science + computing

A Bull Group Company

Android NDK
API + ABI

Dr. Olaf Flebbe
science + computing ag
IT-Dienstleistungen und Software für anspruchsvolle Rechnernetze
Tübingen | München | Berlin | Düsseldorf

Über mich

science + computing
A Bull Group Company

- Atari/Minix (Floatingpoint Implementierung)
- Linux (1994 ?) libm.so.4
- Psion5 (EPOC, ARM) perl, python
- ...
- 1997 science+computing …
- …

Langfristiges Ziel der Arbeiten

science + computing
A Bull Group Company

- Verteilte Simulation bei FlightGear (www.flightgear.org)
  - Visuals
  - Controls
  - Multiplayer
  - AI
- Standards für verteilte Simulationen:
  - HLA (High Level Architecture)
  - ISO Standards z.B. IEEE 1516 Evolved
- Anbindung von Visuals auf mobilen Gerät
- Anbindung von Sensorik mobiler Geräte

OpenRTI

science + computing
A Bull Group Company

- Autor: Mathias Fröhlich
  - http://gitorious.org/**openrti**
- Implementierung (WIP) von diversen HLA Standards RTI-NG, IEEE 1516, IEEE 1516e
- C++ Implementierung: Multi-threaded, RTTI
- IEEE 1516e Standard verlangt:
  - Wide Strings std::wstring
  - STL Support (std::vector, std::set)
  - Runtime shared libs mit runtime dependency resolution
    - Eigene „Zeitdefinition“ möglich
  - Exceptions
  - D.h. nicht embedded freundlich!

Zielplattform

science + computing
A Bull Group Company

- Android, weil „Linux“ ... Später mehr dazu
- Weil Linuxtag ☺
- Problem: API von Android ist Java/Dalvik
- Strategie 1:
  - Rad neu erfinden
    - Netzwerkprotokoll noch einmal implementieren
    - Aber: Protokoll ist nicht standardisiert, nur die API
- Strategie 2:
  - Code portieren
  - Investitionen sichern

Android NDK

science + computing
A Bull Group Company

- Erzeugt „Native Code“
- C/C++ Umgebung für Android Targets
  - ARM
  - Intel(seit NDK-7)
  - MIPS (seit NDK-8)
- Das Android NDK Enthält einen gcc als fertigen Crosscompiler von Linux/Intel32, Windows32 und MacOSX auf diese Plattformen, inkl. Utilities, Header und Libraries.

Standardkonformität ?

science + computing
A Bull Group Company

- ANSI C, POSIX nahezu vollständig
  - Fehlt: Wide Char API, Multibyte Support „Unicode“
- Kein ANSI C++ !
  - libstdc++ Standard Bibliothek für minimales C++ Subset im ROM.
  - Kein RTTI:
    - Kein dynamic_cast<Type>()
  - Keine STL
    - D.h. Kein std::vector<float>
  - Fehlt
    - Wide Strings std::wstring bsp = L“Breit“;
- Siehe docs/CPLUSPLUS.html

Die ABI‘s

science + computing
A Bull Group Company

- Application binary Interface: Der Kontrakt welcher Code vom Prozessor ausgeführt wird, wie Argumente an Funktionen übergeben werden, wie Aufruf des Betriebssystems ...
  - Beispiel Linux: i586, amd64
- Android Java/Dalvik Code ist portabler Bytecode und kann deswegen auf jeder Android Plattform ausgeführt werden.

  Hier gibt es im Moment nur eine ABI. D.h. Code läuft im Prinzip auf jeder Android Architektur.

Android ABIs

science + computing
A Bull Group Company

- Im Gegensatz zur Java/Dalvik Entwicklung muss man native Code für jede Zielplattform compilieren.
- Intel (32Bit)
- MIPS
- ARM
  - Armeabi
  - Armv7 ABI

ARM ABIs

science + computing
A Bull Group Company

- Sehr verwirrend bezeichnet:
- ARM-v7 Befehlssatz (Ab Cortex A8)
  - Enthält Hardwarebeschleunigte Vector Floating Point Operationen VFLP
  - „Neue“ Befehle (NEON)
- ARM (v4) z.B: Für ARM7 und ARM9 Prozessoren
- Android 4.0 setzt einen ARM-v7 Befehlssatz/Prozessor ab Cortex A8 voraus!
- ARM kennt zwei Betriebsmodi:
- RISC (32Bit) und „thumb“ (16Bit)
- Die Thumb Befehle sind kompakt (16Bit), dafür aber etwas langsamer: Standard bei Android.
- Genauer im NDK docs/CPU-ARCH-ABIS.html

