

Технологии **QNX**® для создания
высоконадежных систем автоматизации



Сергей Зыль, СВД ВС
**Состав и функциональные возможности
QNX SDP, QNX Neutrino и КПДА.10964-01**





QNX SDP состоит из двух частей:

- ❑ QNX Momentics - инструментальный комплект
- ❑ QNX Neutrino – компоненты среды исполнения

Для российских предприятий, имеющих соответствующие лицензии, доступна также

- ❑ ЗОСРВ «Нейтрино» – компоненты среды исполнения



QNX Neutrino 6.5.0 поддерживает следующие аппаратные платформы:

- ARM-LE
- ARMv7-LE
- MIPS-BE
- MIPS-LE
- PPC-BE
- PPC-LE
- SH4-LE
- X86-LE





QNX Neutrino 6.5.0 соответствует стандартам:

- ✓ IEEE POSIX (PSE52)
- ✓ ISO/IEC 15408 (EAL 4+)
- ✓ IEC 61508 (SIL 3)
- ✓ OpenGL® ES
- ✓ IEC 62304
- ✓ ISO 26262





QNX Neutrino и IEC 61508

SIL (МЭК 61508)	Коэффициент снижения риска	Суммарное время эксплуатации (ч)
4	10 000 – 100 000	3×10^9
3	1 000 – 10 000	3×10^8
2	100 – 1 000	3×10^7
1	10 - 100	3×10^6

← **34246,58 лет**

Данные приведены для уровня доверительной вероятности 95%

- QNX Neutrino: статистика «Proven-in-Use» по использованию для 6.3.0 и выше (все SP) (GE Energy – для x86 и PowerPC)



QNX Neutrino 6.5.0 соответствует стандартам:

- ✓ IEEE POSIX (PSE52)
- ✓ ISO/IEC 15408 (EAL 4+)
- ✓ IEC 61508 (SIL 3)
- ✓ OpenGL® ES
- ✓ IEC 62304
- ✓ ISO 26262

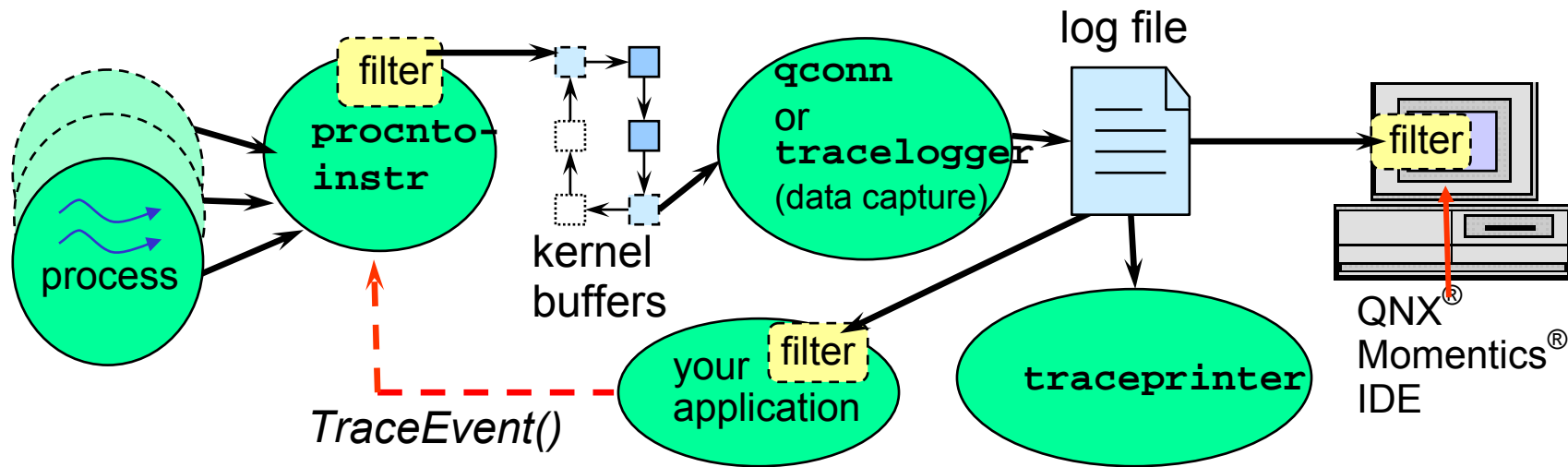




- Поддержка многоядерных процессоров
- Инструментированное ядро**
- Адаптивное квотирование
- Быстрая активация устройств (IDA)
- Файловые системы
- Сетевые возможности
- Графические интерфейсы
- Обеспечение самовосстановления



