



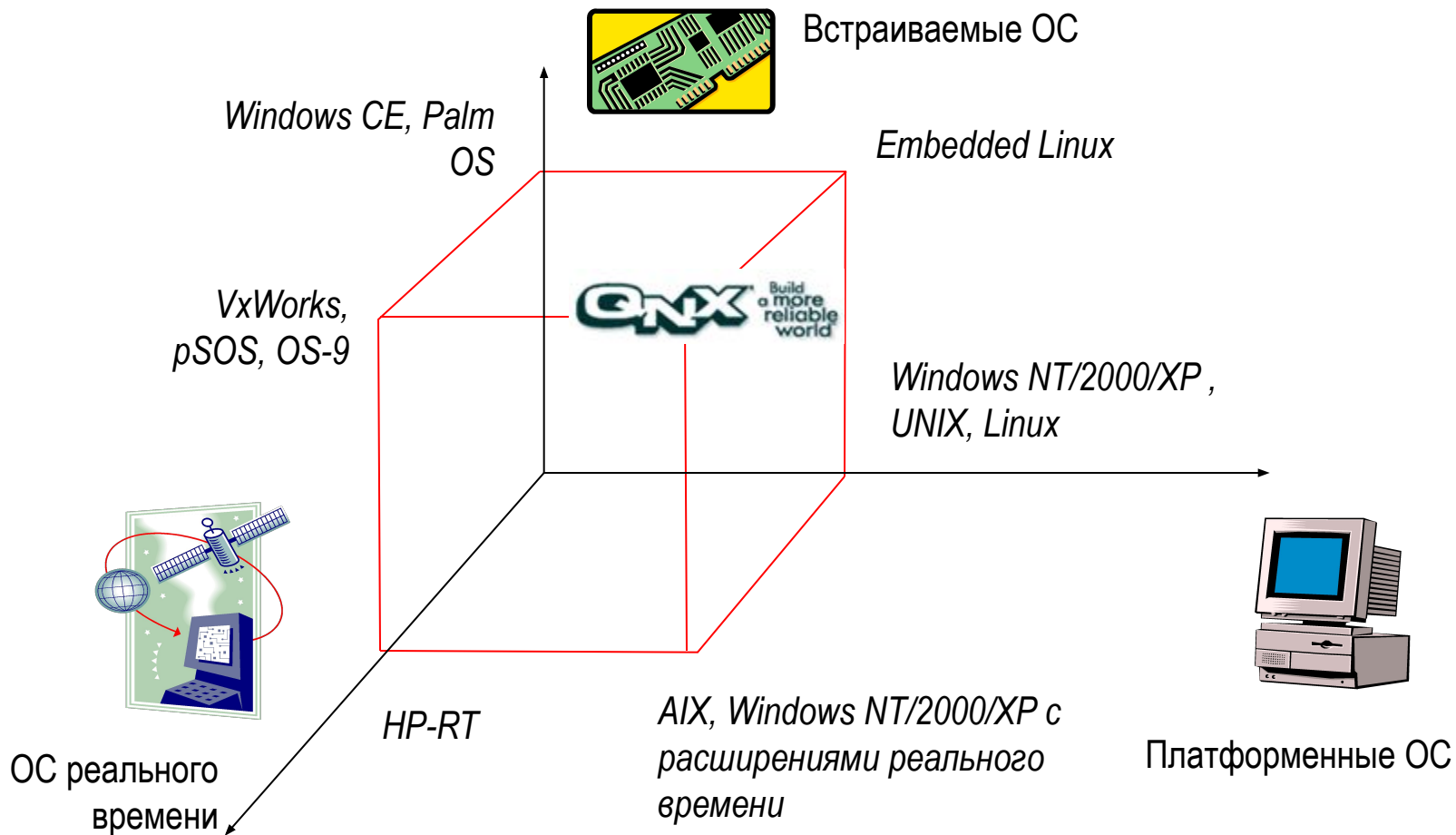
# ОС реального времени QNX и интегрированный комплект разработчика QNX Momentics

Александр Трофимов  
SWD Software Ltd.



OCPB QNX

# Чем QNX отличается от других ОС?

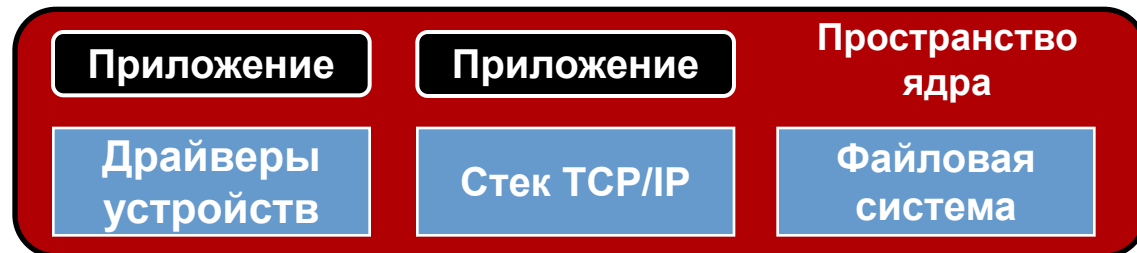




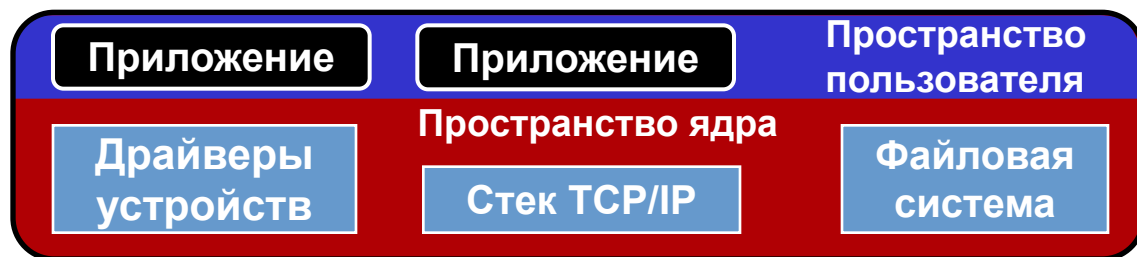
# QNX как ОС жесткого реального времени

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

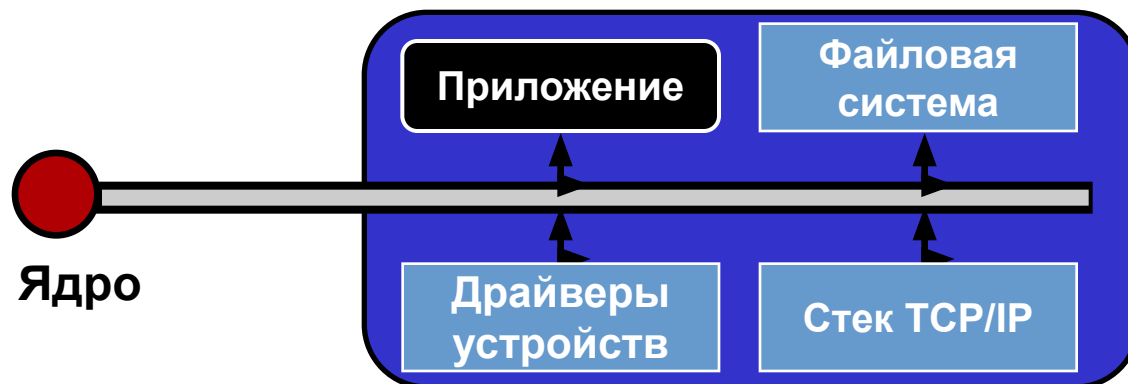
Исполняемый модуль  
реального времени  
(напр. VxWorks)



Монолитное ядро  
(напр. Linux)

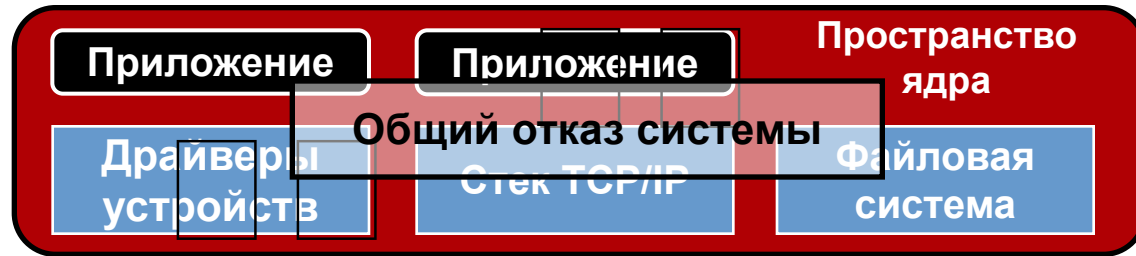


Микроядро  
(напр. QNX Neutrino)



## Исполняемый модуль реального времени

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



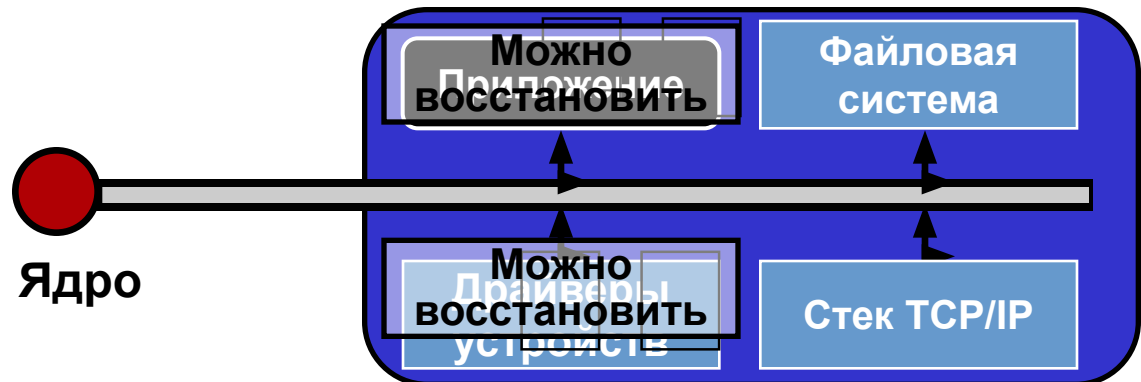
## Монолитное ядро (NT / Unix / и т.п.)

