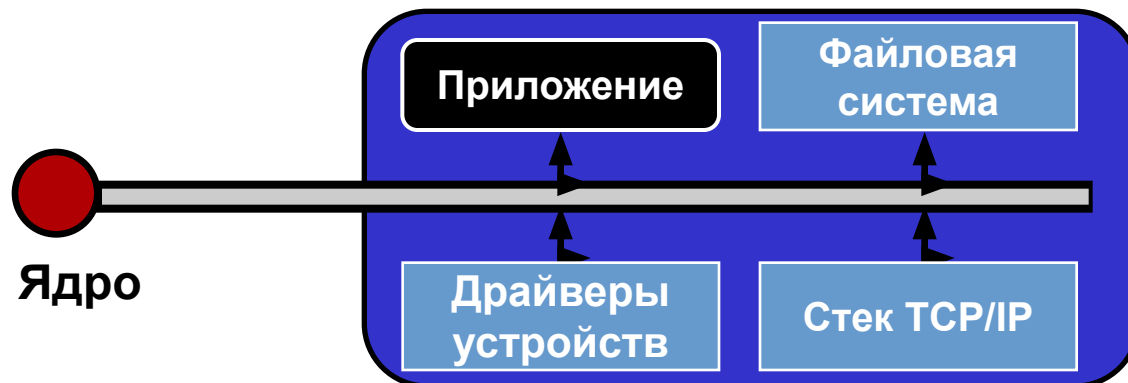
**Исполняемый модуль
реального времени
(напр. VxWorks)**



**Монолитное ядро
(напр. Linux)**

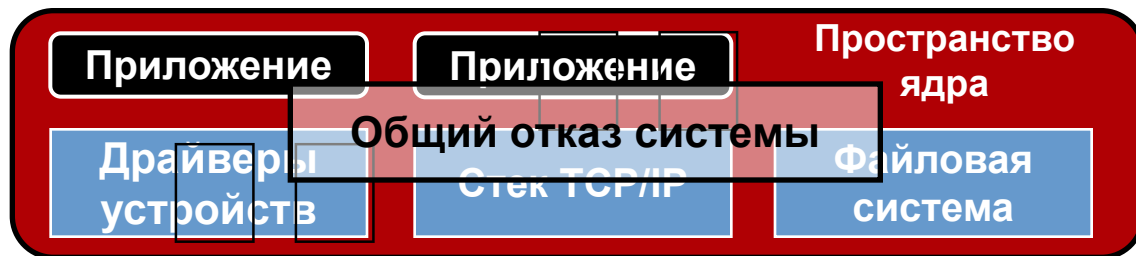


**Микроядро
(напр. QNX Neutrino)**



Исполняемый модуль реального времени

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



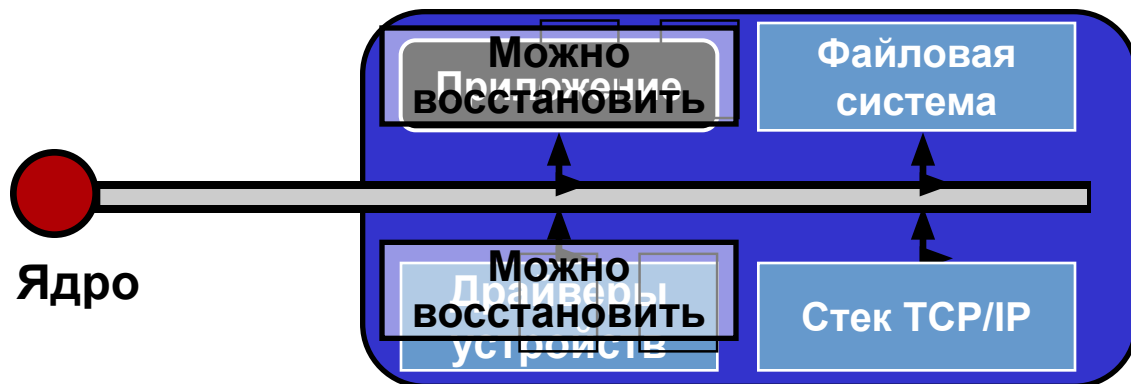
Монолитное ядро (NT / Unix / и т.п.)

- > MMU, частичная защита
- > Защищены только приложения



Микроядро (QNX Neutrino)

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



□ Задачи общаются посредством сообщений



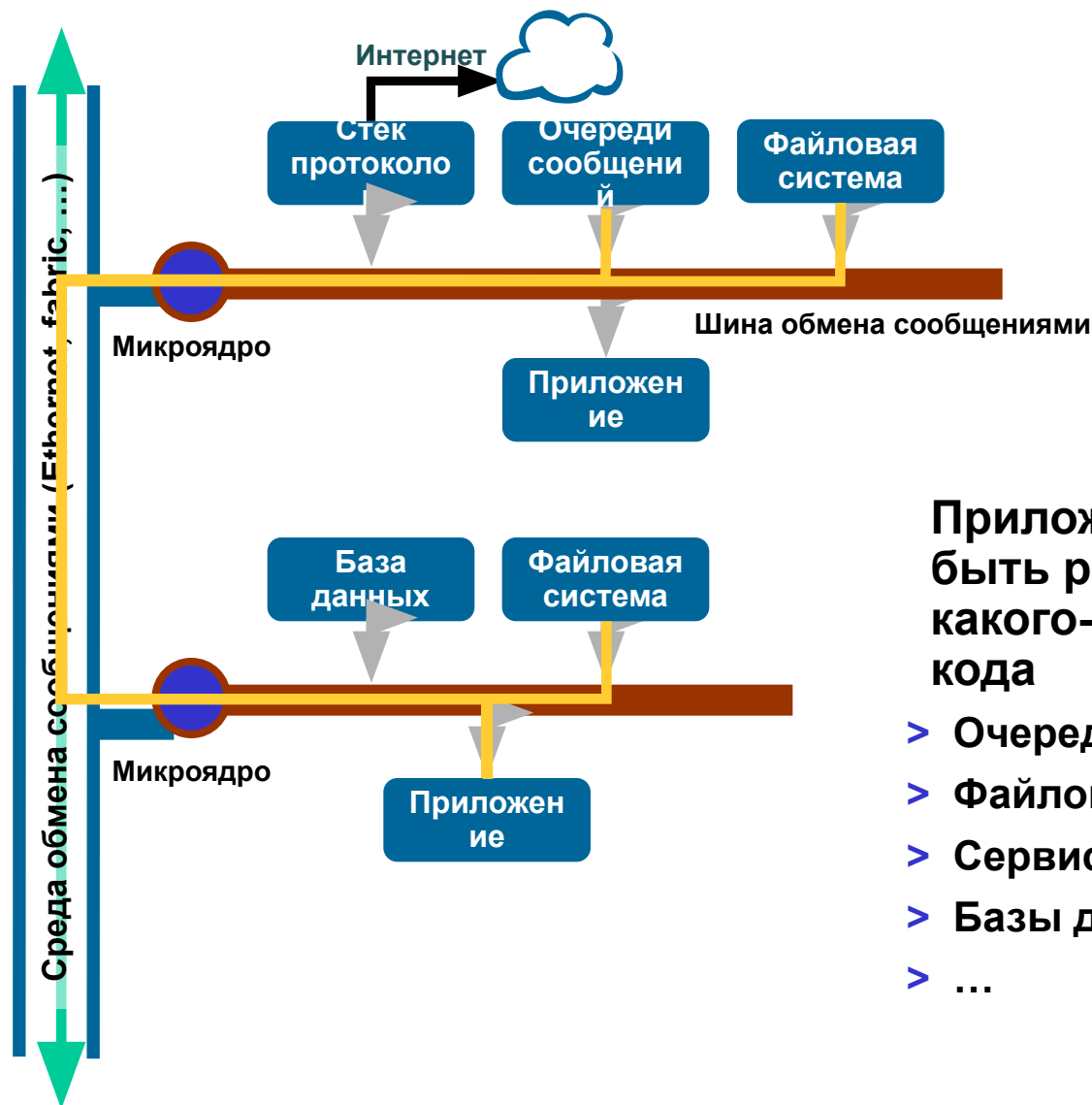
□ Использование сообщений органично "развязывает" задачи

□ Над сообщениями надстроены вызовы POSIX

```
fd = open("/dev/tcpip", ...)
read, write, stat, devctl, ...
close
```

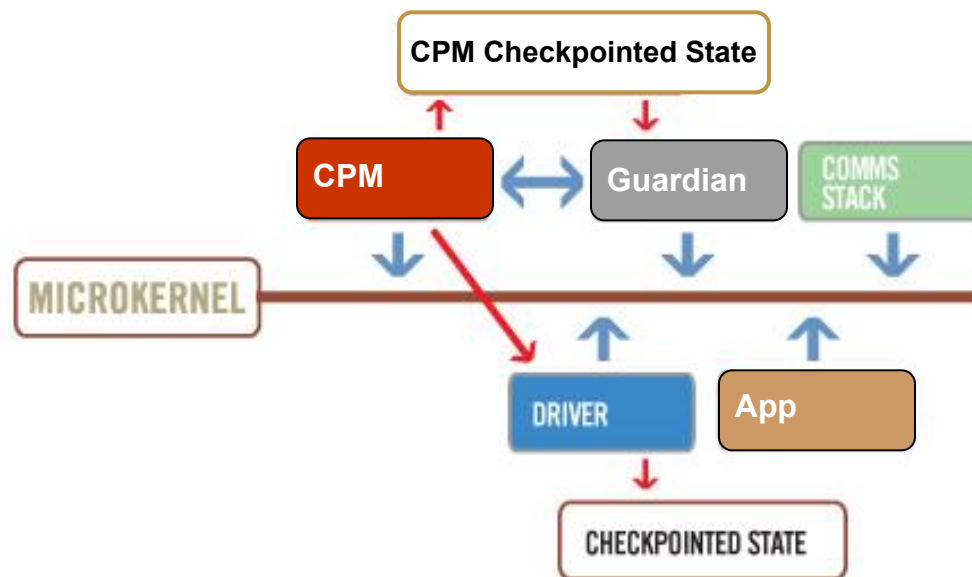
□ ... и другие вызовы POSIX

- realtime signals
- pipes and POSIX mqueues
- mutexs, condvars, semaphores
- barriers, sleep
- reader/writer locks

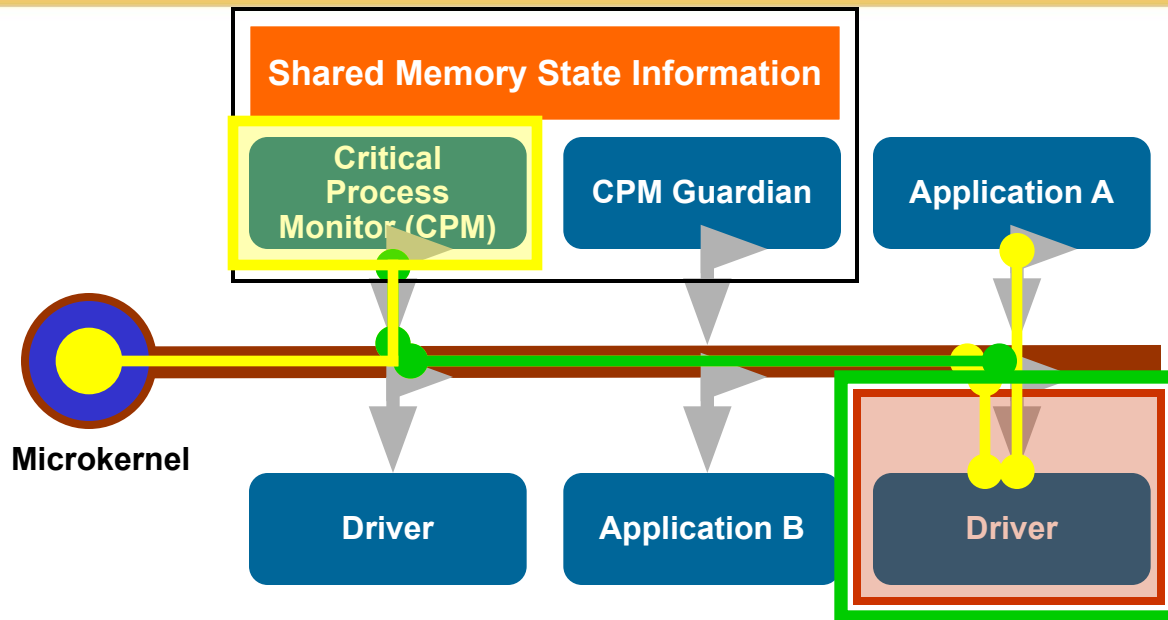


Приложения/серверы могут быть распределенными без какого-либо специального кода

- > Очереди сообщений
- > Файловые системы
- > Сервисы
- > Базы данных
- > ...