Системное профилирование



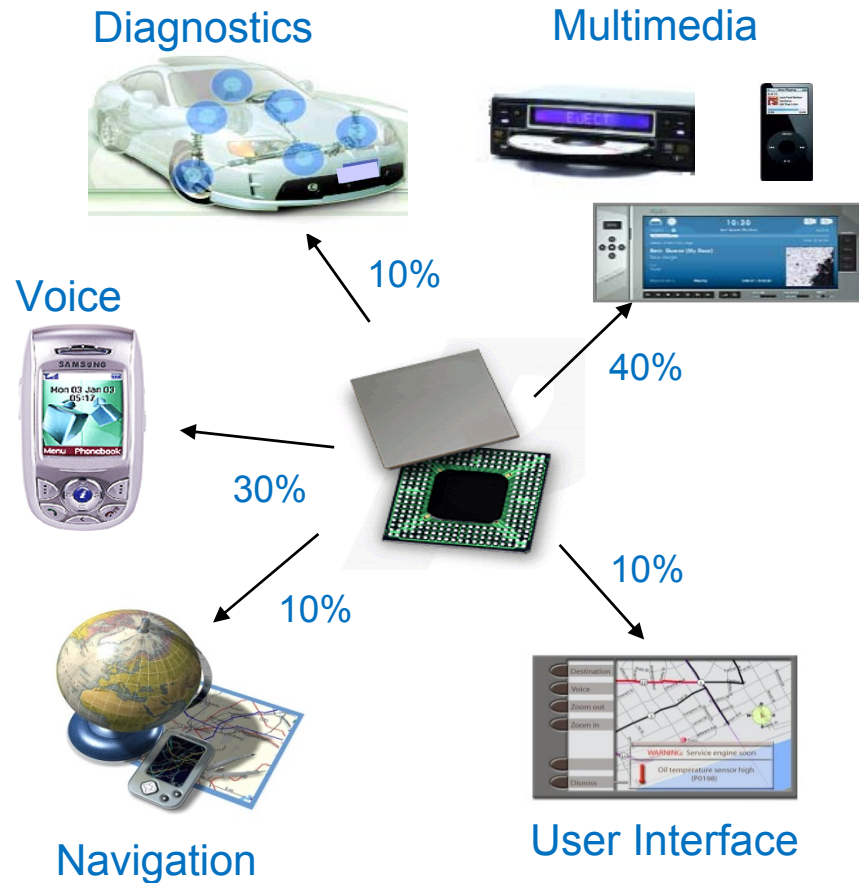
- ❑ Instrumented Kernel Technology



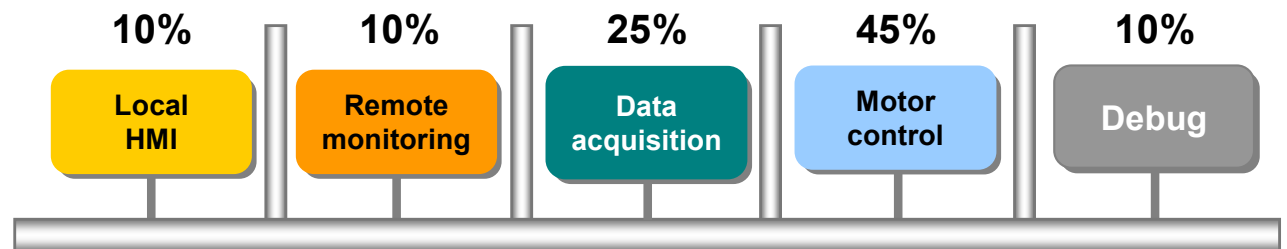
- Поддержка многоядерных процессоров
- Инструментированное ядро
- Адаптивное квотирование**
- Быстрая активация устройств (IDA)
- Файловые системы
- Сетевые возможности
- Графические интерфейсы
- Обеспечение самовосстановления