- > MMU, частичная защита
- > Защищены только приложения

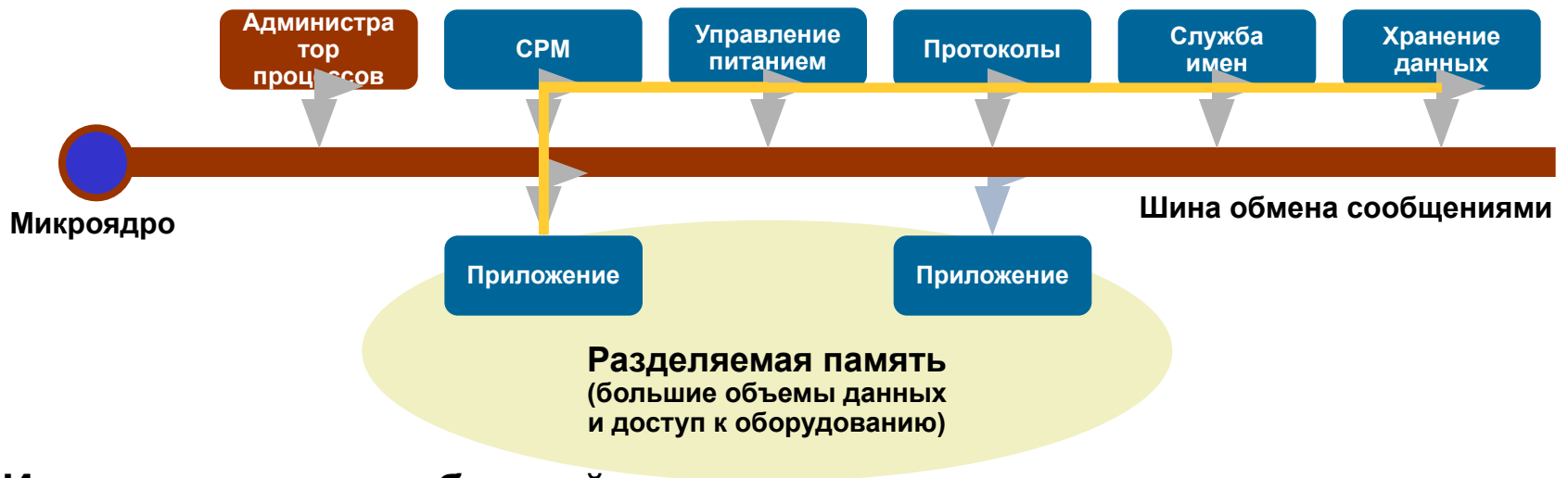


## Микроядро (QNX Neutrino)

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



## □ Задачи общаются посредством сообщений



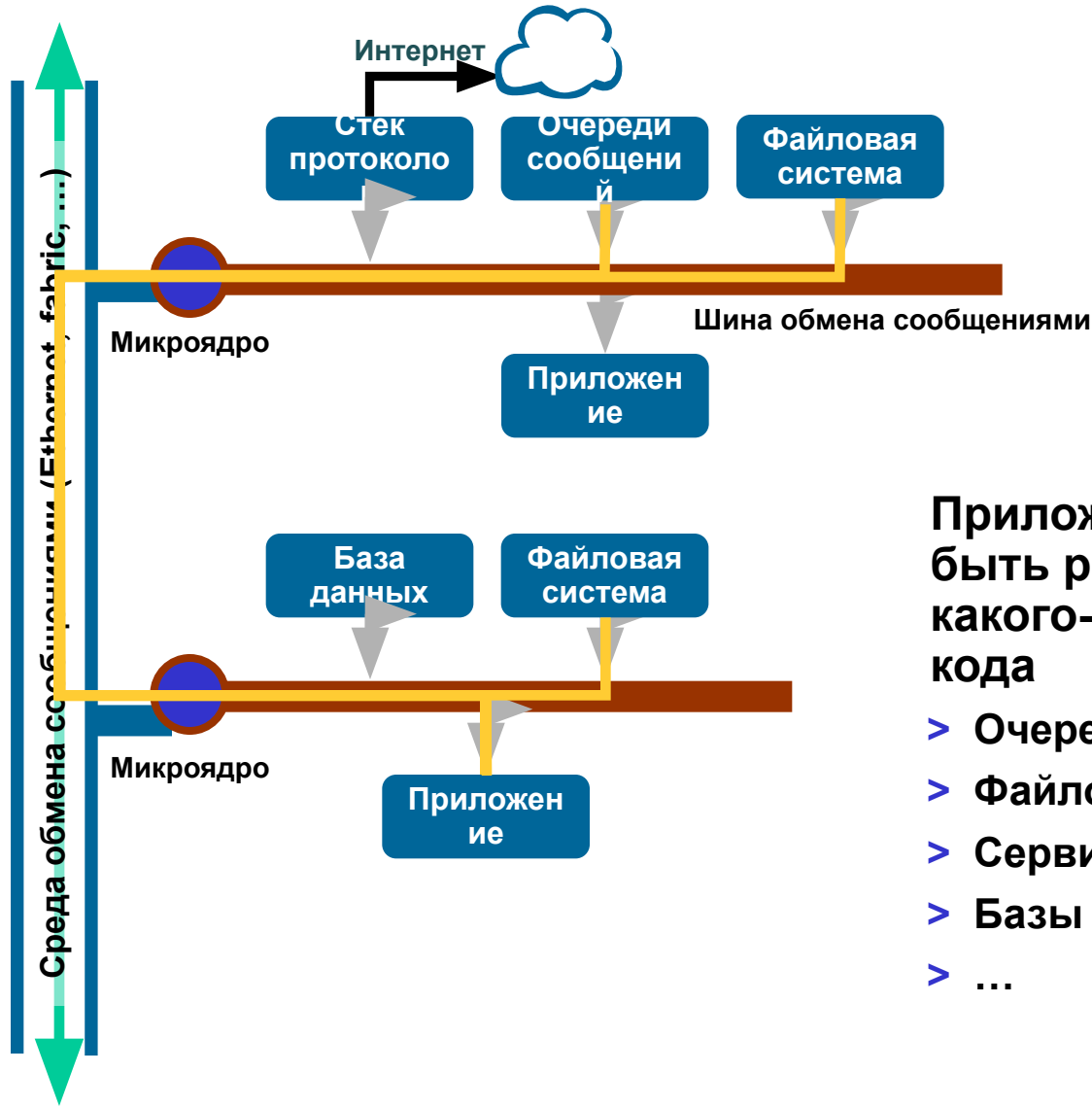
□ Использование сообщений органично "развязывает" задачи

□ Над сообщениями надстроены вызовы POSIX

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

□ ... и другие вызовы POSIX

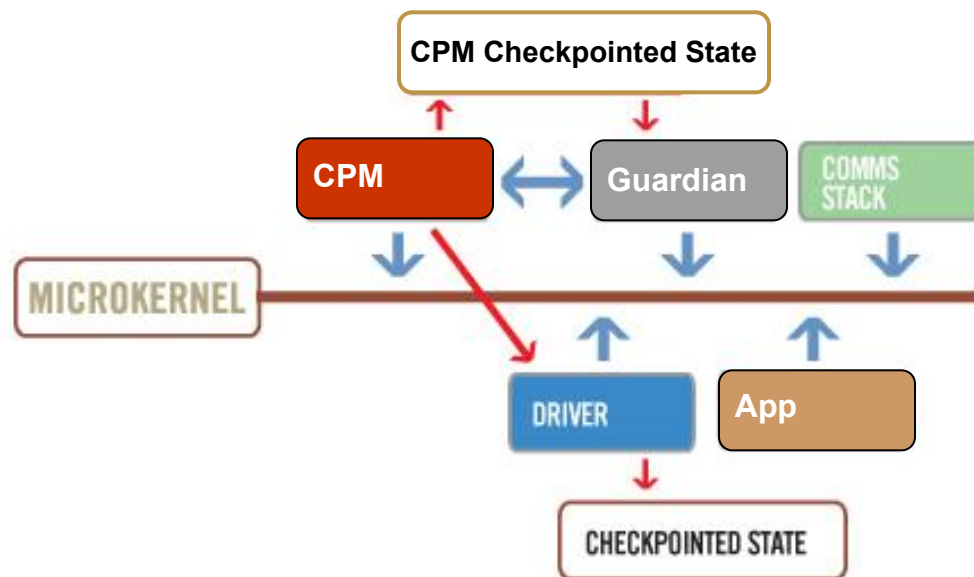
- realtime signals
- pipes and POSIX mqueues
- mutexs, condvars, semaphores
- barriers, sleepon
- reader/writer locks



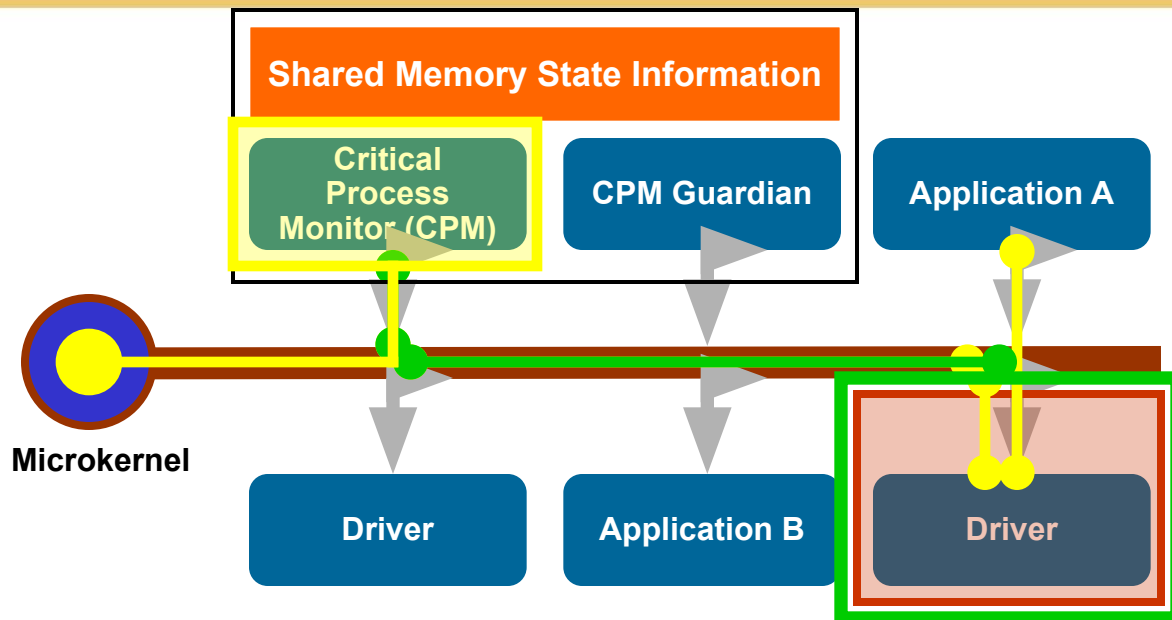
**Приложения/серверы могут быть распределенными без какого-либо специального кода**