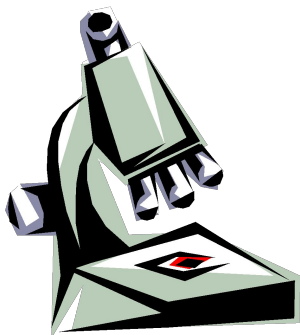
- **Основа системы высокой готовности – Монитор Ключевых Процесов (CPM). Выполняет мониторинг выбранных компонентов и обеспечивает восстановление после сбоев**
- **Процесс-охранник дублирует CPM**
- **Клиентская библиотека позволяет компонентам незамедлительно и прозрачно восстанавливать все соединения**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



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



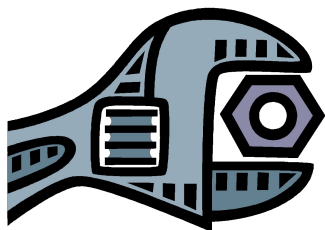
QNX как встраиваемая и масштабируемая ОС



- **Компактность и неприхотливость**
 - уместается 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?»





- **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

□ Что такое Адаптивная Декомпозиция?

- Новый продукт QNX, расширяющий OSCPВ 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

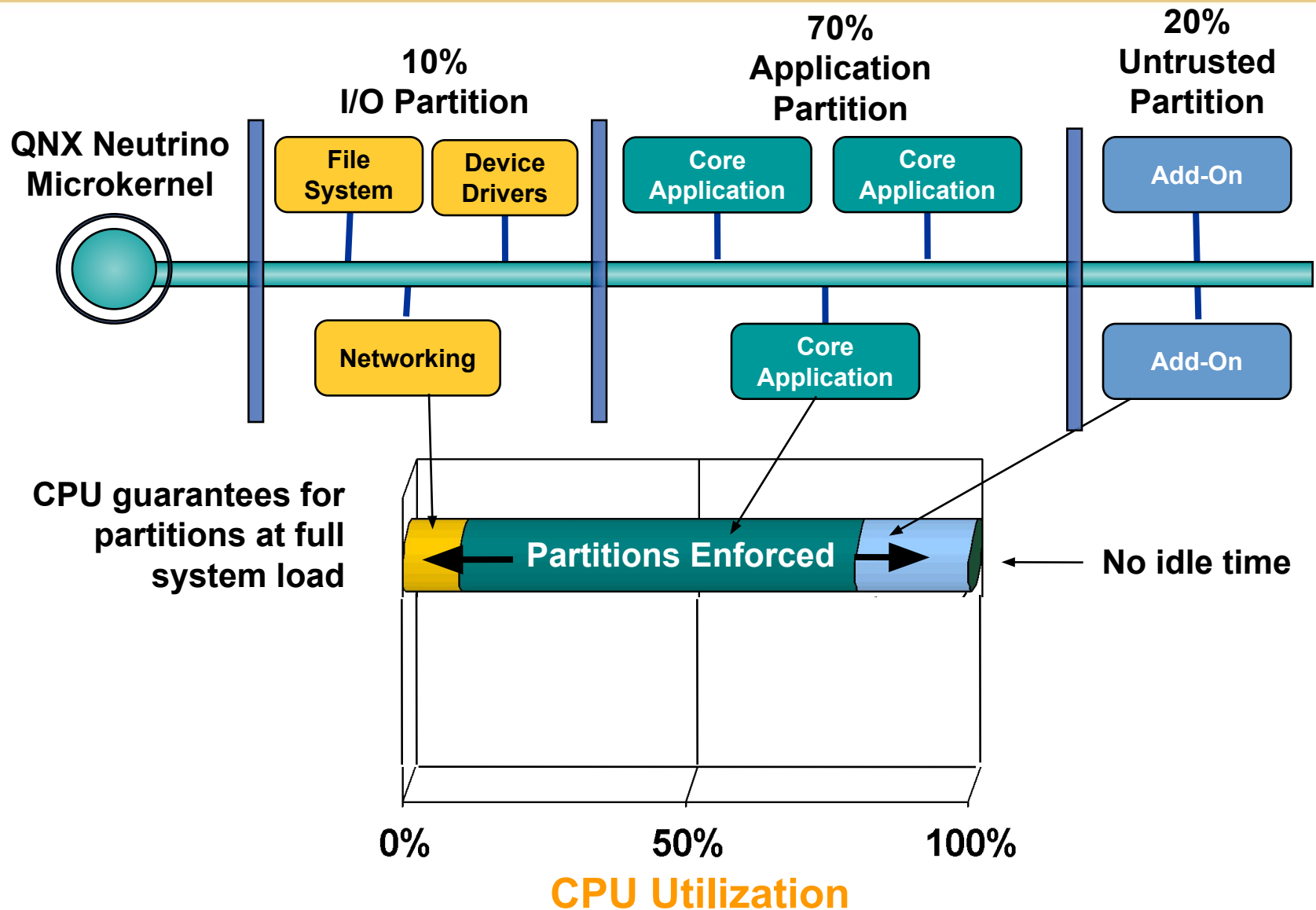


Зачем нужна Adaptive Partitioning?

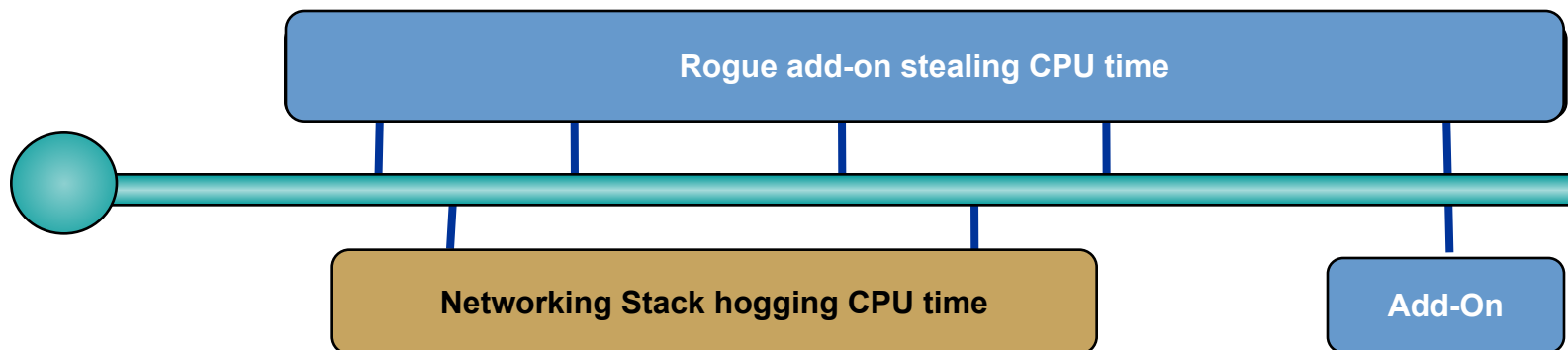
- **Основанное на приоритетах вытесняющее планирование может гарантировать выполнение приоритетных задач в системах реального времени**
 - Как только задача готова к выполнению, она сразу же получает ресурсы процессора
 - С усложнением системы, множество задач борются за ресурсы CPU и становится сложно 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 находиться 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**



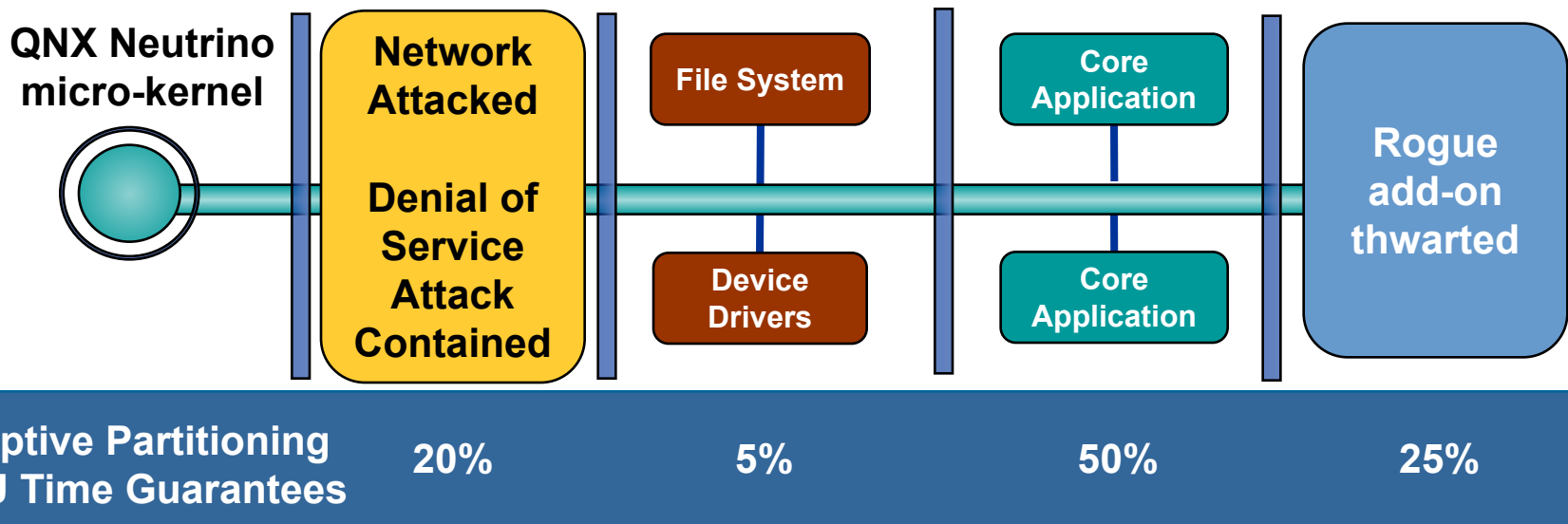
Максимизация производительности



- **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**



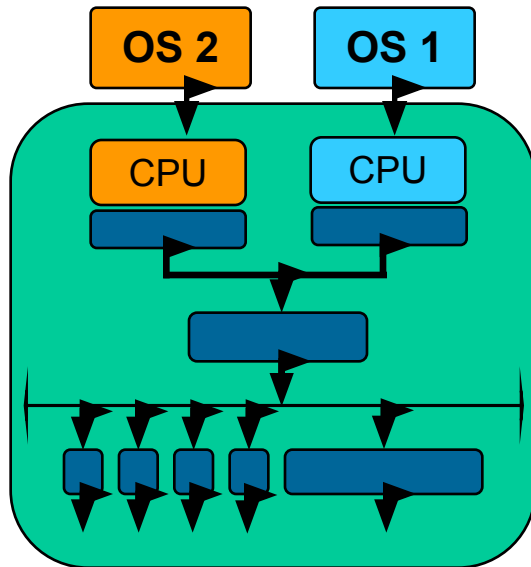
- **Создайте разделы для защиты критичных системных ресурсов**
 - Гарантия ресурсов CPU для базовых функций
 - Наследование партиций гарантирует время CPU всем сервисам ОС (драйвера, файловые системы, сетевая система)
- **Защита основные приложений от угроз**
 - Минимизация влияние вредоносного ПО
 -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