Адаптивное квотирование ЦПУ



□ Adaptive Partitioning Technology

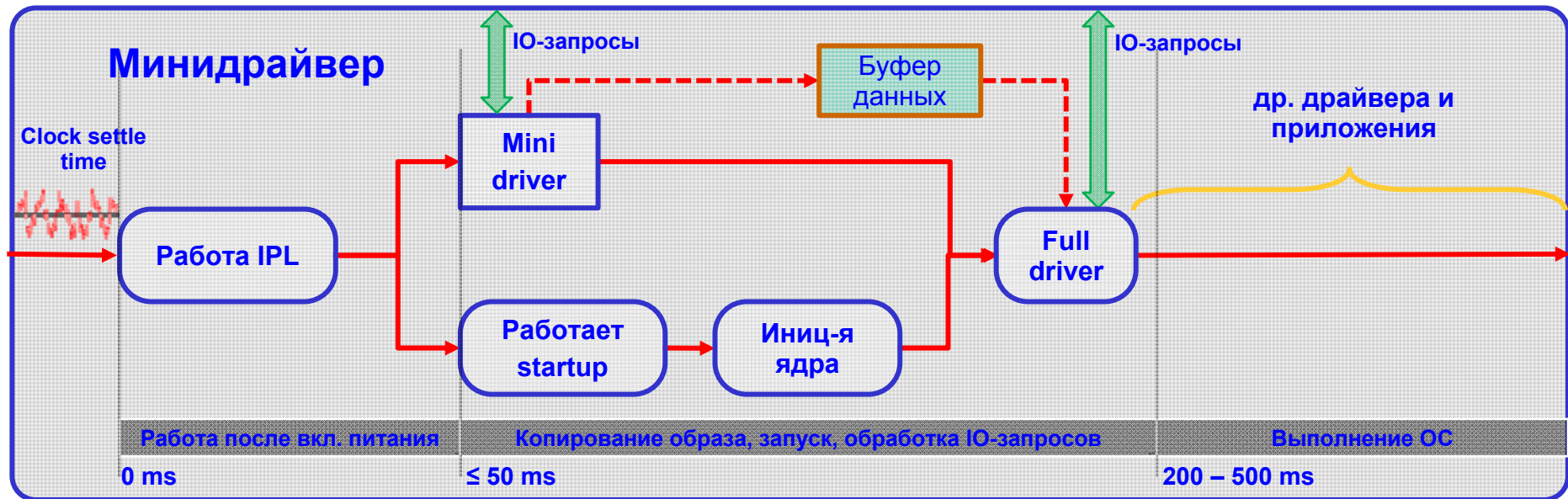




- Поддержка многоядерных процессоров
- Инструментированное ядро
- Адаптивное квотирование
- Быстрая активация устройств (IDA)**
- Файловые системы
- Сетевые возможности
- Графические интерфейсы
- Обеспечение самовосстановления



Быстрая активация устройств



- ❑ Instant Device Activation Technology



- Поддержка многоядерных процессоров
- Инструментированное ядро
- Адаптивное квотирование
- Быстрая активация устройств (IDA)
- Файловые системы**
- Сетевые возможности
- Графические интерфейсы
- Обеспечение самовосстановления



ОЗУ-резидентные:

Image (procnto)
/dev/shmem/ (procnto)
RAM (devb-ram или io-blk.so)

На накопителях:

QNX4, Power-Safe (QNX6)
FAT, NTFS
Ext2, HFS, HFS+
UDF, CD-ROM, CD-RW, DVD

На Flash (NAND, NOR)

FFS3, ETFS

Сетевые

NFS, CIFS (SMB)