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





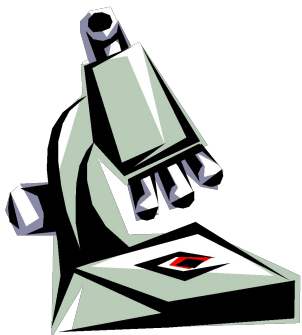
- **Основа системы высокой готовности – Монитор Ключевых Процессов (CPM). Выполняет мониторинг выбранных компонентов и обеспечивает восстановление после сбоев**
- **Процесс-охранник дублирует CPM**
- **Клиентская библиотека позволяет компонентам незамедлительно и прозрачно восстанавливать все соединения**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



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



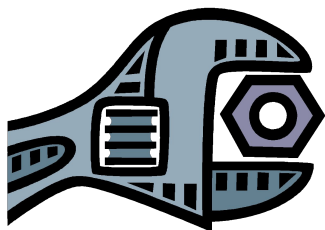
# QNX как встраиваемая и масштабируемая ОС



- **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



- **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 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



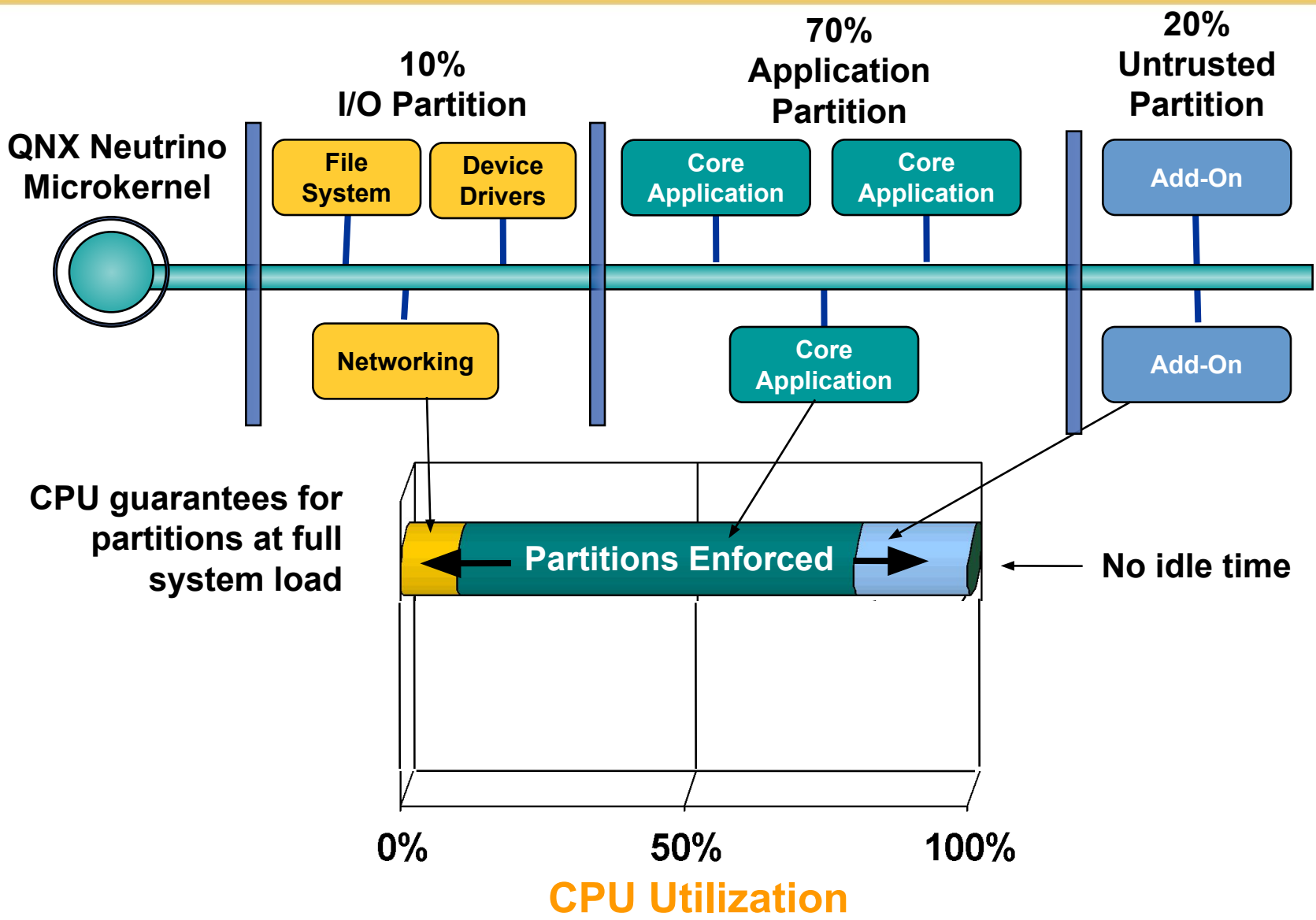
# Зачем нужна Adaptive Partitioning?

- **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 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**