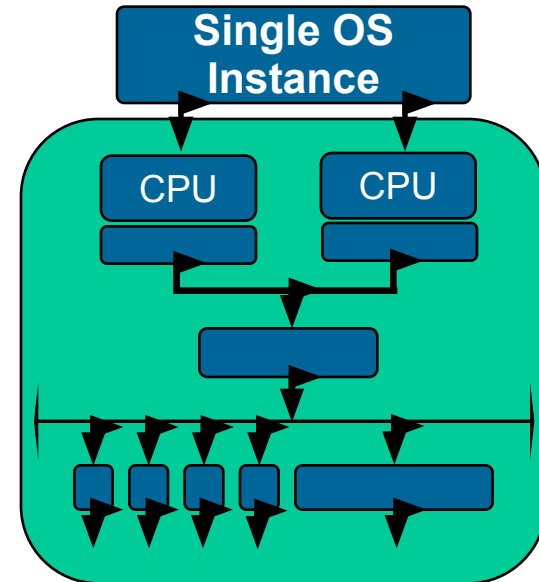
- **QNX Neutrino Multi-core Technology Development Kit**
единственная в отрасли полнофункциональная платформа
для нового поколения multi-core кристаллов
- **При помощи QNX Neutrino Multi-Core Technology Development Kit вы сможете:**
 - Быстро перенести приложения для однопроцессорной архитектуры на любую 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

Asymmetric



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

Symmetric



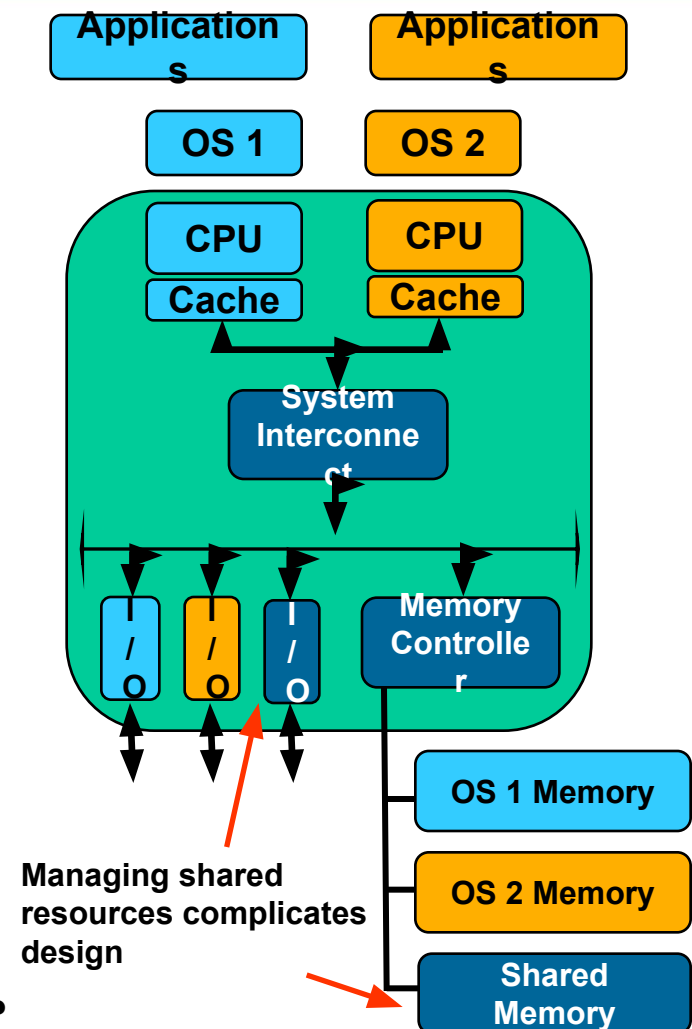
- 2 ядра, одна ОС
- QNX, Linux

Asymmetric Model – «3A»:

- 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

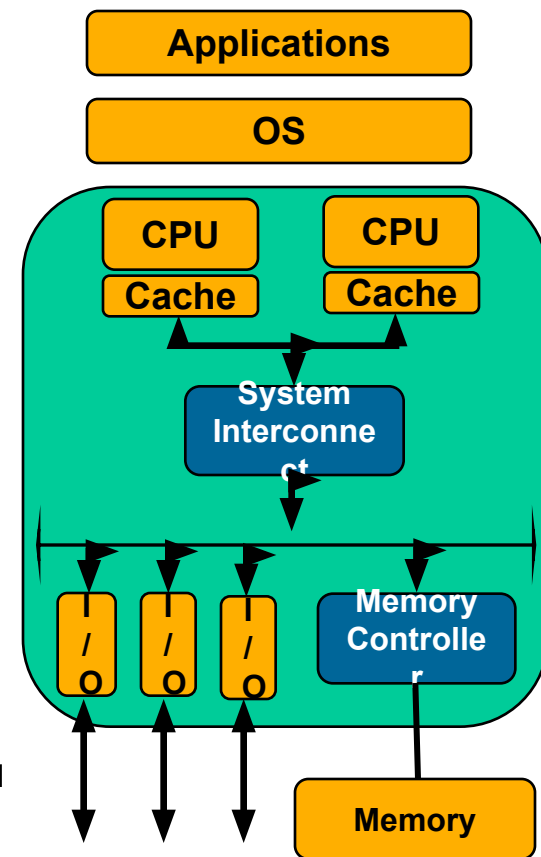


□ Symmetric Model – «3А»:

- Хорошо масштабируется. Безшовная поддержка многоядерности без модификации кода
- 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 обеих моделей

- ОС работает в симметричном режиме с возможностью «привязать» приложения к конкретному ядру
- Одна ОС имеет полный контроль
 - Ресурсы распределяются 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 обеих моделей

□ Простота перехода на multi-core

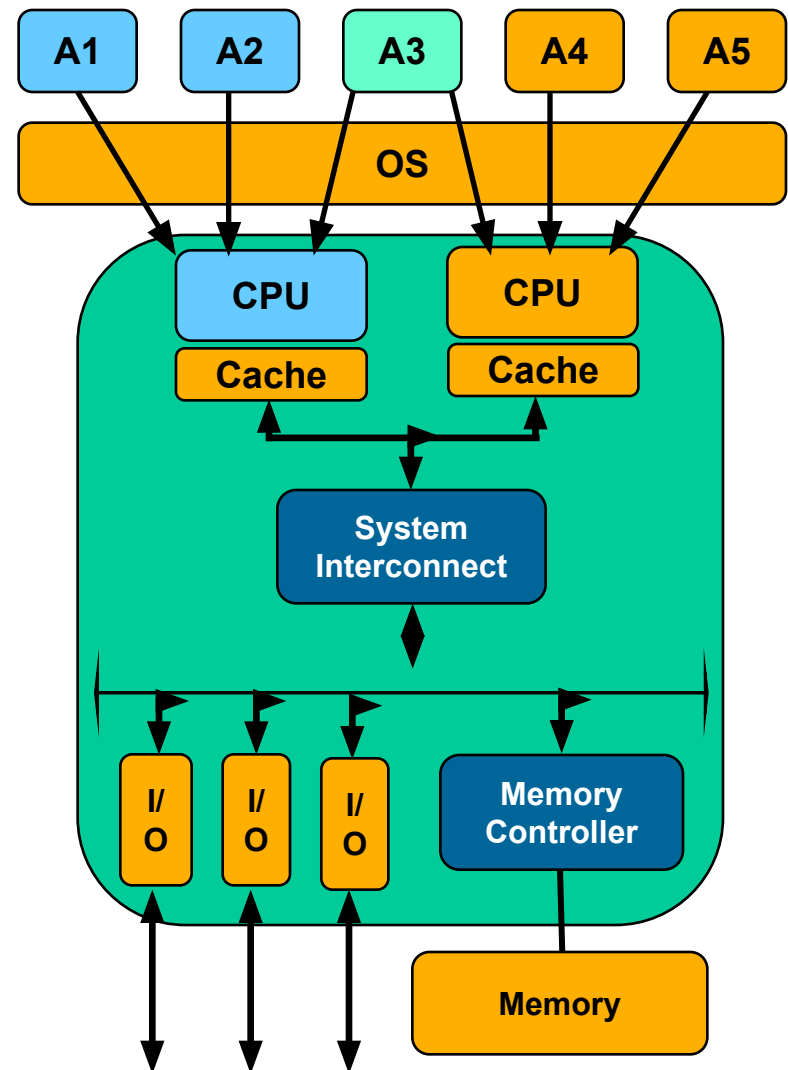
- Обычное приложение будет работать в multi-core варианте без каких-либо модификаций
- Нет необходимости проверять или переделывать существующий код для обработки вопросов параллельности
- Приложения могут работать как полностью симметричном режиме так и в bound режиме на одной системе