SQL

QDB

Виртуальные

Inflator

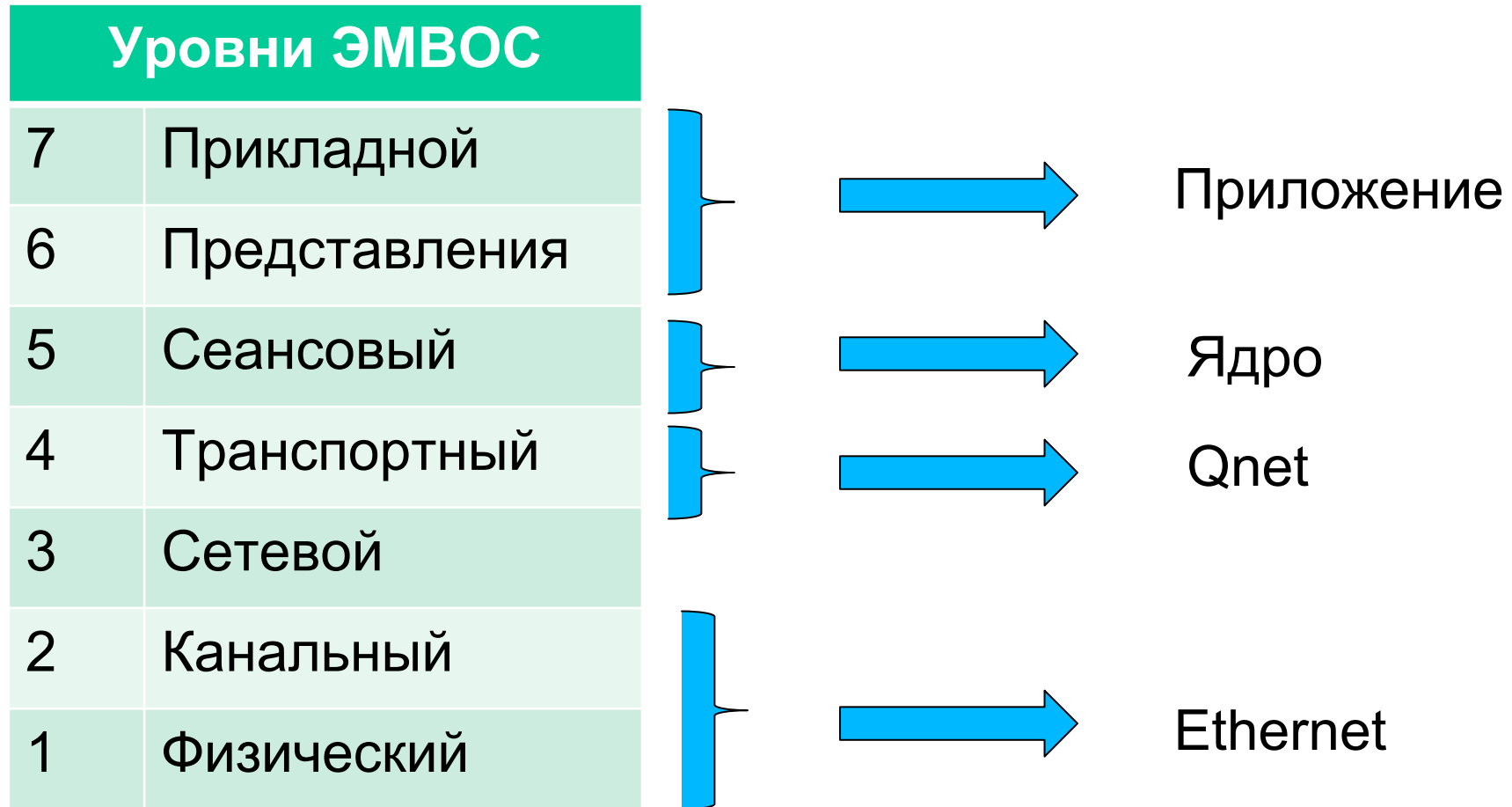


- Поддержка многоядерных процессоров
- Инструментированное ядро
- Адаптивное квотирование
- Быстрая активация устройств (IDA)
- Файловые системы
- Сетевые возможности**
- Графические интерфейсы
- Обеспечение самовосстановления



Прозрачно распределенная сеть Qnet

Транспортный уровень протокола Qnet (4 уровень модели OSI) располагается ниже логического уровня микроядра QNX!!!





- Поддержка многоядерных процессоров
- Инструментированное ядро
- Адаптивное квотирование
- Быстрая активация устройств (IDA)
- Файловые системы
- Сетевые возможности
- Графические интерфейсы (отдельная презентация)
- Обеспечение самовосстановления

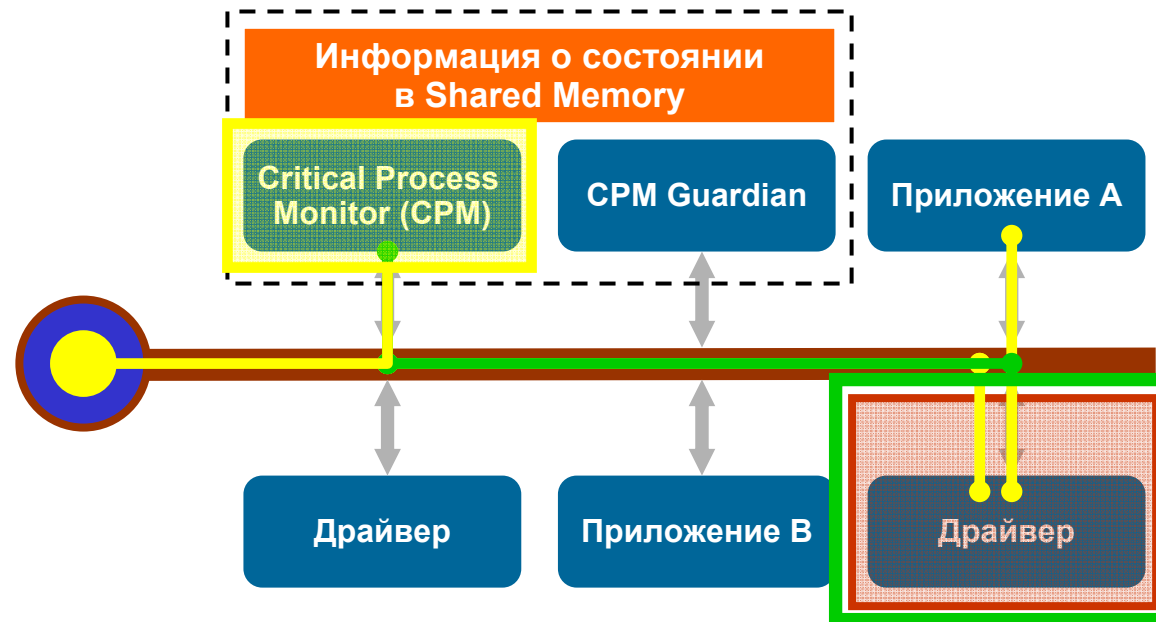


- Поддержка многоядерных процессоров
- Инструментированное ядро
- Адаптивное квотирование
- Быстрая активация устройств (IDA)
- Файловые системы
- Сетевые возможности
- Графические интерфейсы
- Обеспечение самовосстановления**



Монитор ключевых процессов

1. Сбой драйвера (некорректный доступ к памяти)
2. Ядро извещает CPM о сбое процесса



3. Сохраняется диагностическая информация (core-файл и kev-файл)
4. Драйвер завершается возвращает системе ресурсы; канал IPC разрушен
5. CPM перезапускает драйвер
6. Клиентская библиотека CPM восстанавливает каналы IPC
7. Драйвер восстанавливает сервис по состоянию на последней точке синхронизации из CPM



QNX SDP состоит из двух частей:

- ❑ **QNX Momentics - инструментальный комплект**
- ❑ QNX Neutrino – компоненты среды исполнения

Для российских предприятий, имеющих соответствующие лицензии, доступна также

- ❑ ЗОСРВ «Нейтрино» – компоненты среды исполнения



Хост-платформы QNX Momentics:

MS Windows



Linux



QNX Neutrino



Все – x86



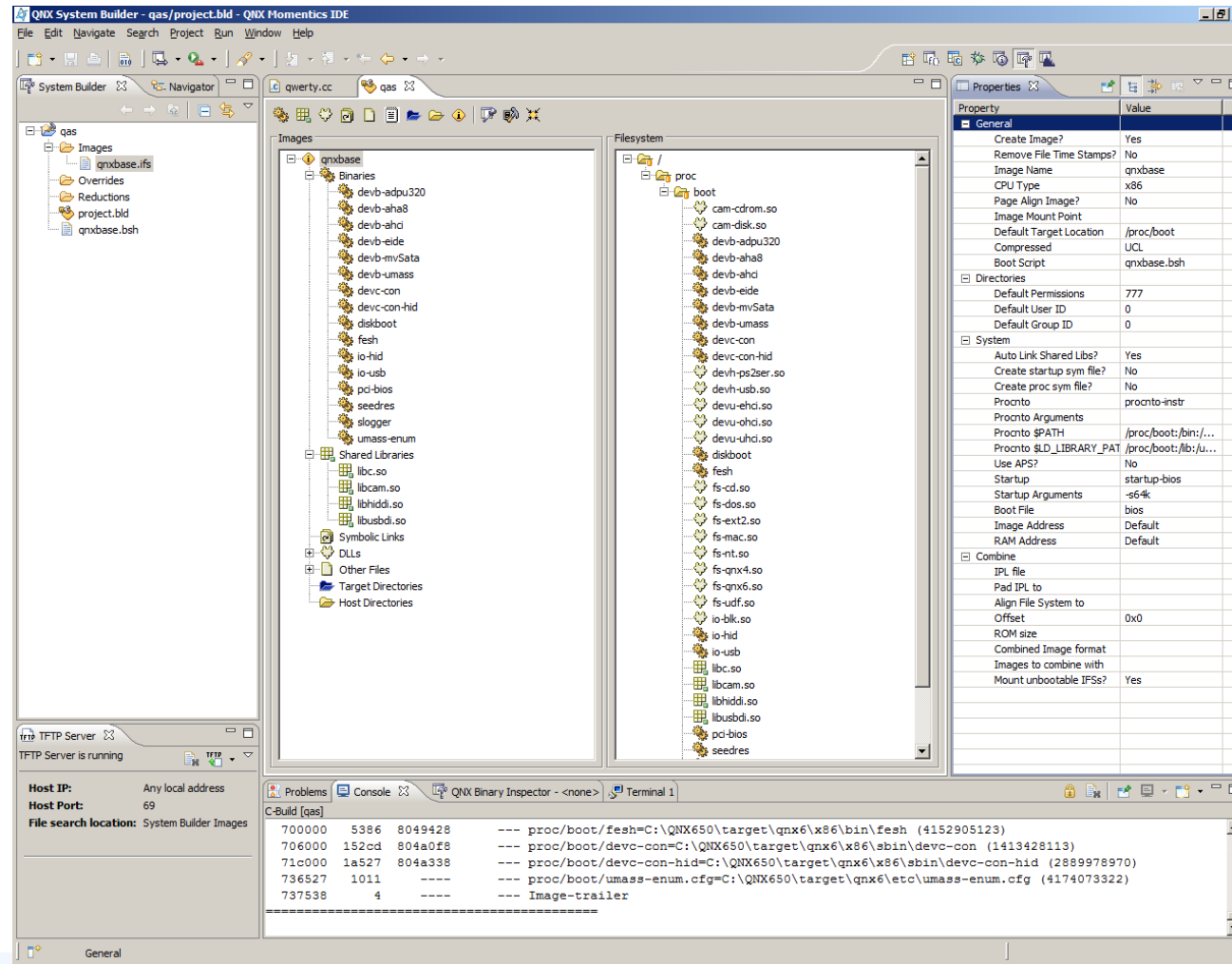


Функционально QNX Momentics состоит из следующих частей:

- ❑ Инструменты для формирования из «кирпичиков» адаптированных под конкретное назначение целевых систем QNX Neutrino и их анализа - для каждой из 3-х поддерживаемых хост-платформ
- ❑ Инструменты для разработки, отладки и тестирования собственных компонентов



Построение целевых систем (QNX System Builder)