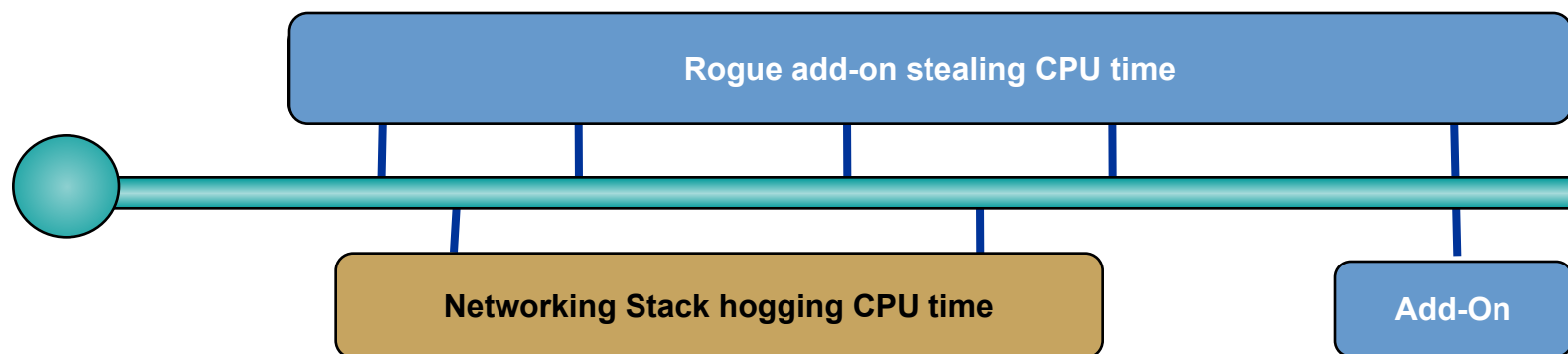




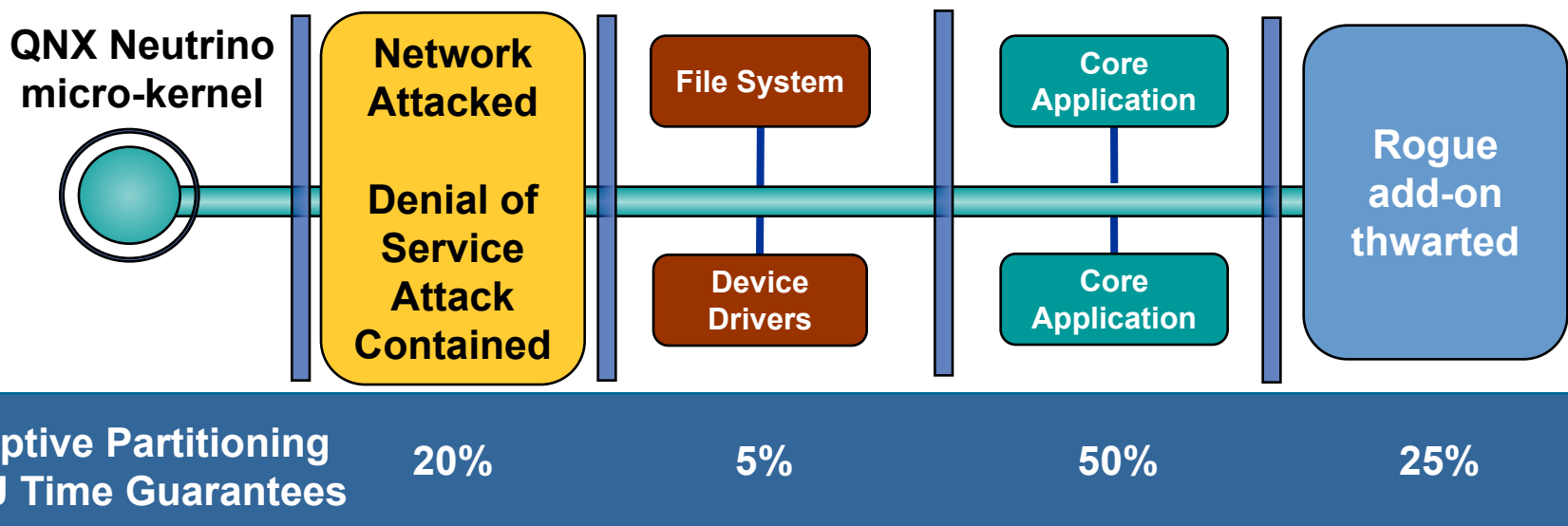
# Максимизация производительности



- **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**



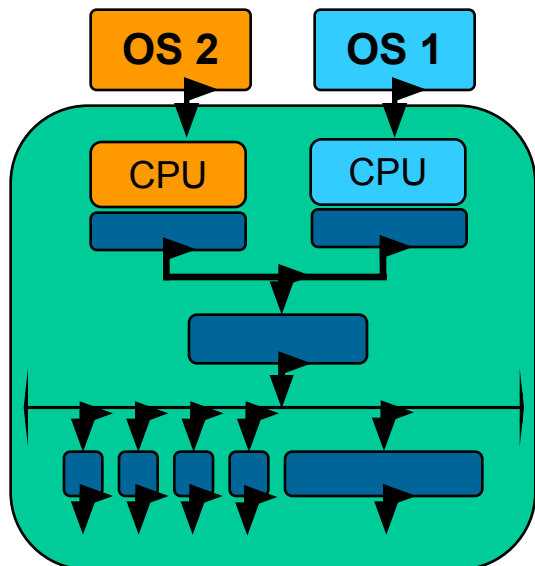
- **Создайте разделы для защиты критических системных ресурсов**
  - Гарантия ресурсов CPU для базовых функций
  - Наследование партиций гарантирует время CPU всем сервисам ОС (драйвера, файловые системы, сетевая система)
- **Защита основные приложений от угроз**
  - Минимизация влияние вредоносного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





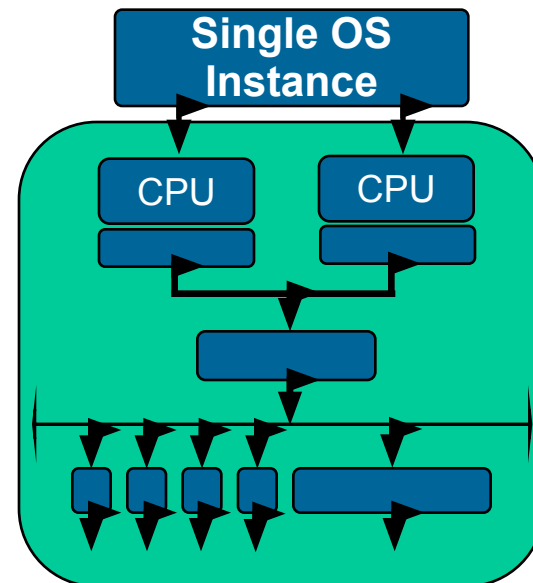
- **QNX Neutrino Multi-core Technology Development Kit**  
единственная в отрасли полнофункциональная платформа для нового поколения multi-core кристаллов
- **При помощи QNX Neutrino Multi-Core Technology Development Kit вы сможете:**
  - Быстро перенести приложения для однопроцессорной архитектуры на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



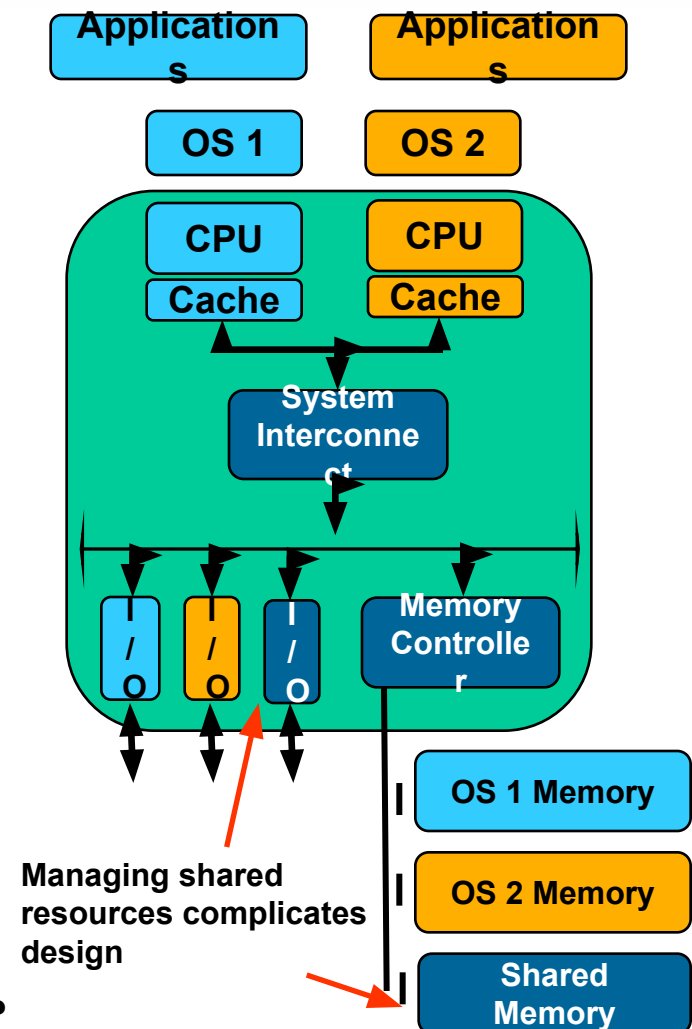
- 2 ядра, одна ОС
- QNX, Linux

## Asymmetric Model – «3А»:

- 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

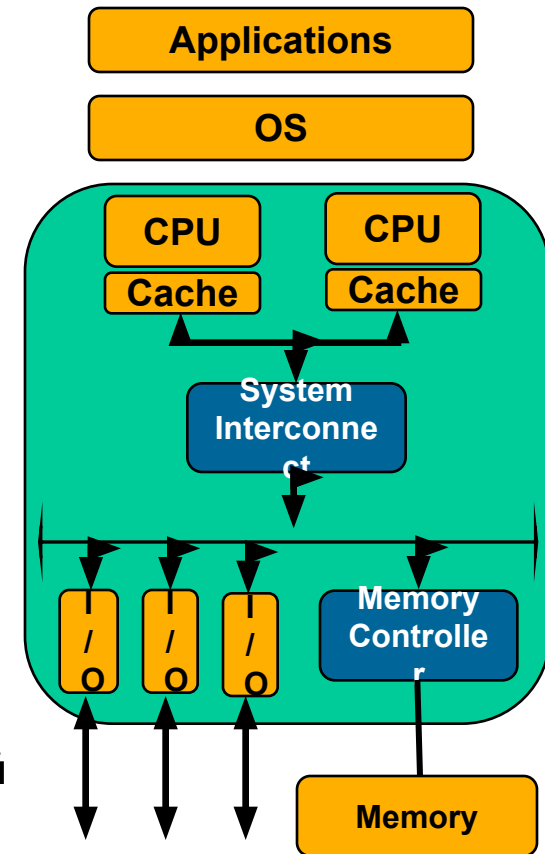


## □ Symmetric Model – «3А»:

- 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

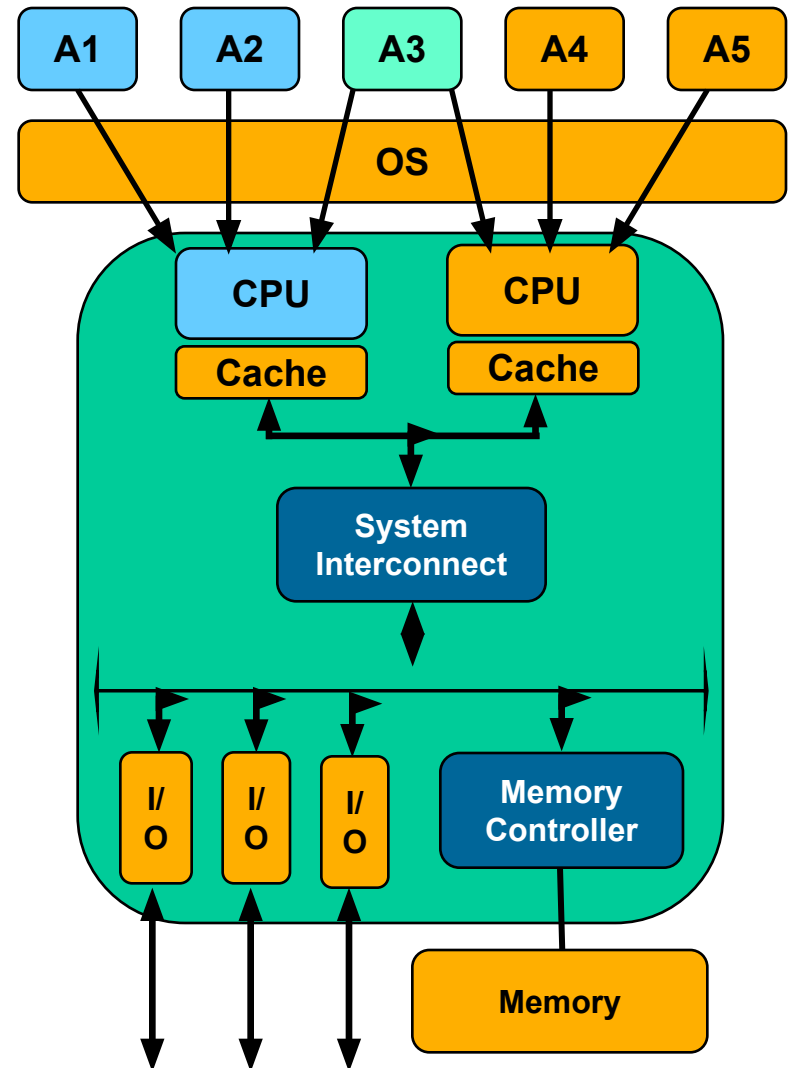
- Обычное приложение будет работать в multi-core варианте без каких-либо модификаций
- Нет необходимости проверять или переделывать существующий код для обработки вопросов параллельности
- Приложения могут работать как полностью симметричном режиме так и в bound режиме на одной системе