□ 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 приложения

Лучшее из обеих моделей

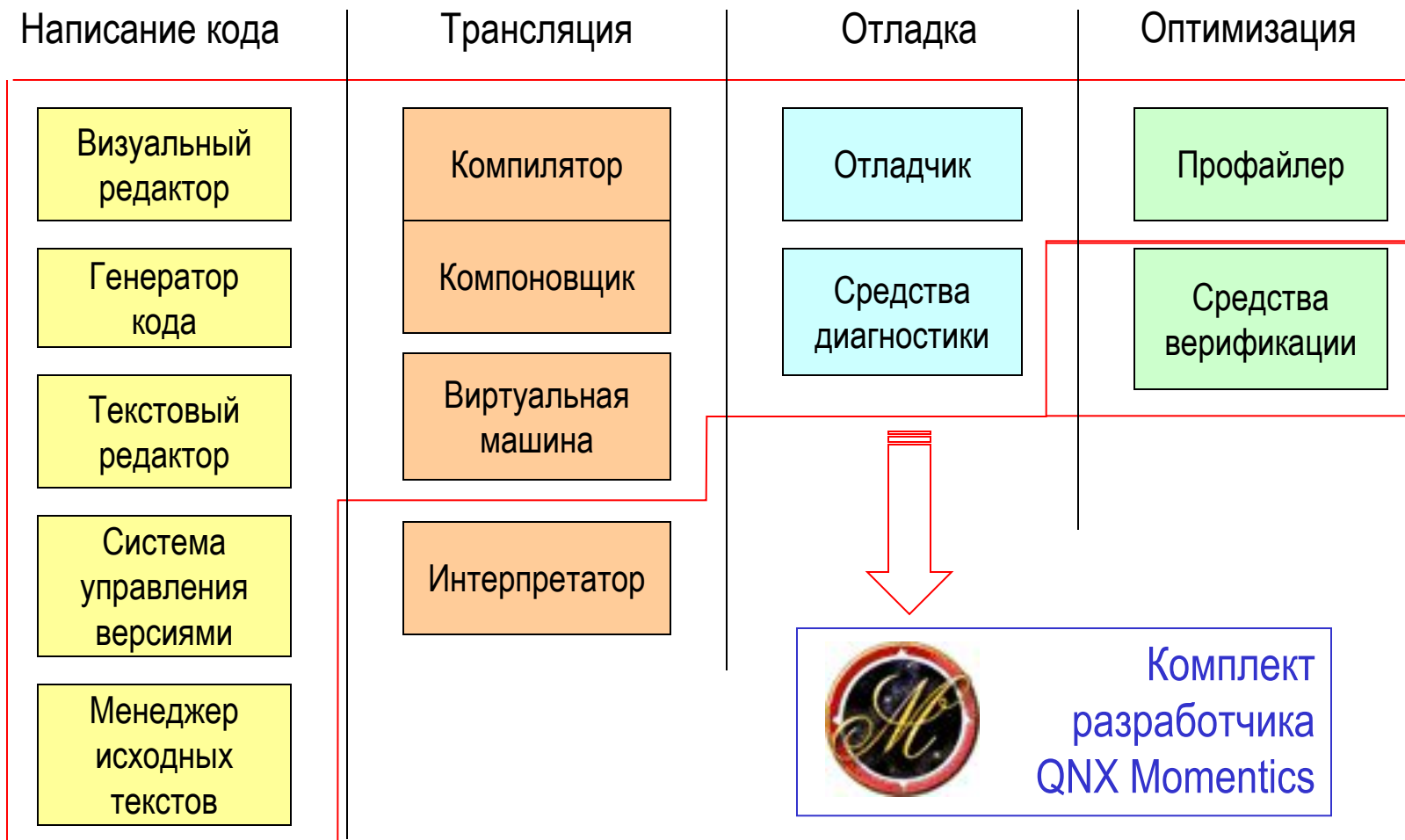
- ▣ *Bound Multiprocessing* собирает в себе лучшее из симметричной и ассиметричной моделей
- ▣ Поддерживает существующие и оптимизированные для multi-core приложения
- ▣ Разработчик имеет полный контроль над приложениями
- ▣ Балансировка нагрузки
 - Как автоматическая на уровне ОС, так и настраиваемая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





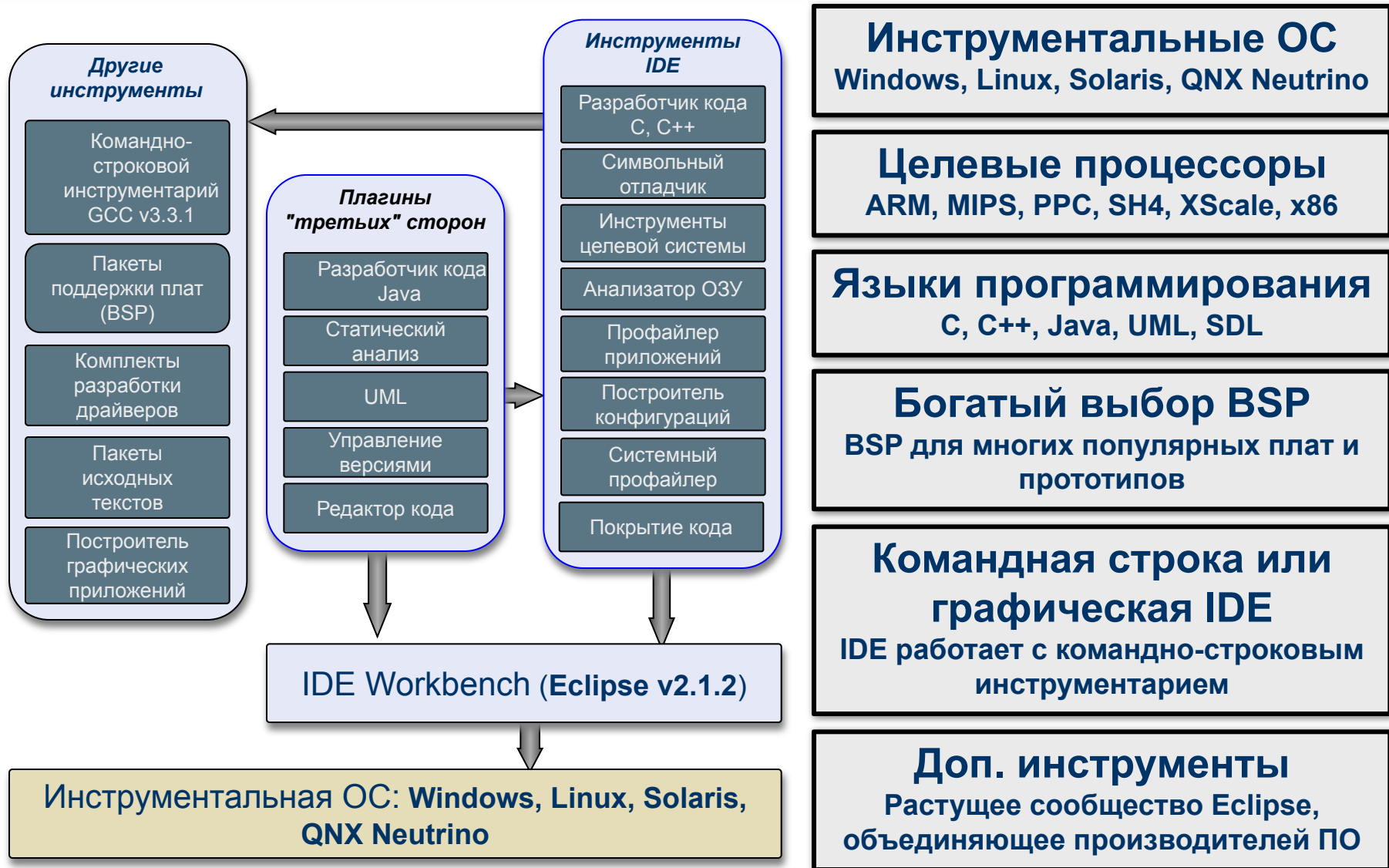
Комплект разработчика QNX Momentics

Цикл разработки





Вкратце о QN X 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 профилирование
 - 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



- 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

Идентификация ключевых слов, синтаксиса и парных скобок с первого взгляда

Закладки и задачи

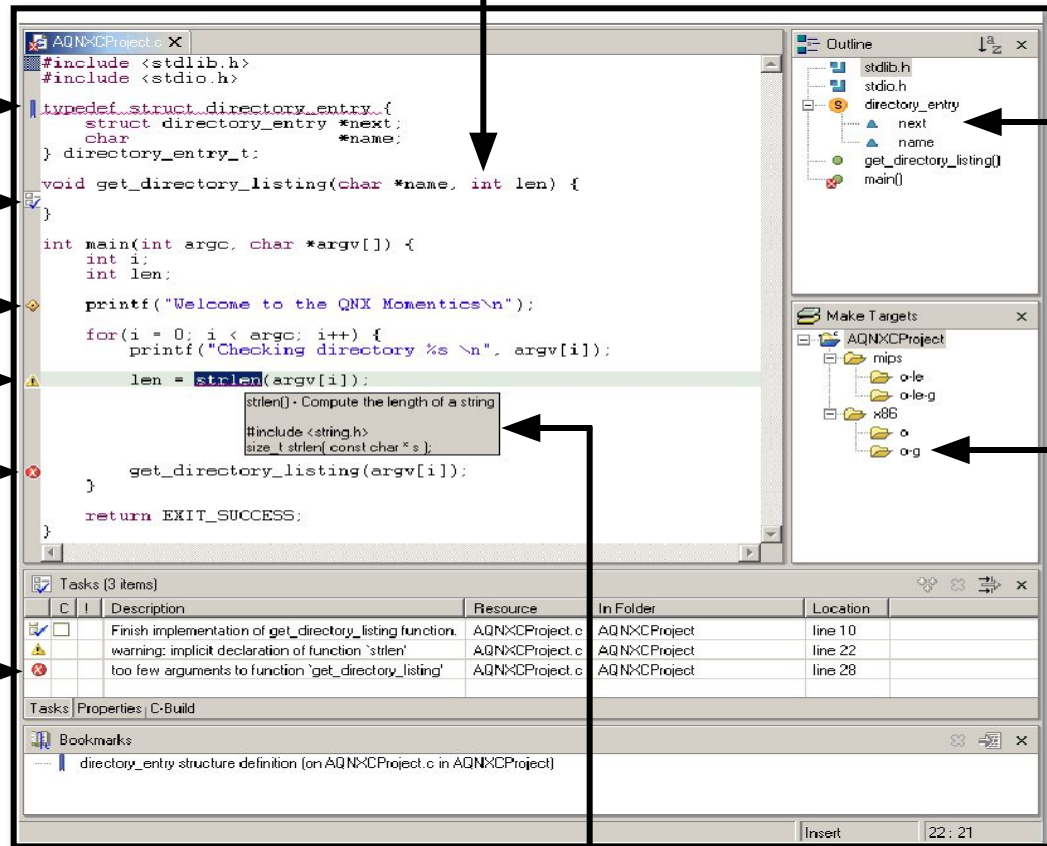
Задание точек останова перед компиляцией

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



The screenshot shows the QNX IDE interface with the following components:

- Main Editor:** Displays a C program named `AQNXCPProject.c`. The code includes `<stdlib.h>` and `<stdio.h>`, defines a `directory_entry_t` structure, and implements a `get_directory_listing` function. A tooltip for `strlen` is visible over the line `len = strlen(argv[i]);`.
- Outline Window:** Located at the top right, it shows a hierarchical list of identifiers: `stdlib.h`, `stdio.h`, `directory_entry` (with sub-items `next` and `name`), `get_directory_listing()`, and `main()`.
- Make Targets Window:** Located below the Outline window, it shows a tree structure for building the project, including `AQNXCPProject`, `mips` (with sub-items `o-le` and `o-le-g`), `x86` (with sub-item `o`), and `o-g`.
- Tasks Window:** Located at the bottom left, it lists three items:

C	I	Description	Resource	In Folder	Location
☒		Finish implementation of get_directory_listing function.	AQNXCPProject.c	AQNXCPProject	line 10
☐		warning: implicit declaration of function 'strlen'	AQNXCPProject.c	AQNXCPProject	line 22
☐		too few arguments to function 'get_directory_listing'	AQNXCPProject.c	AQNXCPProject	line 28
- Bookmarks Window:** Located at the bottom, it shows a bookmark for `directory_entry` structure definition (on AQNXCPProject.c in AQNXCPProject).