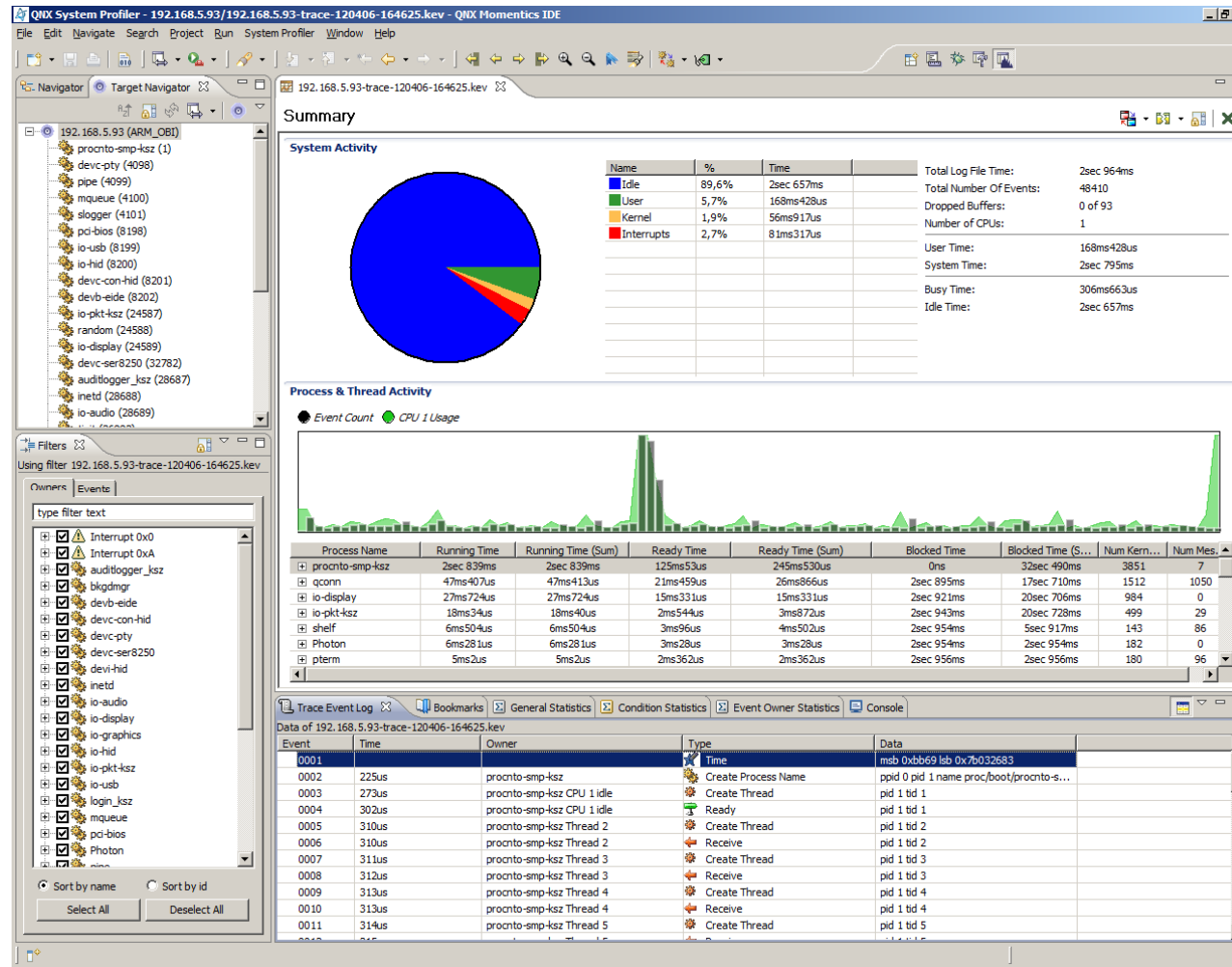
Анализ целевых систем (QNX System Information)

The screenshot displays the QNX System Information tool interface. On the left, a 'Target File System Navigator' shows a directory tree for the path '192.168.5.93:/etc'. The right pane shows 'Process Information' for the target IP '192.168.5.93', last updated on 06/16/2012. A legend indicates thread states: green for 'Servicing request', blue for 'Waiting for request', purple for 'Waiting for reply', and light blue for 'Waiting for service'. A diagram shows 'tinit (36882) Thread 1' and 'qconn (1433631) Thread 1' connected to 'procnto-smp-ksz (1) Channel 1', which in turn connects to a vertical stack of 16 'procnto-smp-ksz (1) Thread' boxes (Thread 2 through Thread 16). A table below the diagram lists the objects and their owners.