### □ 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

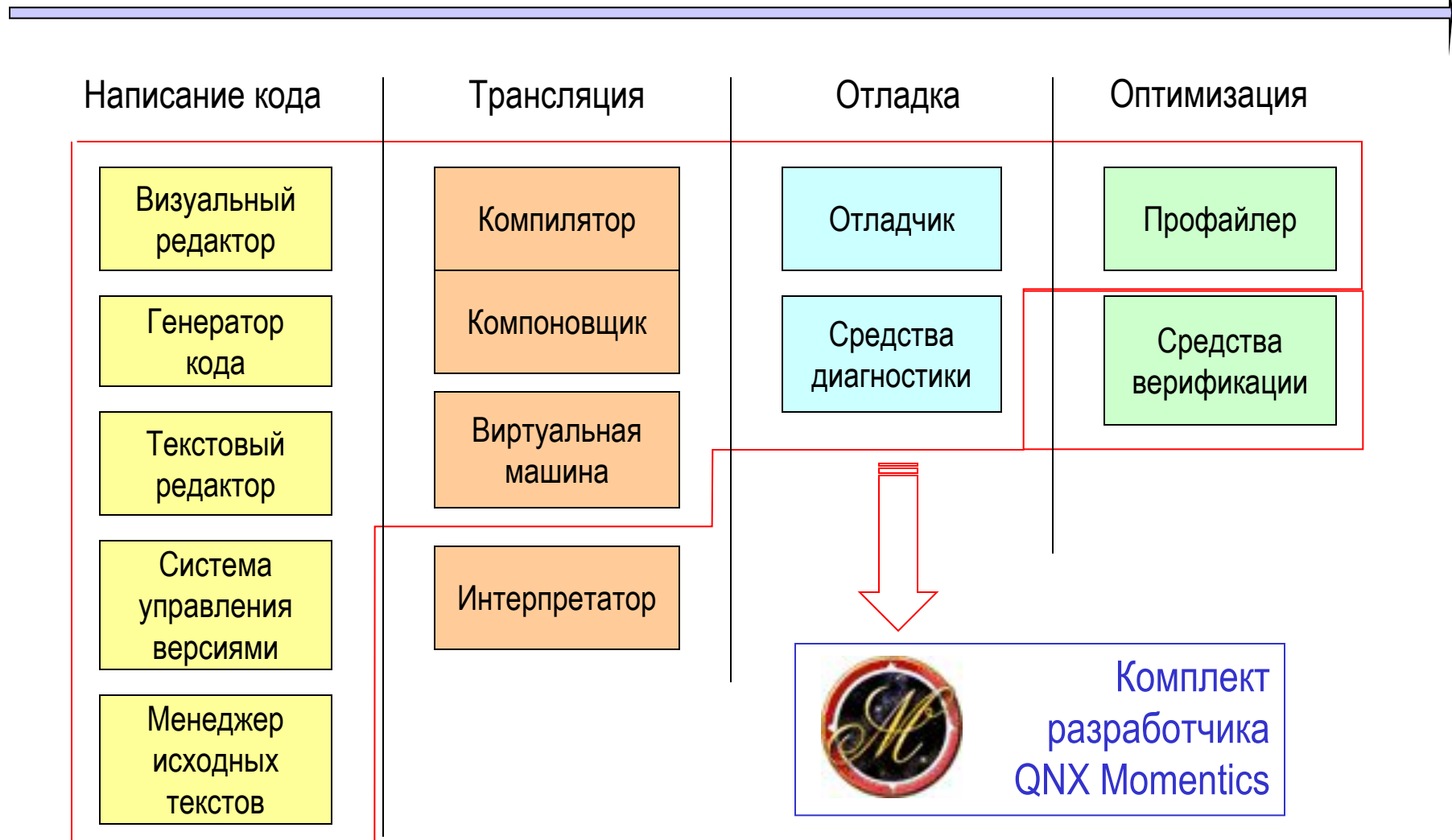
- **Bound Multiprocessing** собирает в себе лучшее из симметричной и ассиметричной моделей
- Поддерживает существующие и оптимизированные для multi-core приложения
- Разработчик имеет полный контроль над приложениями
- **Балансировка нагрузки**
  - Как автоматическая на уровне ОС, так и настраиваемая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





# Комплект разработчика QNX Momentics

## Цикл разработки





## □ 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**

The screenshot shows the QNX IDE interface with a C source file open. The code includes headers, a structure definition, and a function. Annotations highlight key words and syntax. The 'Tasks' window at the bottom shows compiler warnings. The 'Outline' window on the right shows the project structure.

```

#include <stdlib.h>
#include <stdio.h>

typedef struct directory_entry {
    struct directory_entry *next;
    char *name;
} directory_entry_t;

void get_directory_listing(char *name, int len) {
}

int main(int argc, char *argv[]) {
    int i;
    int len;

    printf("Welcome to the QNX Momentics\n");

    for(i = 0; i < argc; i++) {
        printf("Checking directory %s \n", argv[i]);

        len = strlen(argv[i]);
    }

    get_directory_listing(argv[i]);

    return EXIT_SUCCESS;
}
    
```

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

**Список идентификаторов позволяет перейти к любой точке в исходном тексте**

**Щелкните два раза, чтобы построить проект для любой платформы**

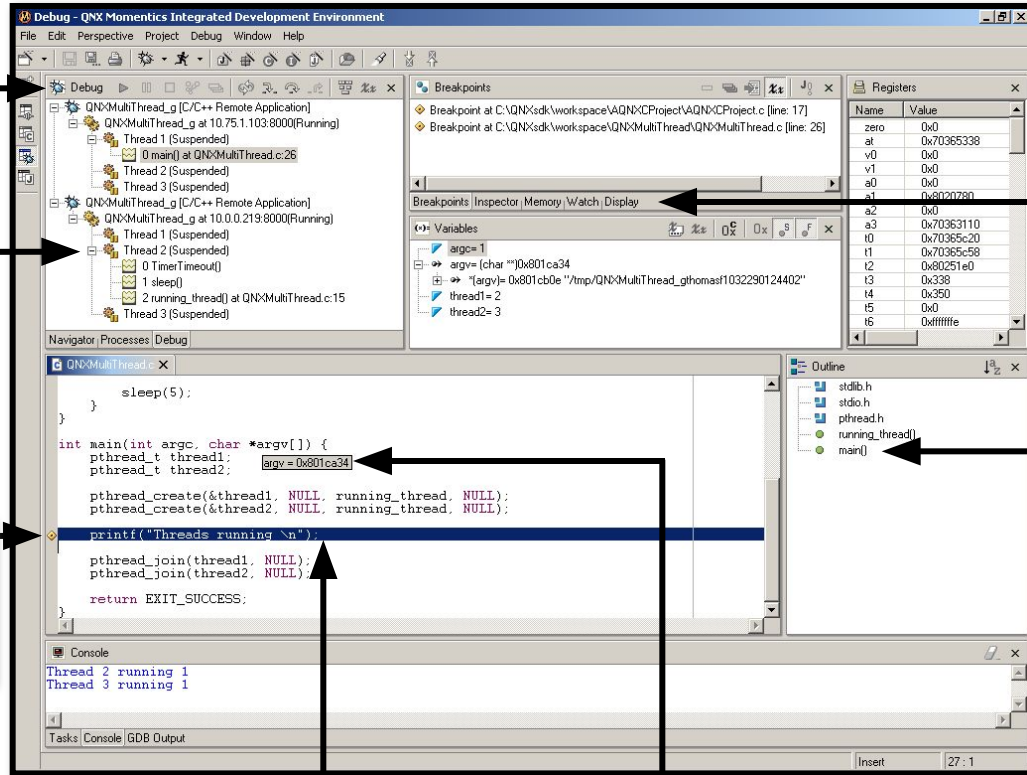
**Наведите указатель мыши на функцию, чтобы посмотреть ее аргументы и нужные заголовки, или на имя переменной, чтобы увидеть ее тип**



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

## Системная информация и использование памяти

The screenshot displays the QNX IDE interface with several key components:

- System Summary:** Shows host information (thomas2), OS version (6.2.1), and system memory (28M/383M).
- Processes (42):** A table listing running processes and servers with columns for Name, Heap, CPU, and Start.
- Memory of devb-fdc:** A memory map showing segments for Program, Stack, Heap, and Library.
- Process devb-fdc:** A detailed view of a specific process, including its arguments, threads, environment, and owner information.
- Thread Information:** A table listing threads for the devb-fdc process, including thread ID, algorithm, stack space, CPU time, and start time.
- System Blocking Graph:** A visual representation of system blocking, showing threads and their states (e.g., servicing request, waiting for request).

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.

The screenshot displays the QNX Profiler interface with several key components:

- Profiler Window:** Shows the project structure and active threads. Thread #1 is highlighted with a red bar, indicating it is the most loaded.
- Sampling Information Table:** A table listing functions and their performance metrics.
 