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

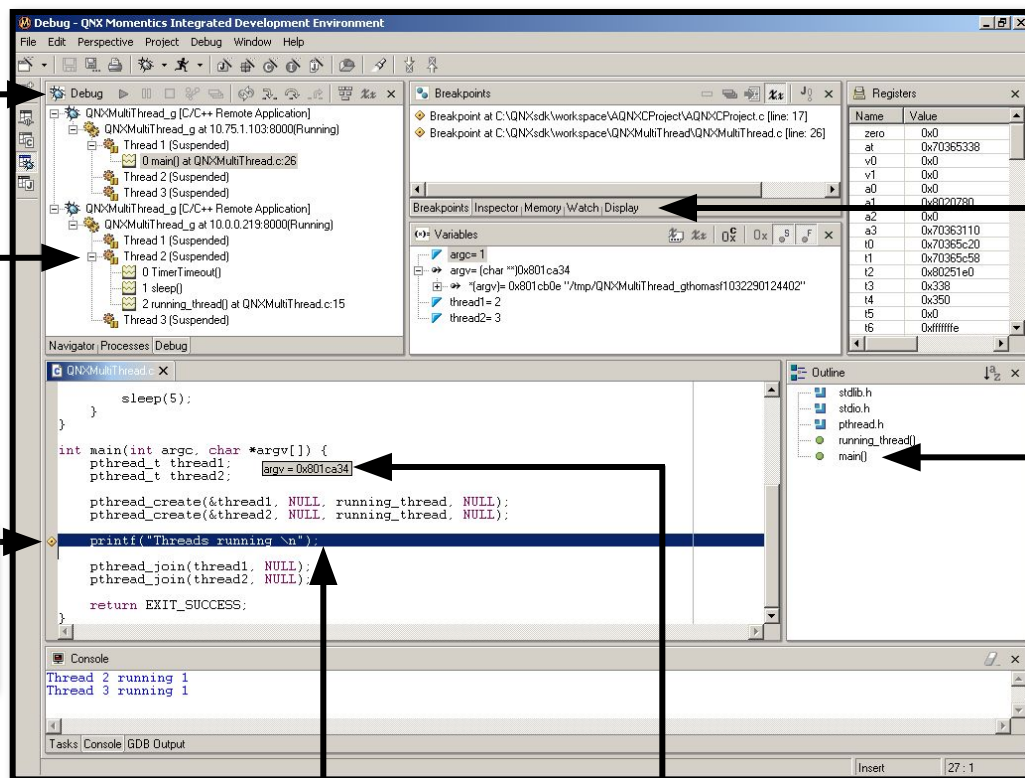


QNX IDE: символьный отладчик + анализатор ОЗУ

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

QNX System Information - QNX Momentics Integrated Development Environment

File Edit Perspective Project Window Help

System Summary

Hostname: thomasf2 Board: x86pc

OS Version: 6.2.1 (2002/11/05-01:46:41est)

Boot Date: Wed Dec 31 16:00:00 PST 1969

1 x86 cpu
x86 @ 451Mhz

System Memory: 28M/383M

Processes (42)

Applications				Servers		
Name	Heap	CPU	Start	Name	Heap	CPU
(unavailable)	0	999us	Tue Nov 25 0...	tinit	16K	6ms
pdebug	16K	1sec 142...	Mon Nov 25 0...	mqueue	16K	5ms
ksh	48K	21ms	Mon Nov 25 0...	pci-bios	16K	29ms
sh	48K	10ms	Mon Nov 25 0...	spooler	16K	14sec 2...
ksh	48K	11ms	Mon Nov 25 0...	dhcp.client	16K	10ms

System Summary | Malloc Information

Memory of devb-fdc

Process Map

0x08048000 Program 0x0804EFFF

Name	VAddr	Size	Map	Offset
Stack		28K		
Program		24K/4K		
devb-fdc code/text	0x08048000	24K	Shared Read/Execute	0x00000000
devb-fdc data/bss	0x0804e000	4K	Private Read/Write	0x00005000
Heap		200K		
Objects		0		
Library		476K...		
ldqx.so.2 code/text	0xb0300000	312K	Shared Read/Execute	0x00448000
ldqx.so.2 data/bss	0xb034e000	16K	Private Read/Write/Execute	0x00495000

Process devb-fdc

Arguments
devb-fdc cam quiet blk auto=partition.cache=100k

Threads

Thread	Priority	Name	State
1	10o	SigWaitInfo	
2	21r	Receive	

Environment
=/sbin/enum-devices
PATH=/sbin:/usr/sbin:/bin:/usr/bin

Owner
UID: 0 GID: 0
EUID: 0 EGID: 0

Thread Information

Process	Thread	Algorithm	Stack Space	CPU Time	Start Time
devb-fdc	1	Other	516K	11ms	Mon Nov 25 0...
devb-fdc	2	RR	12K	999us	Mon Nov 25 0...
devb-fdc	3	Other	16K	3ns	Mon Nov 25 0...
devb-fdc	4	Other	16K	9ns	Mon Nov 25 0...
devb-fdc	5	Other	16K	3ns	Mon Nov 25 0...
devb-fdc	6	Other	16K	3ns	Mon Nov 25 0...

System Blocking Graph

■ Servicing request ■ Waiting for request ■ Waiting for reply ■ Waiting for service

devb-fdc Channel 1

devb-fdc thread 3

devb-fdc thread 4

devb-fdc thread 5

devb-fdc thread 6

Object

Object	Object Owners	State	Blocked ...	S...
devc-ser8250 Channel 1	devc-ser8250 : 1	Receive	pdebug : 1	R...
devb-fdc Channel 1	devb-fdc : 2	Receive		
	devb-fdc : 3	Receive		
	devb-fdc : 4	Receive		

Используй-
вание
процессора
и "кучи"
процессами

Используй-
вание
памяти
конкретным
процессом

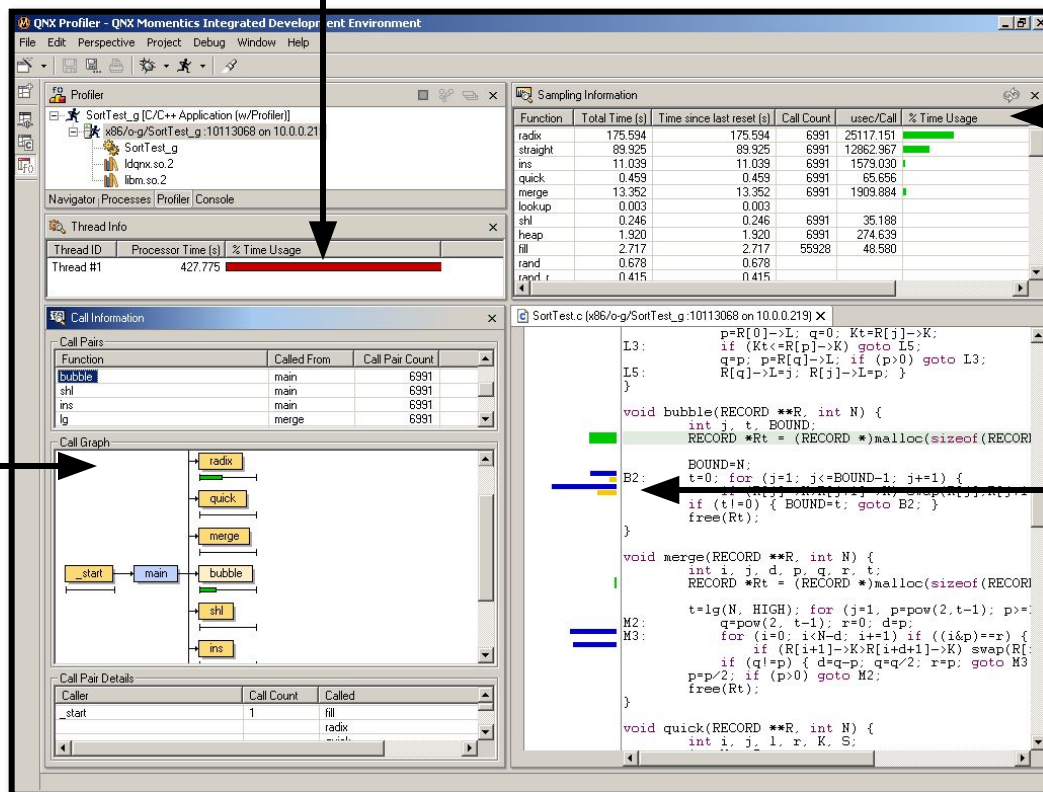
Просмотр
окружения
для каждого
процесса

Сортировка и
анализ потоков
по различным
атрибутам

Просмотр
отношений
блокирования

Определение наиболее загруженных потоков

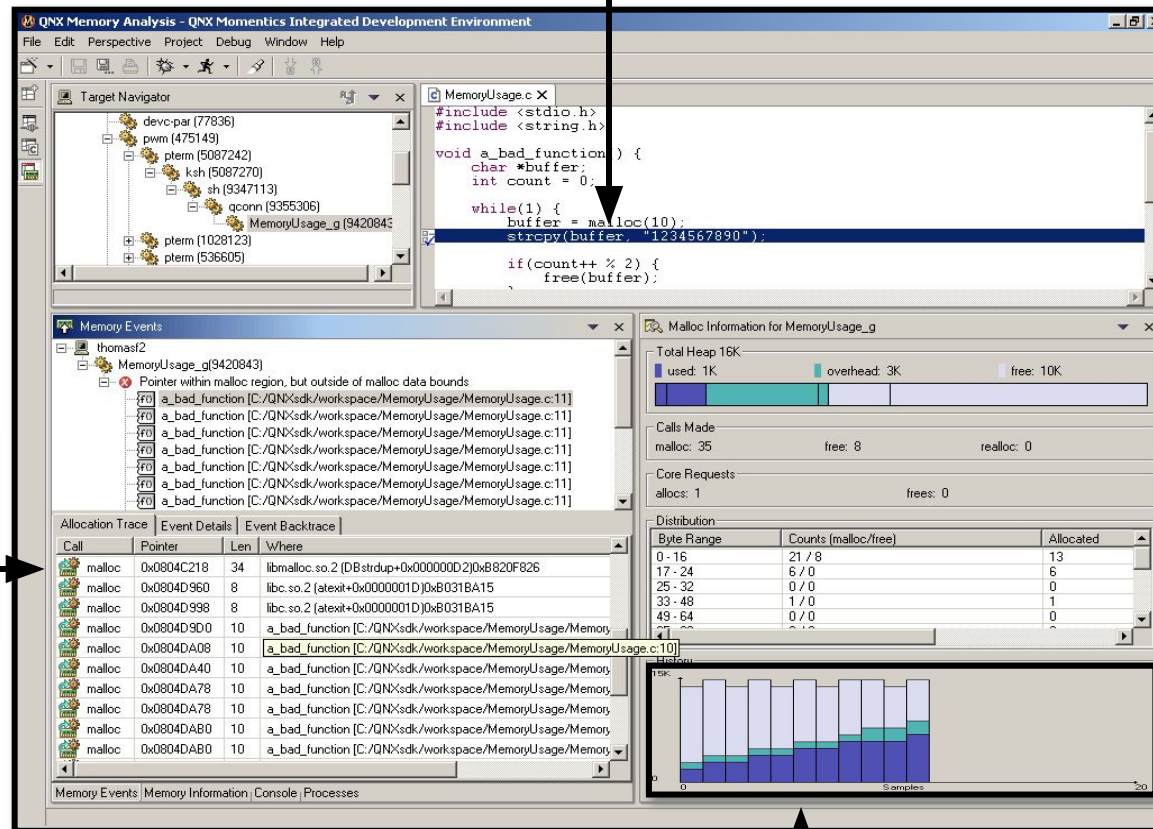
Сортировка результатов по общему времени, процентной доле от общего времени, числу вызовов и т.п.