Android APIs und ihre NDK Unterstützung

science + computing
A Bull Group Company

- Die Verschiedenen Release unterstützen unterschiedliche Funktionalitäten (Auszug)
- Android 2.2 (Froyo) Android-8
  - Libjnigraphics (JNI Graphic)
- Android 2.3 (Gingerbread) Android 9
  - Embedded OpenGL, libandroid (Durchgriff auf Android)
  - Einzelne pthread_* ein paar fake wide api befehle
- Android 4.0 (Ice Cream Sandwich) Android-14
  - libOpenMAX, OpenSLES (Sound)
  - ARMEABI-v7

Probleme

science + computing
A Bull Group Company

- clock_nanosleep Kernel Syscall Implementierung fehlerhaft (Falsche Anzahl Parameter)
- Der runtime Linker /system/bin/linker ist sehr einfach gestrickt und kann keine zirkulären Referenzen
- Multibyte/Unicode Unterstützung nicht existent/fehlerhaft. (wchar_t == char)
- Bionic libc ist sehr einfach, wohl eher eine Minimalimplementierung, um die Dalvik Engine zu unterstützen. (Wie bei Symbian...)

Header sind anders aufgeteilt

C++

science + computing
A Bull Group Company

- Das Android NDK unterstützt mehrere Laufzeitumgebungen
- Siehe docs/CPLUSPLUS-SUPPORT.html

	C++ Exceptions	C++ RTTI	Standard Library
system	no	no	no
gabi++	no	yes	no
stlport	no	yes	yes
gnustl	yes	yes	yes

- Gnustl entspricht der libstdc++ von Desktop Linux Systemen.

C++

science + computing
A Bull Group Company

- „System“ libstdc++.so ist minimal Laufzeitbibliothek (new, delete, iostream). Diese ist im „ROM“
- Jede Applikation kann und muss eine erweiterte Standard C++ Umgebung mitbringen.

CrystaX

science + computing
A Bull Group Company

- Projekt von **Dmitry Moskalchuk:**
- Verbesserung der Android „gnustl“ Bibliothek:
- Wide Char (unicode) support
- Gcc-4.6.3 toolchain (Besseres Exception Handling)
- ...
- http://www.crystax.net
- libcrystax_shared.so für Fixes an der libc.so im ROM, kann Applikationslokal sein
- Ein Problem mit select() hat Dmitry in Windeseile gelöst.
- Unterstützt Dmitry!

Emulator

science + computing
A Bull Group Company

- android avd
- Emulator-arm –data-partition-size 111 um
- Emulator hat ausgehendes Netz (NAT)
- Eingehende Ports mit
- adb forward tcp:port1 tcp:ports2
- Unterschied zu einem Gerät: adb shell liefert root shell

Android Filesystem

science + computing
A Bull Group Company

- /system (ro), nicht schreibbar
- /data Applicationen
- /data/data: Apps
- /data/local <- Hier kann man entwickeln
- /data/tmp
- /sdcard (noexec mount): Hier können keine Executables liegen.

Nutzen des NDK

science + computing
A Bull Group Company

- Erstellen eine Standalone Toolchain
  - `build/tools/make-standalone-toolchain.sh`
  - Dann mit autoconf etc weitermachen
- Skript buggy (zumindest in NDK v7), sodass C++ Support fehlt.
- Verwendung der Build Skripte von google
  - ndk-build
  - Man schaut sich die Flags/Bibliotheken von ndk-build V=1 ab
  - z.B.: für cmake toolchain file

Externe Tools

science + computing
A Bull Group Company

- Strace (Download)
- Valgrind (Android/ARM Support upstream)

Remote Debugging

science + computing
A Bull Group Company

- Device:
  - Gdbserver aus dem NDK kopieren!
  - gdbserver localhost:PORT cmd args

- HOST:
- Gdbserver aus dem NDK kopieren!
- TCP Forward in den Emulator
  - adb forward tcp:port tcp:port
- Und .gdbinit

Remote Debugging

science + computing
A Bull Group Company

- .gdbinit
- target remote :PORT
- set solib-search-path /.../android-ndk-r7-crystax-5.beta2/platforms/android-9/arch-arm/usr/lib/:/.../android-ndk-r7-crystax-5.beta2/sources/cxx-stl/gnu-libstdc++/libs/armeabi/4.6.3:/.../install-android/lib
- break main
- set history save on

DEMO

science + computing
A Bull Group Company

science + computing

A Bull Group Company

Vielen Dank für Ihre Aufmerksamkeit.

Folien auf www.oflebbe.de

Vortrag von: Dr. Olaf Flebbe
science + computing ag
www.science-computing.de

Telefon: 07071 9457-0

E-Mail: of@oflebbe.de, o.flebbe@science-computing.de