Function	Total Time (s)	Time since last reset (s)	Call Count	usec/Call	% Time Usage
radix	175.584	175.594	6391	25117.151	
straight	89.925	89.925	6391	12862.967	
ins	11.039	11.039	6991	1579.030	
quick	0.459	0.459	6991	65.656	
merge	13.352	13.352	6991	1909.884	
lookup	0.003	0.003			
shl	0.246	0.246	6991	35.188	
heap	1.920	1.920	6991	274.639	
fill	2.717	2.717	5928	48.580	
rand	0.678	0.678			
rand	0.415	0.415			
- Call Information Window:** Shows call pairs and a call graph. The call graph is a hierarchical tree where 'start' calls 'main', which calls 'bubble', 'shl', and 'ins'. 'bubble' further calls 'radix', 'quick', and 'merge'.
- Code Editor:** Displays C++ code for 'bubble' and 'merge' functions. Colored bars on the left of the code indicate execution time for each line.
 

```

      L3: p=R[0]->L; q=0; Kt=R[j]->K;
          if (Kt<R[p]->K) goto L5;
      L5: q=p; p=R[q]->L; if (p>0) goto L3;
          R[q]->L=j; R[j]->L=p;
      }

      void bubble(RECORD **R, int N) {
          int j, t, BOUND;
          RECORD *Rt = (RECORD *)malloc(sizeof(RECORD) * N);
          BOUND=N;
          B2: t=0; for (j=1; j<=BOUND-1; j+=1) {
              if (t!=0) { BOUND=t; goto B2; }
              free(Rt);
          }

          void merge(RECORD **R, int N) {
              int i, j, d, p, q, r, t;
              RECORD *Rt = (RECORD *)malloc(sizeof(RECORD) * N);
              t=lg(N, HIGH); for (j=1, p=pow(2, t-1); p<=
                  q=pow(2, t-1); r=0; d=p;
                  for (i=0; i<N-d; i+=1) if ((i&p)=r) {
                      if (R[i+1]->K>R[i+d+1]->K) swap(R[i],
                          R[i+d]); if (i&=d) { d=q/2; r=p; goto M3
                          p=p/2; if (p>0) goto M2;
                      }
                  }

          void quick(RECORD **R, int N) {
              int i, j, l, r, K, S;
      
```

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 interface. At the top, the 'Target Navigator' shows a project tree with 'MemoryUsage\_g' selected. The main editor shows the source code for 'MemoryUsage.c', with a buffer overflow in the 'a\_bad\_function' highlighted. Below the code, the 'Memory Events' pane shows a list of events, including a 'Pointer within malloc region, but outside of malloc data bounds' error. To the right, the 'Malloc Information for MemoryUsage\_g' pane provides a summary of heap usage, including a bar chart showing 'used: 1K', 'overhead: 3K', and 'free: 10K'. At the bottom, the 'Allocation Trace' pane shows a table of memory allocation events.

Call	Pointer	Len	Where
malloc	0x0804C218	34	libmalloc.so.2 (DB strdup+0x000000D2)0x8820F826
malloc	0x0804D960	8	libc.so.2 (atexit+0x0000001D)0x8031BA15
malloc	0x0804D998	8	libc.so.2 (atexit+0x0000001D)0x8031BA15
malloc	0x0804D9D0	10	a_bad_function [C:/QNX/sdk/workspace/MemoryUsage/MemoryUsage.c:11]
malloc	0x0804DA08	10	a_bad_function [C:/QNX/sdk/workspace/MemoryUsage/MemoryUsage.c:10]
malloc	0x0804DA40	10	a_bad_function [C:/QNX/sdk/workspace/MemoryUsage/MemoryUsage.c:11]
malloc	0x0804DA78	10	a_bad_function [C:/QNX/sdk/workspace/MemoryUsage/MemoryUsage.c:11]
malloc	0x0804DA78	10	a_bad_function [C:/QNX/sdk/workspace/MemoryUsage/MemoryUsage.c:11]
malloc	0x0804DAB0	10	a_bad_function [C:/QNX/sdk/workspace/MemoryUsage/MemoryUsage.c:11]
malloc	0x0804DAB0	10	a_bad_function [C:/QNX/sdk/workspace/MemoryUsage/MemoryUsage.c:11]

Отслеживание операций распределения памяти

Просмотр объема свободной и используемой 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

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

- Code Coverage Sessions:** A tree view showing coverage for 'server\_demo' (32.56%), 'server.c' (32.56%), 'handle\_read' (0%), and 'main' (75.61%).
- Code Editor:** Displays the source code for 'server.c' with a vertical bar on the right side indicating coverage status for each line (green for covered, red for not covered).
- Debug Console:** Shows the execution of 'server\_demo' with PID 4300816.
- Properties Window:** Displays coverage statistics for the selected file.

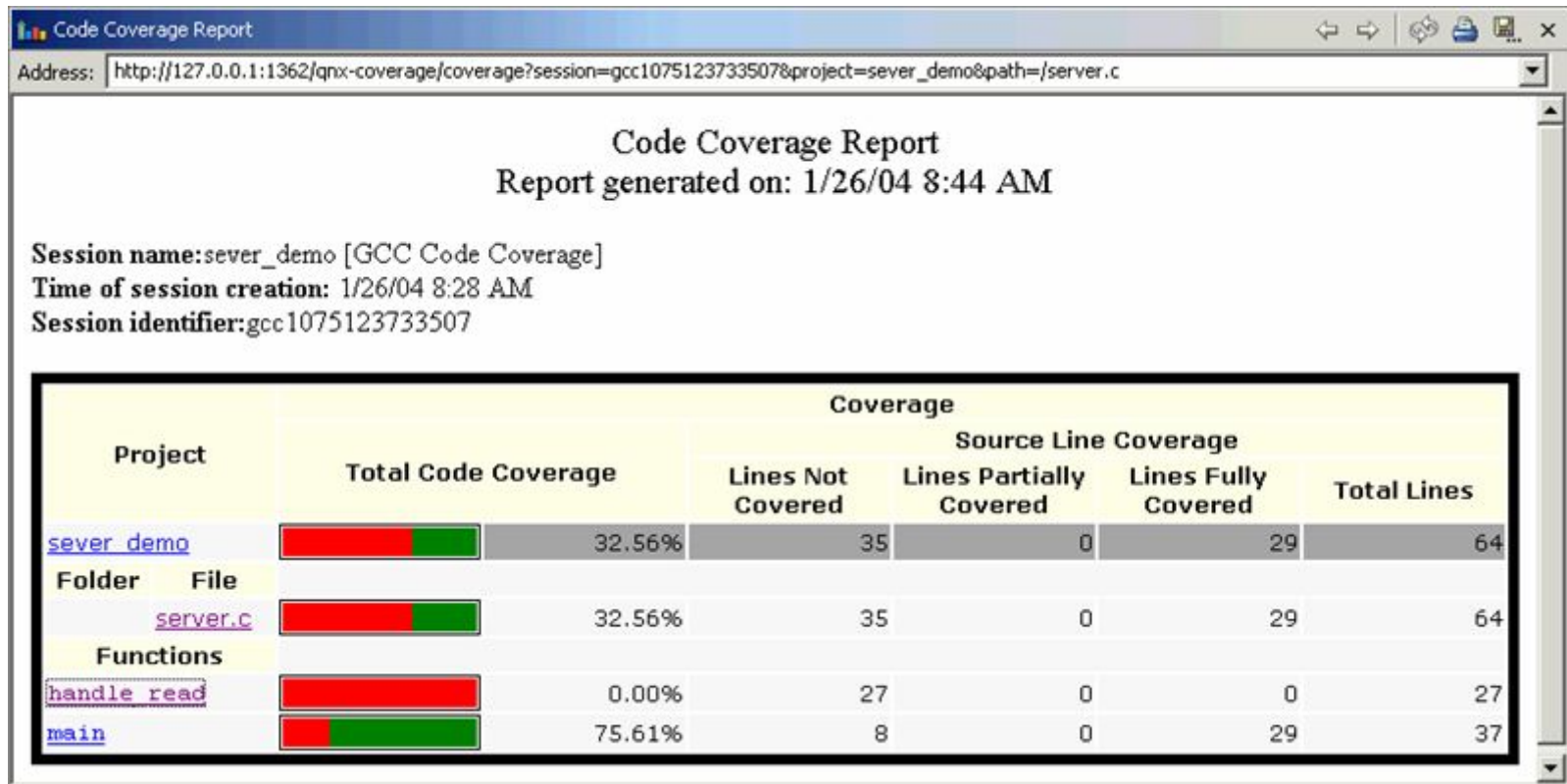
Property	Value
Coverage Info	
Lines Fully Covered	45.31% (29 lines)
Lines Not Covered	54.69% (35 lines)
Lines Partially Covered	0% (0 lines)
Total Coverage	32.56%
Info	
derived	false
editable	true
last modified	5/21/03 2:15 PM
linked	false
location	E:\QNX630\workspace\server_demo\server.c
name	server.c