Дерево вызовов помогает сразу же оценить динамическую структуру выполнения приложений

Определение строк кода, потребляющих наибольшее количество процессорного времени

Локализация переполнения буферов и запуск отладчика



The screenshot displays the QNX Memory Analysis tool within the QNX Momentics IDE. The interface includes several key components:

- Target Navigator:** Shows a tree view of the target system components, including 'devc-par (77836)', 'pwm (475149)', 'pterm (5087242)', 'ksh (5087270)', 'sh (9347113)', 'qconn (9355306)', 'MemoryUsage_g (9420843)', 'pterm (1028123)', and 'pterm (536605)'.
- Code Editor:** Displays the source code for 'MemoryUsage.c'. The function 'a_bad_function' is highlighted, showing a buffer overflow scenario where 'strcpy(buffer, "1234567890");' is called on a buffer of size 10.
- Memory Events:** A list of memory events, including 'Pointer within malloc region, but outside of malloc data bounds'.
- Allocation Trace:** A table showing memory allocation events with columns for Call, Pointer, Len, and Where.
- Memory Information:** A panel showing 'Total Heap 16K', 'used: 1K', 'overhead: 3K', and 'free: 10K'. It also includes a 'Distribution' table and a 'Histogram' chart.

Отслежива-
ние операций
распреде-
ления 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 сессии
"живые" результаты
бинарного покрытия
вплоть до отладки
функций

Отладка:
просмотр
запущенных
процессов

Code Coverage Sessions

- server_demo [GCC Code Coverage] (1/26/04 8:28 AM)
 - server_demo [32.56%]
 - server.c [32.56%]
 - handle_read [0%]
 - main [75.61%]

Code Coverage Sessions | Navigator

Debug

- server_demo [C/C++ QNX QConn (IP)]
- server_demo [GCC Code Coverage] (Running - idle)
- server_demo_g on tx86 (tx86p233) pid #300816 (1/26/04)

Properties

Property	Value
Coverage Info	
Lines Fully Covered	45.31% (29 lines)
Lines Not Covered	54.69% (35 lines)
Lines Partially Covered	0% (0 lines)
Total Coverage	32.56%
Info	
derived	false
editable	true
last modified	5/21/03 2:15 PM
linked	false
location	E:\QNX630\workspace\server_demo\server.c
name	server.c

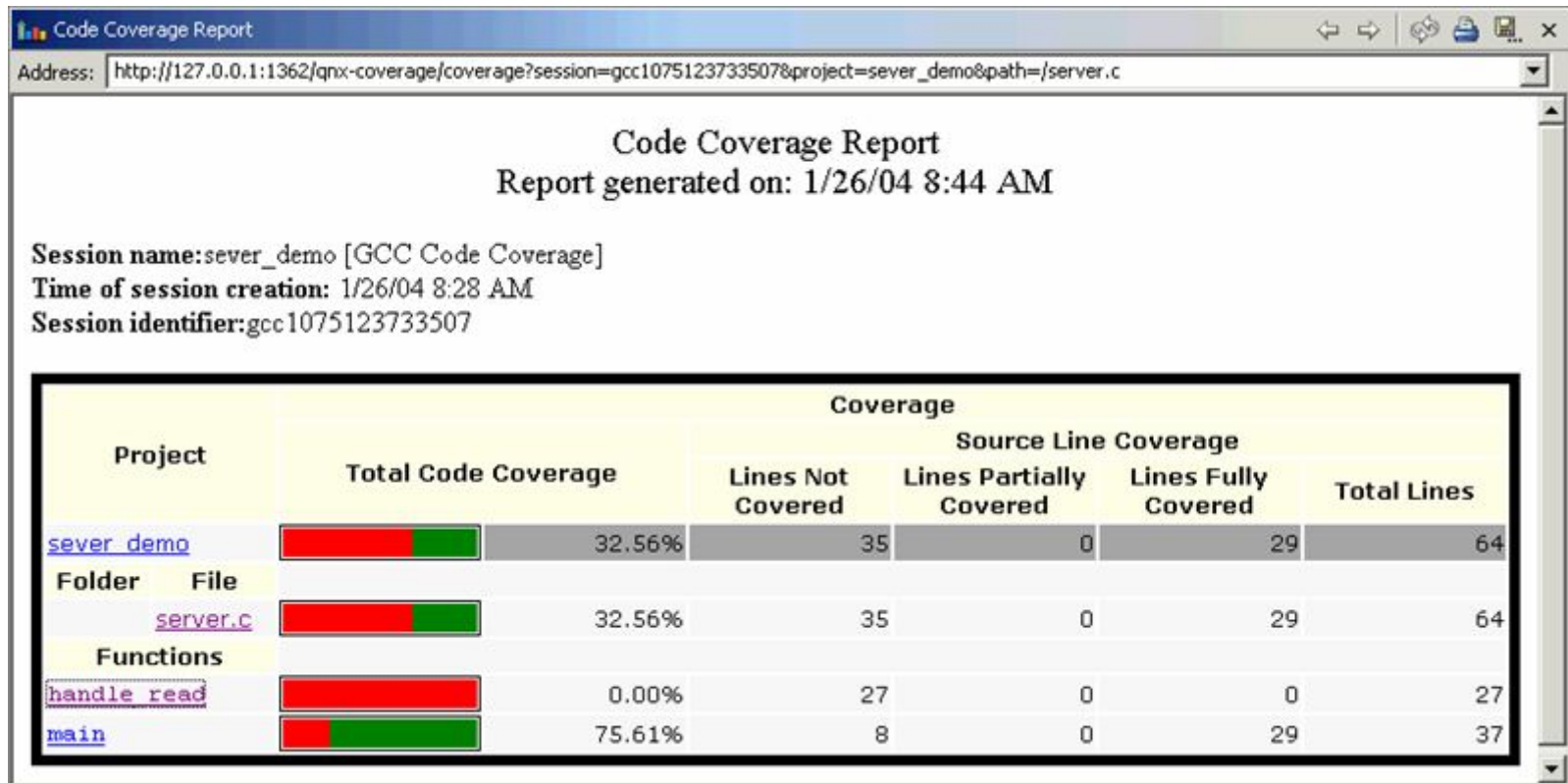
за сессии:
я оценка



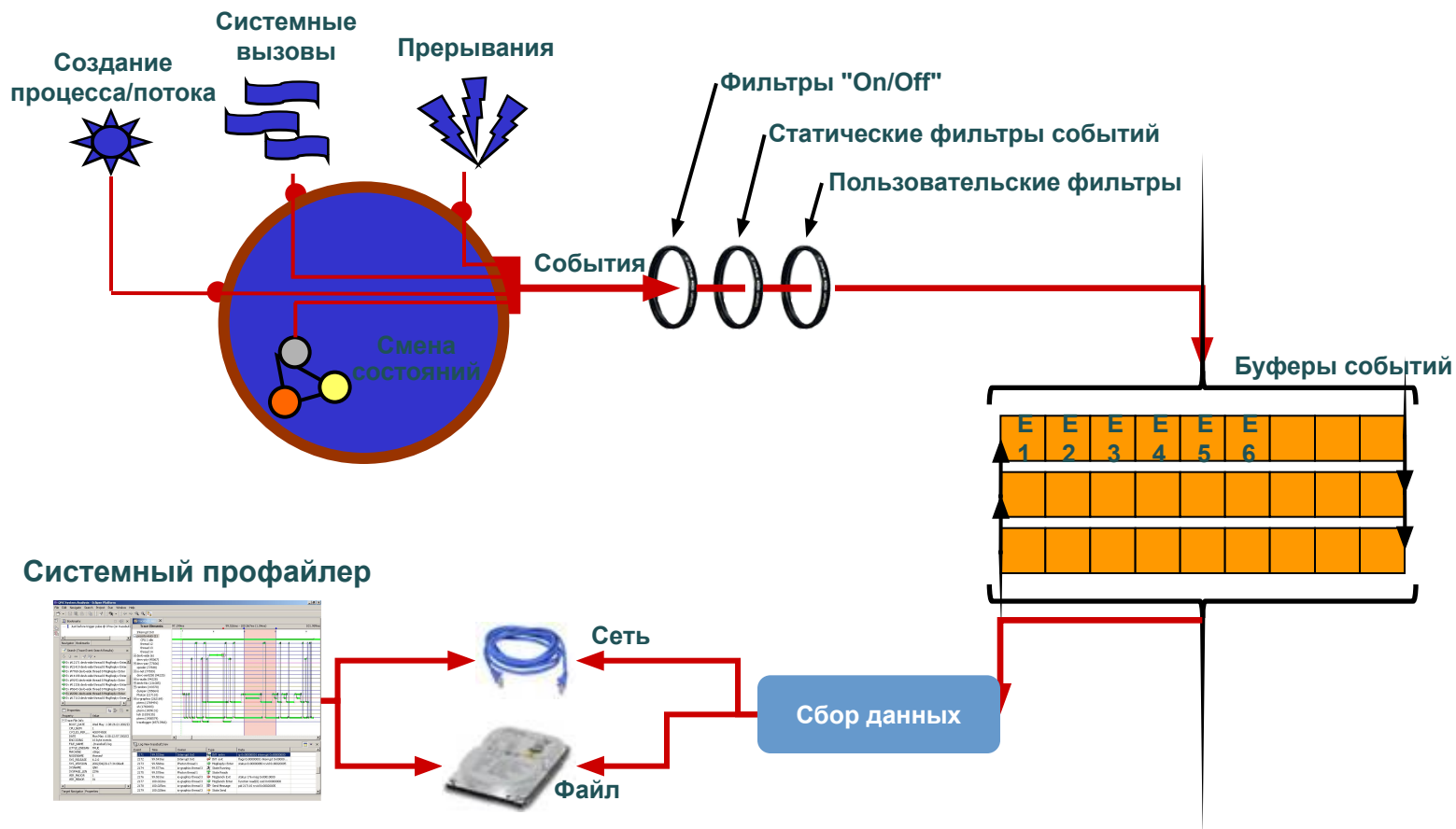
QNX IDE: отчет о покрытии кода

Генератор отчетов:

- Отчеты в формате HTML для дальнейшего анализа, по каждой сессии
- Статистика для контроля качества