Object	Object Owners	State	Blocked Threads
procnto-smp-ksz...	procnto-smp-ksz (1) Thread 9	Running	tinit (36882) Thread 1
	procnto-smp-ksz (1) Thread 2	Receive	qconn (1433631) Thread 1
	procnto-smp-ksz (1) Thread 3	Receive	
	procnto-smp-ksz (1) Thread 4	Receive	
	procnto-smp-ksz (1) Thread 5	Receive	
	procnto-smp-ksz (1) Thread 7	Receive	
	procnto-smp-ksz (1) Thread 11	Receive	
	procnto-smp-ksz (1) Thread 13	Receive	
	procnto-smp-ksz (1) Thread 14	Receive	
	procnto-smp-ksz (1) Thread 15	Receive	
	procnto-smp-ksz (1) Thread 16	Receive	
devr-ntv (4098)	devr-ntv (4098) Thread 1	Receive	sh (2011166) Thread 1



Анализ целевых систем (QNX System Profiling)





Функционально QNX Momentics состоит из следующих частей:

- ❑ Инструменты для формирования и анализа целевых систем QNX Neutrino
- ❑ Инструменты для разработки, отладки и тестирования собственных компонентов-«кирпичиков» для работы на поддерживаемой аппаратуре – на каждой из 3-х хост-платформ



Анализ целевых систем (QNX Memory Analysis)

The screenshot displays the QNX Memory Analysis tool interface. The main window shows the source code for 'qwerty.cc' with the following content:

```
#include <cstdlib>
#include <iostream>

int main(int argc, char *argv[]) {
    std::cout << "Welcome to the QNX Momentics IDE" << std::endl;
    return EXIT_SUCCESS;
}
```

The Memory Events table at the bottom right shows 237 events:

Kind	Requested	Actual	Pointer	TID	Location
malloc (_dl_alloc)	312	312	0x8074048	1	
malloc (_dl_alloc)	16	20	0x805c170	1	
malloc (_dl_alloc)	16	20	0x805c1b0	1	
malloc (_dl_alloc)	312	312	0x807bec0	1	
malloc (_dl_alloc)	16	20	0x805c1f0	1	
malloc (_dl_alloc)	16	20	0x805c230	1	
malloc (_dl_alloc)	312	312	0x807bd58	1	
malloc (_dl_alloc)	16	20	0x805c270	1	
malloc (_dl_alloc)	16	20	0x805c2b0	1	
malloc (_dl_alloc)	312	312	0x807bbf0	1	
malloc (_dl_alloc)	16	20	0x805c2f0	1	
malloc (_dl_alloc)	16	20	0x805c330	1	
malloc	8	20	0x805c370	1	<no source code>
malloc	8	20	0x805c3b0	1	<no source code>
malloc	8	20	0x805c3f0	1	<no source code>
malloc	512	512	0x807b9c0	1	<no source code>
malloc	8	20	0x805c430	1	<no source code>
malloc	8	20	0x805c470	1	<no source code>
malloc	8	20	0x805c4b0	1	<no source code>
malloc	8	20	0x805c4f0	1	<no source code>
new	4	20	0x805c530	1	<no source code>

The console window at the bottom left shows 17 memory leak events:

Severity	Description
LEAK	memory leak of size 4
LEAK	memory leak of size 4
LEAK	memory leak of size 4
LEAK	memory leak of size 4
LEAK	memory leak of size 4
LEAK	memory leak of size 4
LEAK	memory leak of size 4
LEAK	memory leak of size 8
LEAK	memory leak of size 8
LEAK	memory leak of size 8
LEAK	memory leak of size 8
LEAK	memory leak of size 8
LEAK	memory leak of size 8
LEAK	memory leak of size 8
LEAK	memory leak of size 20
LEAK	memory leak of size 20
LEAK	memory leak of size 20



Профилирование приложения (QNX Application Profiler)

The screenshot displays the QNX Application Profiler interface. The main window is titled "QNX Application Profiler - qwerty/qwerty.cc - QNX Momentics IDE". The interface is divided into several panes:

- Profiler Sessions:** Shows a tree view with "qwerty #34 [09.04.12 16:33]" expanded to show "qwerty" and "unknown".
- Execution Time:** Displays a "Threads Tree" table with columns: Name, Deep Time, Shallow Time, Count, Location, and Binary. The table lists various threads and their execution times, with "main (self)" highlighted.
- Console:** Shows the output of the application: "Welcome to the QNX Momentics IDE".
- Source Code:** Displays the source code for "qwerty.cc", showing the main function and standard library includes.