Отладка:  
просмотр  
запущенных  
процессов

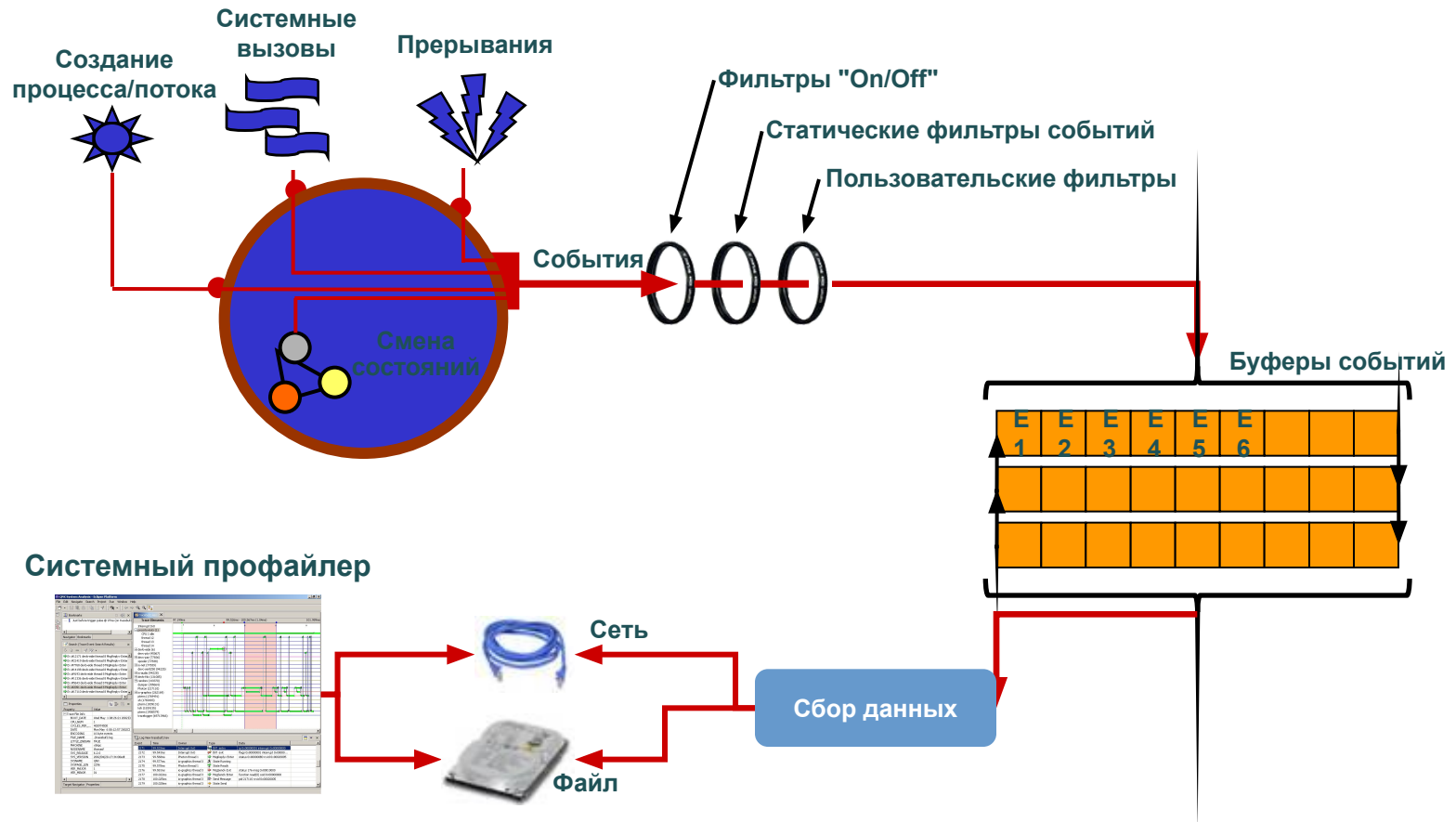
за сессии:  
оценка

## 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





The screenshot displays the QNX System Profiler interface. At the top, a timeline shows the execution of several threads: 'procnto-instr Thread 4', 'Thread 8', 'devc-ply Thread 1', 'tracelogger Thread 1', and 'usertest Thread 1'. A vertical red bar highlights a specific time interval. Below the timeline is the 'Trace Event Log' for 'userevents.kev', listing events with their times, owners, and data. At the bottom, the 'General Statistics' window shows a table of states and their durations.

Event	Time	Owner	Type	Data
17785	1s 65ms 343us	procnto-instr Thr...	_KER_RING0 Exit	ret_val 0x00000000
17786	1s 65ms 347us	procnto-instr Thr...	MsgReply Enter	status 0x00000000 rcvid 0x00000027
17787	1s 65ms 352us	usertest Thread 1	State Ready	pid 917541 tid 1
17788	1s 65ms 353us	procnto-instr Thr...	MsgReply Exit	ret_val 0
17789	1s 65ms 356us	procnto-instr Thr...	MsgReceive Enter	chid 0x00000001 rparts 2080
17790	1s 65ms 358us	procnto-instr Thr...	State Receive	pid 1 tid 8
17791	1s 65ms 359us	tracelogger Thre...	State Running	pid 913444 tid 1

States	Calls	Avg Duration	Max Duration	Min Duration	Total Duration
State Running	3702	71.2us	49ms 956us	3us	2s 635ms 257us
State Ready	2559	150us	34ms 596us	0us	384ms 934us
State Receive	1717	30ms 773us	2s 151ms 130us	6us	50s 930ms 547us
State Reply	1484	5ms 459us	1s 261ms 258us	11us	8s 68ms 755us
State Send	258	28us	256us	20us	7ms 294us
State SigWaitInfo	41	76ms 821us	679ms 614us	827us	2s 361ms 460us
State NanoSleep	11	581ms 879us	1s 19ms 510us	121ms 899us	4s 73ms 154us
State WaitThread	8	6us	8us	4us	52us
State Dead	6	280us	657us	35us	1ms 681us

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

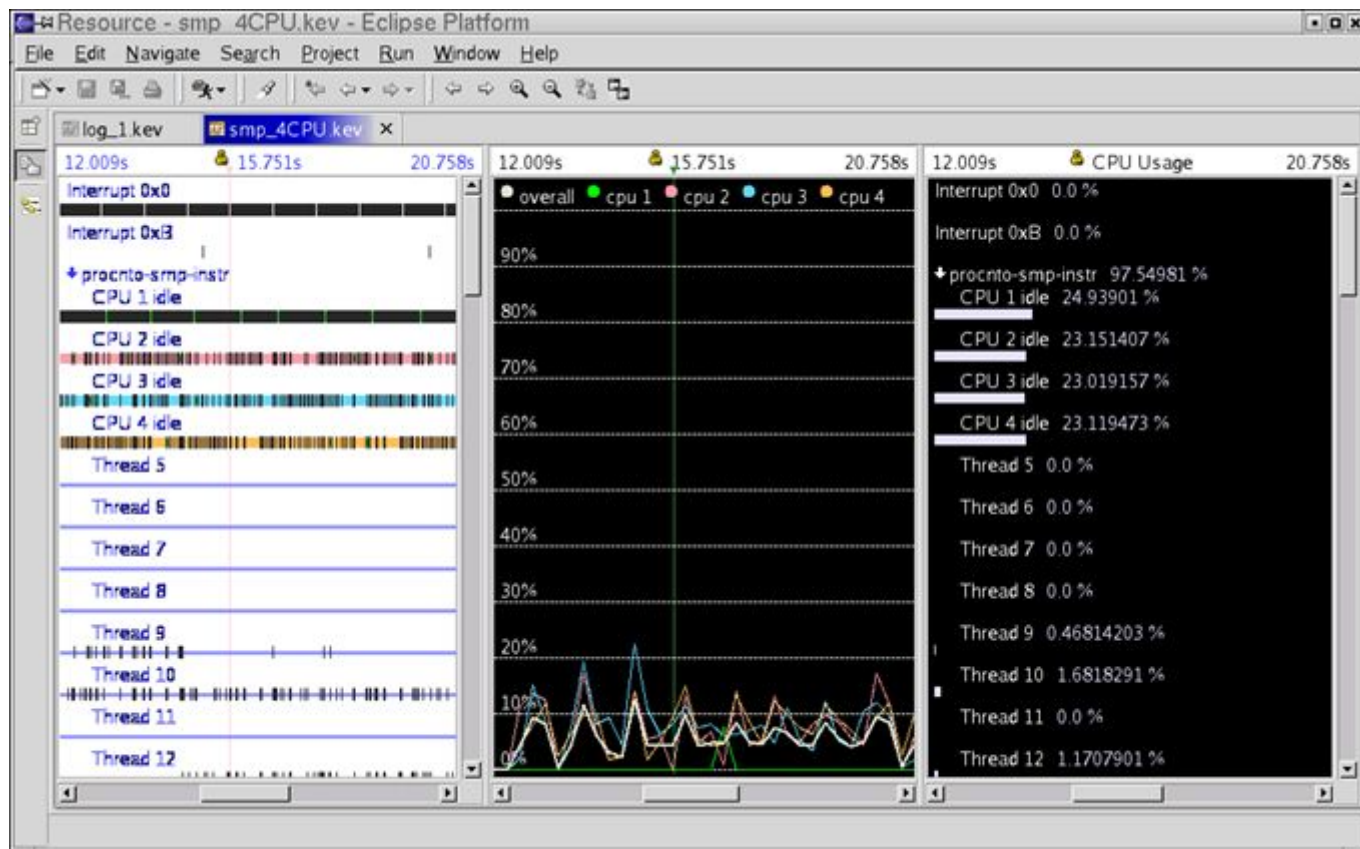
Новое окно статистики

- Табличное представление
- Статистические выборки
- Активность владельцев событий

А также...  
Фильтры пост-обработки переработаны с учетом расширяемости

**% активности  
CPU от общего  
времени**

**Разбиение активности  
CPU по элементам  
трассировки**





# QNX IDE: построитель встраиваемых конфигураций

The screenshot displays the QNX System Builder IDE interface. On the left, the 'Item Browser' shows a hierarchical tree of files and libraries. A yellow callout bubble points to this tree with the text 'Дерево файлов'. The main window shows the details for the 'devc-ser8250' component, including its usage and options. A 'System Optimizer' dialog box is open in the foreground, with a yellow callout bubble pointing to it containing the text '"Мастер" построения'. The dialog has several checkboxes: 'Remove unused files', 'Add missing libraries', and 'Apply diet(s) system wide'. At the bottom, there is a 'Serial Terminal' window showing 'Connected to COM1' and a baud rate of '9600,n,8,1'.

Дерево файлов

"Мастер" построения

System Optimizer  
Please select the optimization options:

- Remove unused files
- Add missing libraries
- Apply diet(s) system wide

< Back Next > Finish Cancel

Property	Value
Code Segment	Use In Place
Data Segment	Copy
Enabled	Yes
File Name	devc-ser8250
File Permissions	777
Group ID	0
Location On Target	/proc/boot
Raw Data	No

Serial Terminal  
Device: COM1 Settings... Send BREAK Clear Terminal Send File...  
Connected to COM1 9600,n,8,1

The screenshot displays a QNX desktop environment with the following elements:

- Terminal Window:** Shows the command `echo "В pterm можно писать по-русски"` and its output, demonstrating Cyrillic text rendering.
- File Editor (Untitled1 [modified]):** Contains a C program:
 