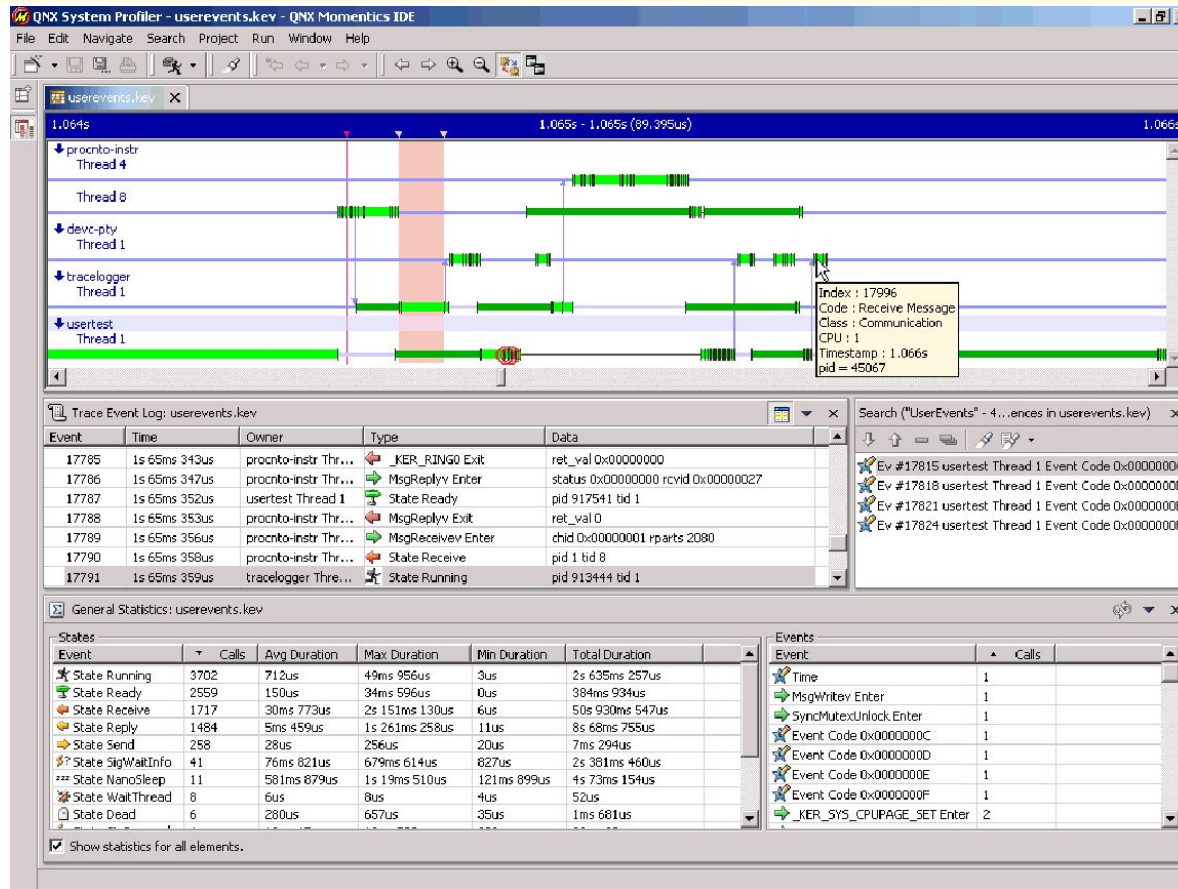


Диагностическая версия ядра ведет журнал событий, фильтрует их и сохраняет в буферах, содержимое которых можно сохранять и анализировать





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...

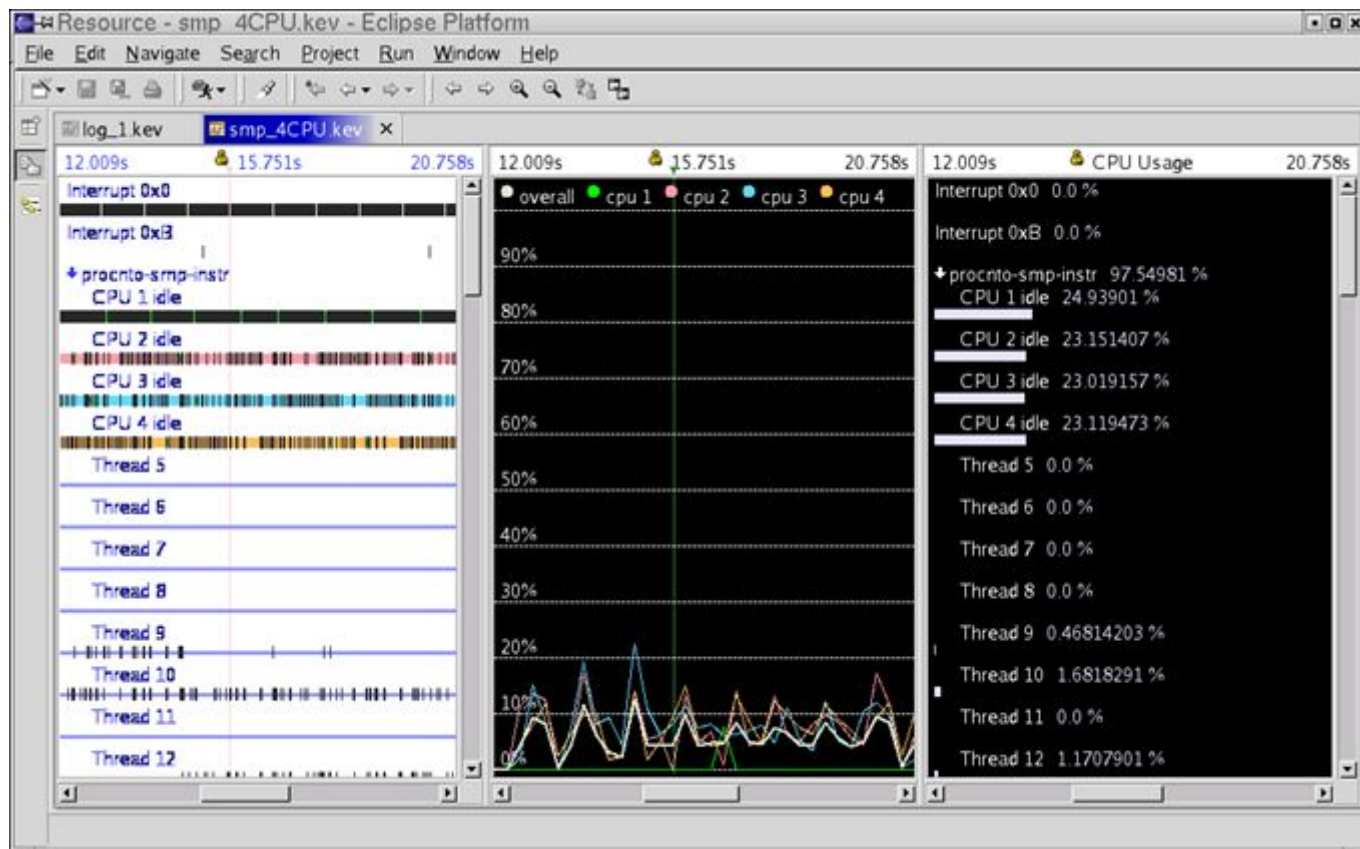
Фильтры пост-обработки переработаны с учетом расширяемости



Системный профайлер: представление "Активность CPU"