Name	Deep Time	Shallow Time	Count	Location	Binary
[100.0%] Thread 1	7,539 ms	7,539 ms	1		
[00,16%] __do_global_ctors_aux	11,834 us		1		qwerty
[00,16%] __do_global_ctors_aux	11,834 us		1		qwerty
[00,14%] __do_global_ctors_a	10,569 us		1		qwerty
global constructors keyed t	10,569 us	10,569 us	1		qwerty
global constructors keyed to m	1,265 us	00,02%	1,265 us		qwerty
[99,84%] _start	7,527 ms		1		qwerty
[99,84%] _start->main	7,527 ms		1		qwerty
[06,78%] _start->main	0,511 ms		1		qwerty
main (self)	28,202 us	28,202 us	1		qwerty
[06,41%] main->std::basic_ostream<char, std::basic_ostream<char, char_traits<char>>><>	0,483 ms		1		qwerty
[01,20%] main->std::cout	90,350 us		2		qwerty
[00,36%] main->std::cout	27,054 us		10		qwerty
[00,23%] main->std::cout	17,438 us		7		qwerty
[00,13%] n	10,030 us		8		qwerty
[00,07%] n	4,901 us		2		qwerty
[<0.01%] std::cout	0,473 us		1		qwerty
[00,11%] std::basic_ostream<char, std::basic_ostream<char, char_traits<char>>><>	8,332 us		1		qwerty
[00,08%] std::basic_ostream<char, std::basic_ostream<char, char_traits<char>>><>	5,968 us		4		qwerty
[00,02%] s	1,869 us		4		qwerty
[00,64%] std::basic_ostream<char, std::basic_ostream<char, char_traits<char>>><>	48,012 us		1		qwerty
[00,59%] std::basic_ostream<char, std::basic_ostream<char, char_traits<char>>><>	44,774 us		3		qwerty
[00,01%] s	1,036 us		2		qwerty
[04,76%] std::basic_ostream<char, std::basic_ostream<char, char_traits<char>>><>	0,358 ms		1		qwerty
[04,09%] std::basic_ostream<char, std::basic_ostream<char, char_traits<char>>><>	0,309 ms		4		qwerty
[<0.01%] std::cout	0,469 us		1		qwerty
main (self)	4,114 ms	50,5%	4,114 ms		qwerty
[38,49%] main->std::basic_ostream<char, std::basic_ostream<char, char_traits<char>>><>	2,910 ms		1		qwerty
[16,54%] main->std::basic_ostream<char, std::basic_ostream<char, char_traits<char>>><>	1,247 ms		10		qwerty
[02,06%] main->std::basic_ostream<char, std::basic_ostream<char, char_traits<char>>><>	0,156 ms		2		qwerty
[00,51%] main->std::cout	38,490 us		4		qwerty
std::basic_ostream<char, std::basic_ostream<char, char_traits<char>>><>	1,091 ms	4,47%	1,091 ms		qwerty
std::basic_ostream<char, std::basic_ostream<char, char_traits<char>>><>	1,641 ms	3,76%	1,641 ms		qwerty
[00,19%] std::basic_ostream<char, std::basic_ostream<char, char_traits<char>>><>	14,165 us		1		qwerty
[00,16%] std::basic_ostream<char, std::basic_ostream<char, char_traits<char>>><>	11,793 us		4		qwerty
[00,03%] std::basic_ostream<char, std::basic_ostream<char, char_traits<char>>><>	1,903 us		4		qwerty
std::basic_ostream<char, std::basic_ostream<char, char_traits<char>>><>	9,890 us	00,13%	9,890 us		qwerty
std::basic_ostream<char, std::basic_ostream<char, char_traits<char>>><>	2,372 us	00,03%	2,372 us		qwerty

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

int main(int argc, char *argv[]) {
    std::cout << "Welcome to the QNX Momentics IDE" << std::endl;
    return EXIT_SUCCESS;
}
```



Тестирование приложения (QNX Code Coverage)

The screenshot displays the QNX Code Coverage IDE interface. The main window shows the source code for `qwerty.cc`. The left sidebar contains a tree view of the project structure and code coverage statistics for various files and functions. The bottom right pane displays a detailed code coverage report.

Code Coverage Report
Report generated on: 16.04.12 14:21

Session name: [qwerty \[GCC Code Coverage\]](#)
Time of session creation: 16.04.12 14:18
Session identifier: com.qnx.tools.ide.coverage.core.gcc.34.gcc1334571511241

Project/Folder/File/Function	Total Code Coverage	Source Line Coverage			Total Lines
		Lines Not Covered	Lines Partially Covered	Lines Fully Covered	
C:\ide-4.7-workspace\qwerty	100.00%	0	0	4	4



QNX SDP состоит из двух частей:

- ❑ QNX Momentics - инструментальный комплект
- ❑ QNX Neutrino – компоненты среды исполнения

Для российских предприятий, имеющих соответствующие лицензии, доступна также

- ❑ **ЗОСРВ «Нейтрино» – компоненты среды исполнения**



Изделие КПДА.10964-01 основано на:

QNX Neutrino RTOS версии 6.5.0

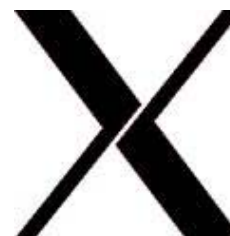
КСЗ «Нейтрино»

Qt

Firefox

GTK+

XPhoton



▪ ARM-LE, MIPS-LE/BE, PowerPC-BE, x86

▪ Документация (переведено > 2 тыс. стр.)



QNX SDP версии 6.5.0

- ❑ Компоненты QNX Neutrino для всех поддерживаемых целевых аппаратных платформ – ARM, MIPS, SH4, PPC, x86
- ❑ Инструменты для формирования целевых систем QNX Neutrino и их анализа для хост-платформ Windows, Linux, QNX
- ❑ Инструменты для разработки, отладки и тестирования собственных компонентов для всех хост-платформ



Спасибо за внимание!

Сергей Зыль

ООО «СВД Встраиваемые Системы»

www.kpda.ru

forum.kpda.ru