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    printf("Привет Мир!!!!\n");
    return EXIT_SUCCESS;
}
```
- Application Charset Dialog:** Shows settings for "Application Charset" (Cyrillic (cp866)) and "Font Charset" (Cyrillic (KOI8-R)).
- System Menu:** Lists various applications and utilities, including "Localization".
- Taskbar:** Shows the taskbar with icons for "Launch", "tty0: sh", "xterm", "Mope(!) анал...", "ptermcs util", and "Untitled1 [m...".

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

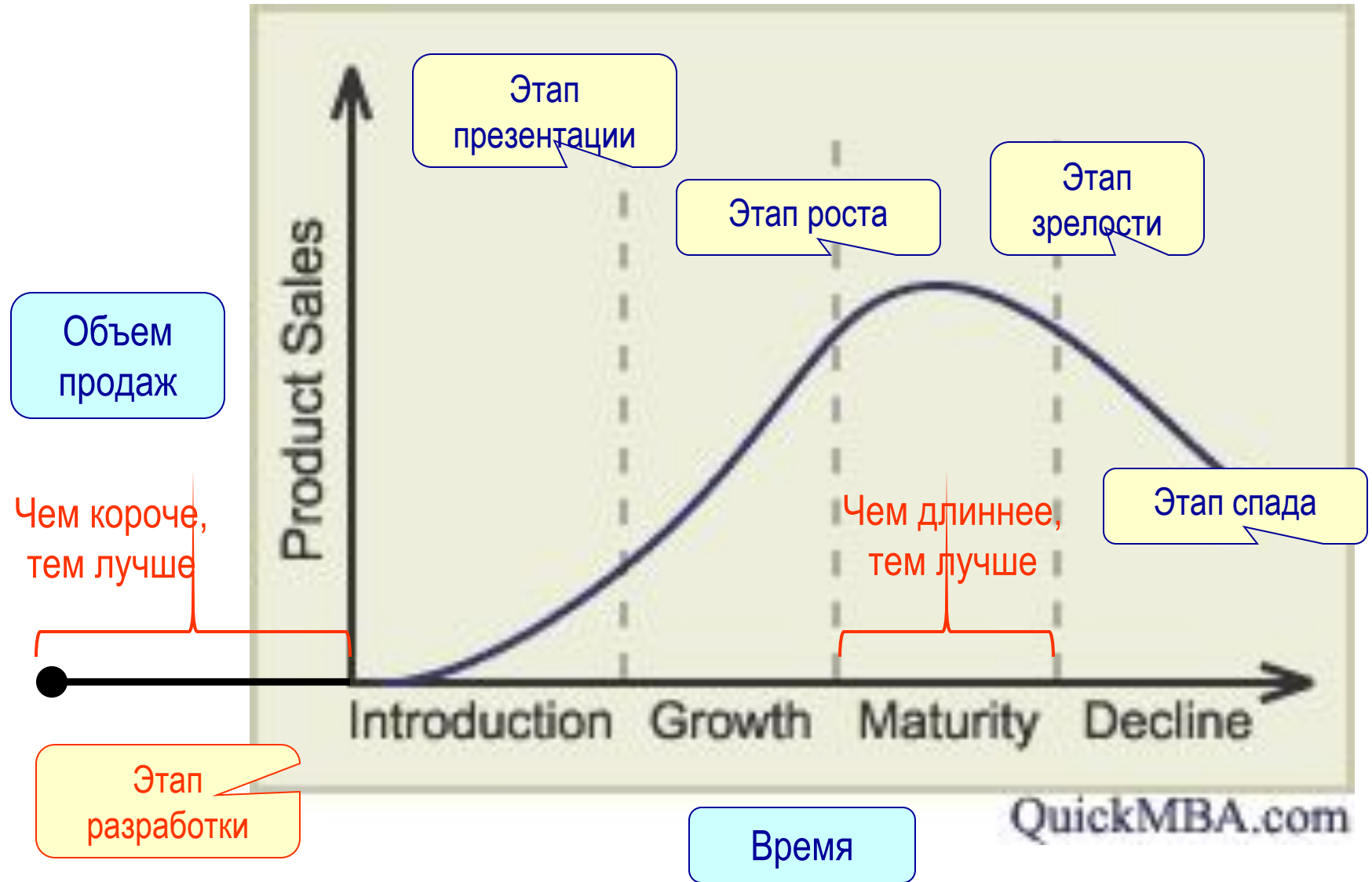
# QNX помогает экономить

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

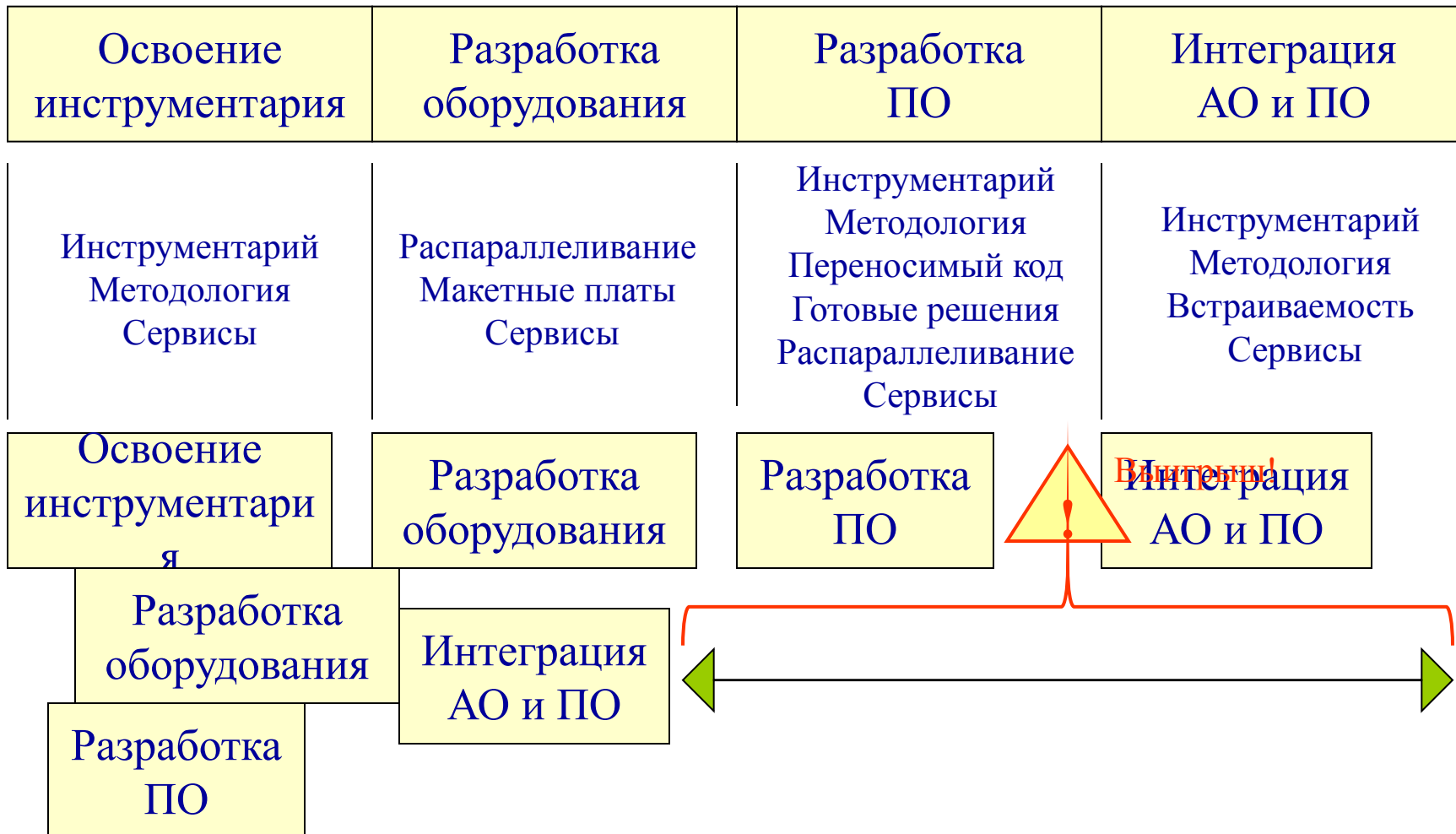
TTM = Time-To-Market,  
"время выхода на рынок"

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

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



Время





# QNX как средство сокращения ТТМ

<b>Эффективный инструментарий</b>	<ul style="list-style-type: none"><li>▪ Интегрированный комплект QNX Momentics</li><li>▪ Инструментарий "третьих" фирм</li><li>▪ Стартовые комплекты</li></ul>
<b>Прозрачная методология разработки</b>	<ul style="list-style-type: none"><li>▪ Изящная архитектура</li><li>▪ Модульная организация</li><li>▪ Встроенная распределенная сеть</li></ul>
<b>Простота адаптации к оборудованию</b>	<ul style="list-style-type: none"><li>▪ Открытая унифицированная модель драйвера</li><li>▪ DDK с примерами</li><li>▪ Доступность исходных текстов</li></ul>
<b>Возможность переноса кода из других проектов, в т.ч. из других ОС</b>	<ul style="list-style-type: none"><li>▪ Полная поддержка POSIX API</li><li>▪ Модель "клиент/сервер" для сервисов ОС</li></ul>
<b>Доступность готовых решений</b>	<ul style="list-style-type: none"><li>▪ Партнерская сеть QNX Partner Network</li></ul>
<b>Простота встраивания</b>	<ul style="list-style-type: none"><li>▪ Компактность</li><li>▪ Модульность</li></ul>
<b>Профессиональные сервисы</b>	<ul style="list-style-type: none"><li>▪ SWD Software Ltd. – поддержка, консалтинг, сертифицированное обучение, заказные разработки</li></ul>



## 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

<b>Дешевая аппаратура</b>	<ul style="list-style-type: none"><li>▪ неприхотливость к ресурсам</li></ul>
<b>Стоимость программных компонентов</b>	<ul style="list-style-type: none"><li>▪ Модульное лицензирование</li></ul>
<b>Надежность</b>	<ul style="list-style-type: none"><li>▪ Надежная архитектура на основе микроядра</li><li>▪ Все системные модули выполняются вне пределов ядра в защищенных адресных пространствах</li><li>▪ Использование аппаратного диспетчера памяти</li></ul>
<b>Живучесть</b>	<ul style="list-style-type: none"><li>▪ Поддержка автоматического самовосстановления на уровне отдельных компонентов</li><li>▪ Поддержка режима высокой готовности (коэффициент готовности 99.999% и выше)</li></ul>
<b>Дешевизна в обслуживании, диагностике и модернизации</b>	<ul style="list-style-type: none"><li>▪ Возможность обновления и перезапуска любого программного модуля "на лету" без перезагрузки ОС</li><li>▪ Поддержка удаленного обновления с использованием защищенных сетевых протоколов</li><li>▪ Масштабируемость и расширяемость</li><li>▪ Полностью русифицирована</li><li>▪ Поддержка удаленного интерфейса пользователя</li></ul>



# Небольшое резюме

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

**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](mailto:qnx@swd.ru)