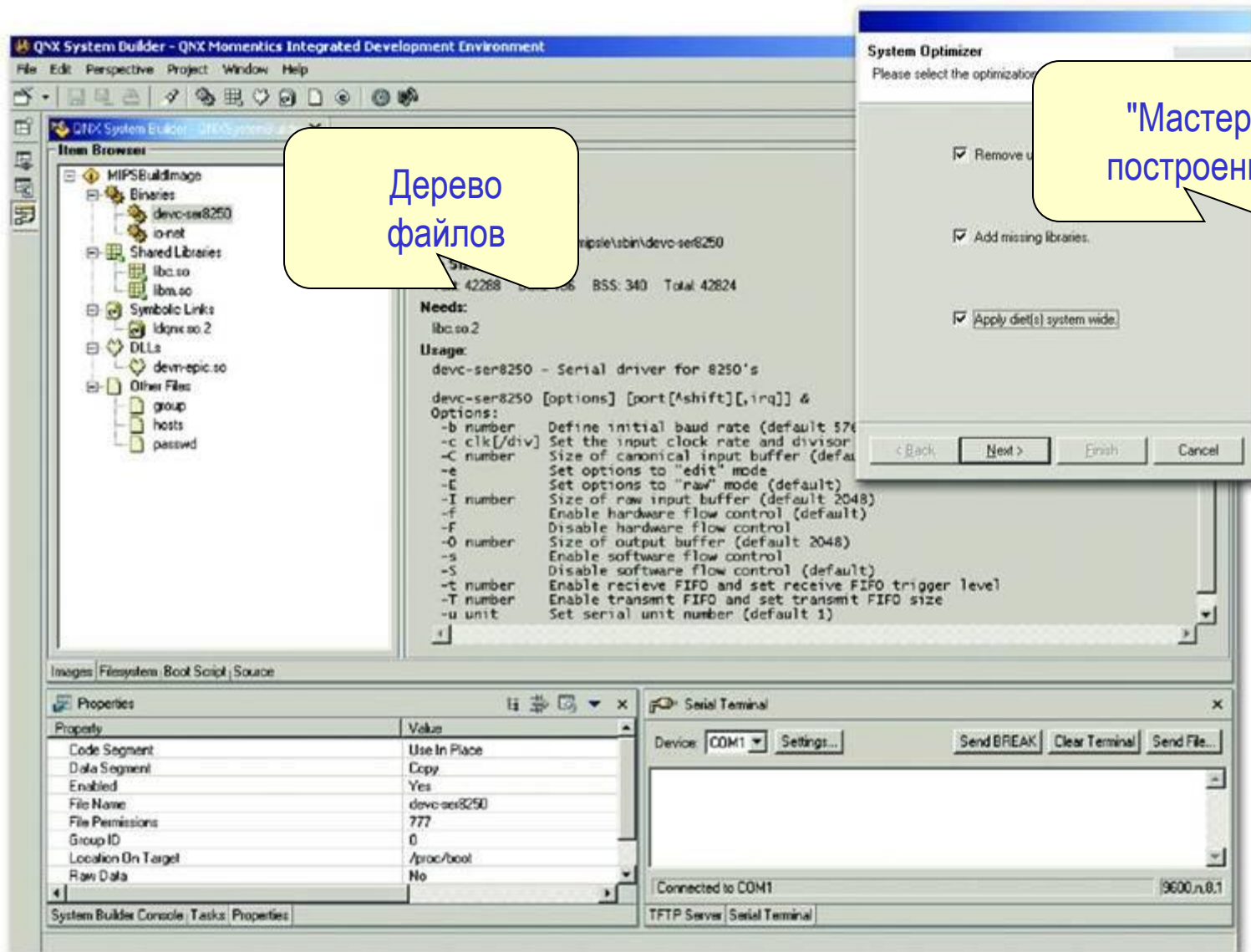
**% активности
CPU от общего
времени**

**Разбиение активности
CPU по элементам
трассировки**

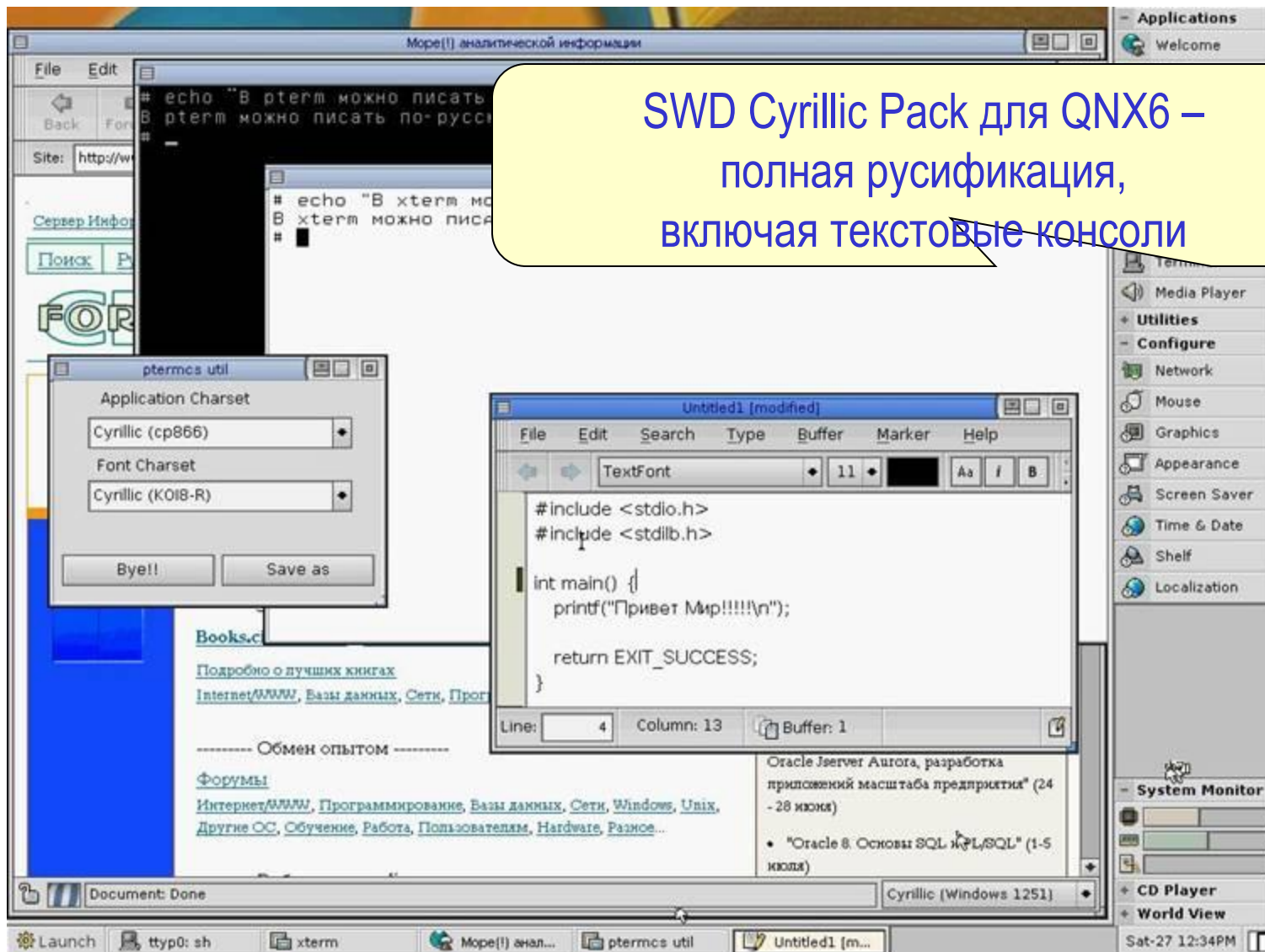




QNX IDE: построитель встраиваемых конфигураций



SWD Cyrillic Pack для QNX6 –
полная русификация,
включая текстовые консоли



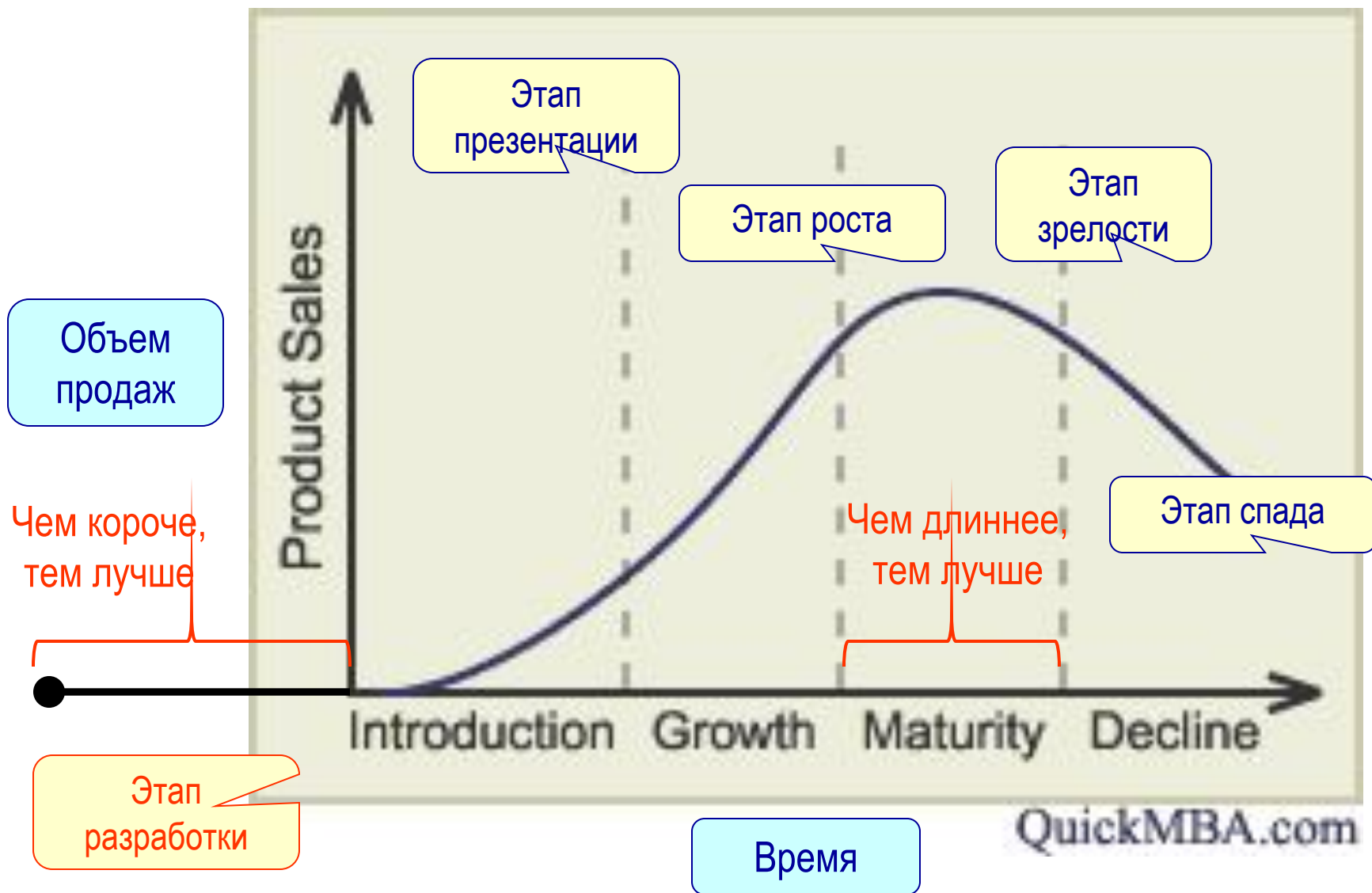
QNX помогает экономить

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

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

- QNX и сокращение суммарной стоимости владения (TCO)

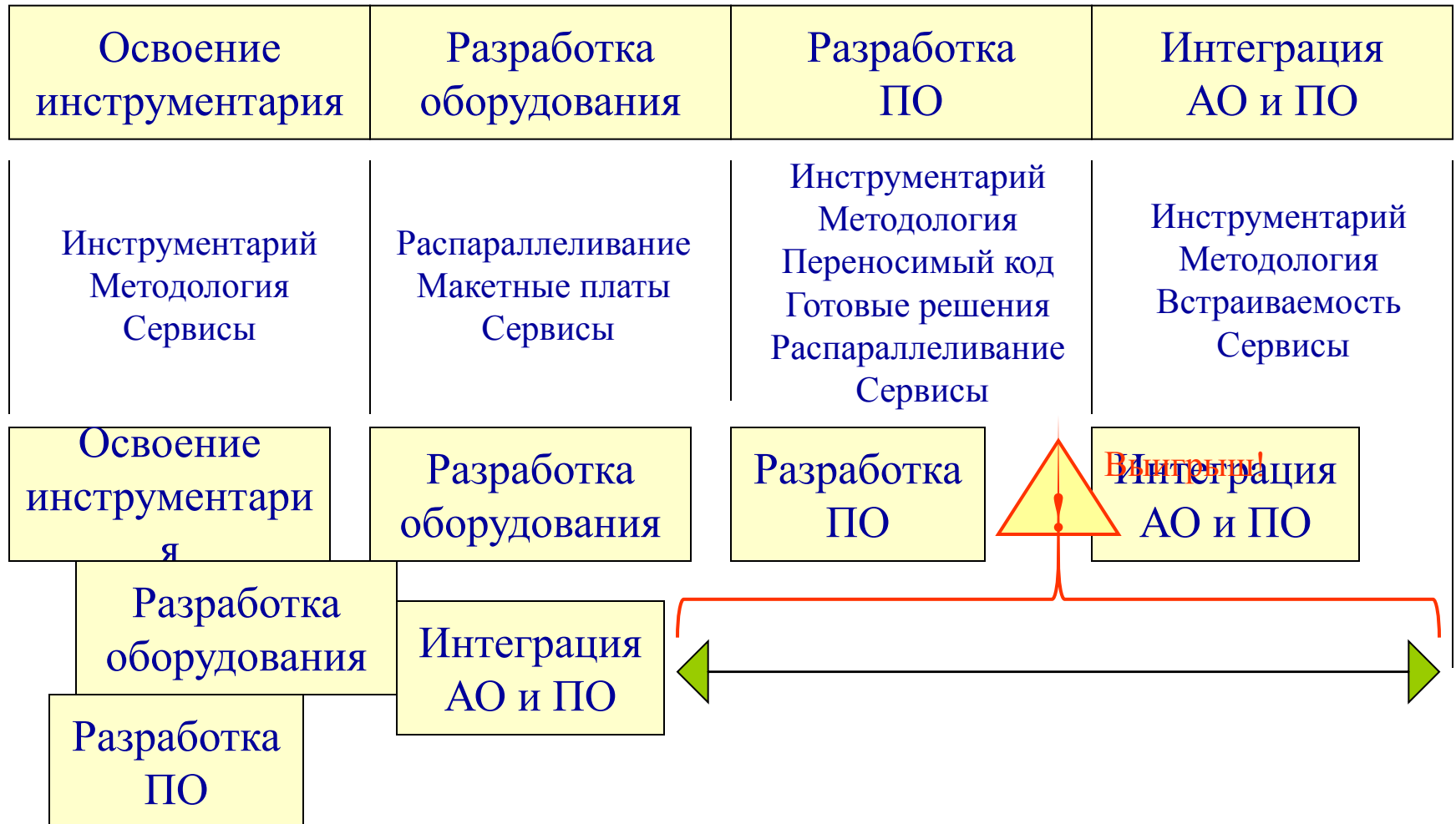
TCO = Total Cost of Ownership,
"суммарная стоимость владения"





Как ОС может сократить TTM?

Время





QNX как средство сокращения ТТМ

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

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



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