
MAMBA
iNdT

Scenes

- Mamona for Developers
- Platform Developers

Mamona for Developers

Why have we started Mamona?

- Maemo is open, but its development process...
- Maemo provides the platform for application development. What about to provide the development for the platform itself?
- Sometimes a fully-emulated ARM environment would be better
- It isn't easy to update core components in Maemo SDK, like toolchain

Mamona as a SDK

- Mamona - pure ARM chroot
- Mamona-ARM as buildroot of sbox2
- Mamona-ARM as buildroot sbox2 encapsulated in Mamona-i386
- Mamona-ARM chroot environment with noemu packages
- Full Emulated System

Mamona - pure ARM chroot

The trivial case, whose motivation is to make the development as simple as possible.

Advantages:

- easy to upgrade or downgrade a package
- The SDK behaves like the real operating system
- More reliable - everything is being emulated
- Easier - developers don't need to worry about different targets
- The toolchain is easily managed using apt-get

Disadvantages:

- The Kernel isn't emulated
- Emulated compilation is too slow

Mamona-ARM as buildroot of sbox2

- Idea to speed up compilation time
- Use cross compilation and cpu transparency provided by Scratchbox 2
- This approach had never been tried due to Scratchbox 2 host dependency

Mamona with sbox2 encapsulated in Mamona-i386

Why have we given up?

- It is very difficult to maintain this system
- Sometimes it is confusing
- It isn't a good idea to use the sb2 command as root user or change the owner of the buildroot tree
- When something doesn't occur as expected it is very hard and painful to discover where the error comes from

Mamona-ARM chroot environment with noemu packages

Ideas to avoid the slow compilation disadvantage:

- Mamona-i386
- i386 binaries running inside ARM chroot avoiding emulation
  - We call it: noemu packages
  - A noemu package is an ARM package with i386 binaries
  - They must be maintained in a separeted repository
  - It is easy to setup the apt repository
  - The developer knows which noemu packages are installed
  - It is simple, reliable, scalable, upgradable and fast

Full Emulated System

Qemu provides two ways of emulation:

- User Mode - applications are emulated individually
- System Mode - the whole system, including kernel and peripherals are emulated

User Mode disadvantage:

- The development that requires some access to hardware resources like bluetooth, wireless and others remains painful, since the kernel isn't being emulated on it

Full Emulated - Currently there is support for:

- integrator boards
- versatile boards
- PXA boards (Intel XScale)

Full Emulation of Nokia Internet Tablets

- N770: We are implementing the OMAP H3 board (OMAP 1710 that is similar to N770)
- N800: Impossible at this moment due to:
  - ARM restrictions
  - OMAP 2420 documentation

After implemented:

- Although this is the best solution, the issue about slow compilation returns
- Mamona noemu packages cannot be used in this type of emulation
- We can minimize the compilation time using distcc or icecc
- The idea isn't to have only full emulated system, but both. So the developer can choose between faster compilation or full emulation

Platform Developers

OpenEmbedded

- What is it?
- Why use it?
- How does it work?

OpenEmbedded - What is it?

- Is it a SDK?
- It isn't a Distribution
- It is a Build System + Build Recipes (packages)

OpenEmbedded - Why use it?

- Hackers like to code and don't want to spend their time packing
- Automate the build avoiding duplicated work
- OE base has a lot of packages - more than 5000
- Get indirect contribution
- Contribute is better than divide
- Avoid the creation of YA-build system

OpenEmbedded - How does it work?

- OE uses the Bitbake task executor in combination with the OpenEmbedded metadata to build packages, and it's using monotone as its version control system
- OE uses compilation and configuration caching at most levels to increase the developer's productivity
- It is possible to generate images to different types of embedded devices

Bitbake

For BitBake there is nothing else but variables and tasks organized as:

- build/conf/local.conf - The file where Bitbake gets it first instructions
- distro conf - The distribution policy. It can overlap some variables previous set in local.conf
- machine conf - Main arguments to cross compilation
- .bb files - packages specification. Variables and some specific functions
- .bbclass files - Generic functions to run for each package. For instance, package_deb.bbclass that has functions to generate the .deb packages
- .inc files - Secondary files just to organize the other ones

OE build/conf/local.conf

- DISTRO - to select a distribution policy.
- MACHINE - to specify a machine to build for.
- BBFILES - to specify which .bb files to consider for your build.
- TMPDIR - where BitBake should create its temporary files.

OE distro conf

The most important variables for a distro conf file:

- DISTRO_NAME
- DISTRO_VERSION
- INHERIT
- MACHINE_KERNEL_VERSION
- PREFERRED_PROVIDERS
- PREFERRED_VERSION(s),
  e.g.:PREFERRED_VERSION_glibc-initial ?= '2.5'

OE machine conf

The most important variables for a distro conf file:

- TARGET_CC_ARCH = '-march=armv5te -mtune=arm926ej-s'
- PACKAGE_ARCH = 'armv5te'
- TARGET_FPU = 'hard'
- EXTRA_IMAGECMD_jffs2_nokia800 = '–pad –little-endian –eraseblock=0x20000 -n'
- IMAGE_FSTYPES = 'jffs2'
- SERIAL_CONSOLE = '115200 ttyS0'
- EXTRA_IMAGEDEPENDS += '0xffff-native'

OE .bb files

This type of file must contain:

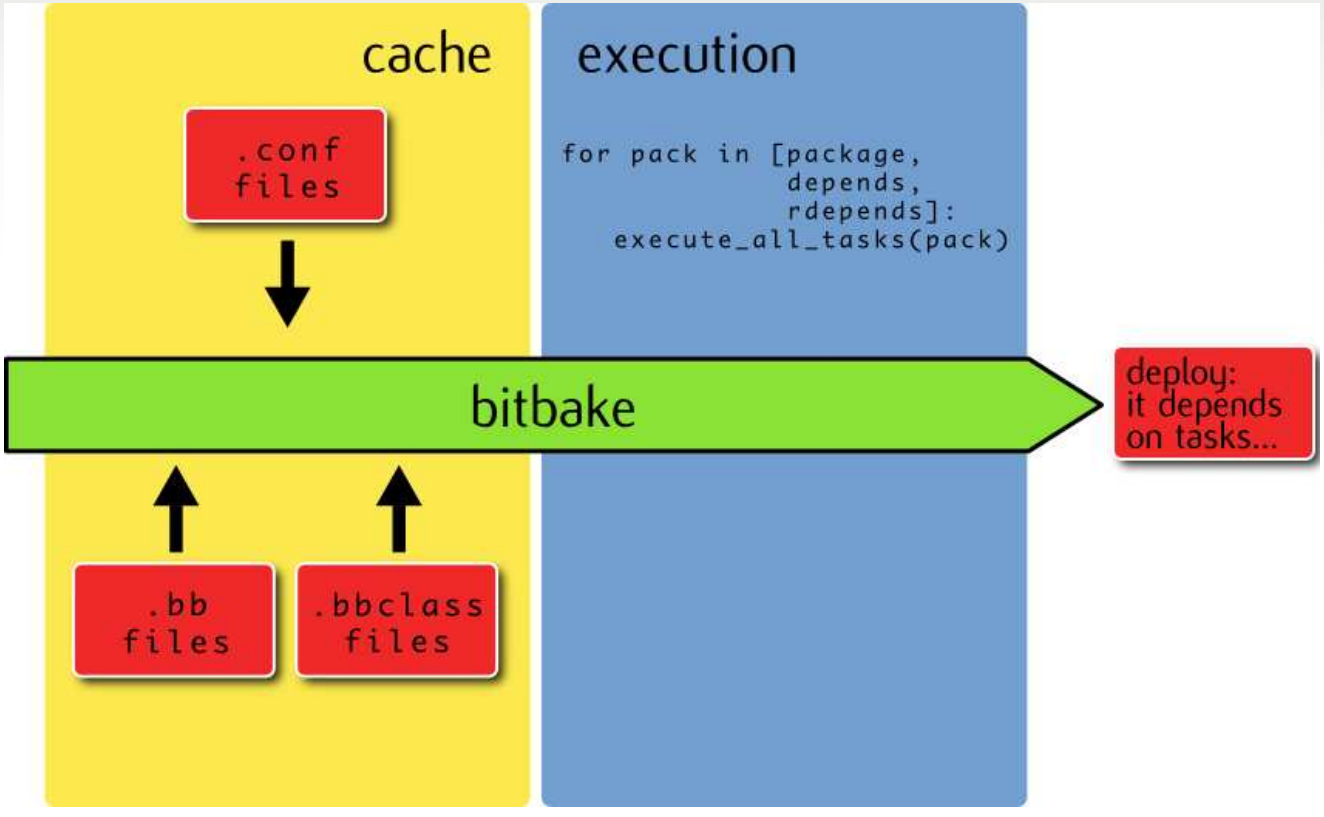
- SRC_URI - source package location
- DEPENDS - list of depends
- RDEPENDS - list of runtime depends
- functions to replace and modify the build defined on a bbclass - autotools or distutils (python)

OE .bbclass files

This file contains functions in shell script or python that can do everything you want. All that you need is to code the function and choose where it fits, e.g

- python do_create_dsc () {...}
- addtask create_dsc before do_configure after do_patch

OE Overview


cache
execution
.conf
files
for pack in [package,
depends,
rdepends]:
execute_all_tasks(pack)
bitbake
deploy:
it depends
on tasks...
.bb
files
.bbclass
files

Plans for Mamona

- The Distro running on internet tablets
  - A free IPR distro
  - Free contributions without legal hassle
- The SDK for the Mamona distro
- The SDK for the Maemo
- System Emulation
- RoadMap

The Distro running on internet tablets

Mamona is already running on N800 with:

- Enlightenment
- Wi-fi - WEP
- Closed programns on initfs - We don't distribute it

What doesn't work yet? Why?

- Wi-fi with WPA - Closed sources
- Sound - Closed sources related to DSP

What do we wish?

- Open all programns of initfs

The SDK for the Mamona distro

- Mamona-ARM chroot environment - 0.1
- Noemu repository - 0.1
- Full Emulation - ?

The SDK for the Maemo

- Mamona doesn't have compatibility with Maemo yet
- We were thinking ahead, and hoping that when we have our first stable release, Maemo would be catching up on us in terms of package and toolchain versions
- By taking a look at Chinook Beta, we can notice that Maemo is almost there
- Maemo and Mamona will have very similar packages at some point in a not so distant future.

System Emulation

- We are implementing the OMAP H3 board
- After this first implementation it will be easy to extend it and implement for other devices based on OMAP

RoadMap - 2007

- Mamona 0.1 - November, 15
  - Mamona distro for N800
    - Enlightenment
    - wi-fi
    - bluetooth
    - python2.5
    - sound*
  - Mamona SDK for Mamona distro
  - OpenEmbedded infrastructure for Mamona

RoadMap - 2008

- Mamona 0.2
  - February, 15
  - Mamona SDK for Maemo (Chinook)
  - OpenEmbedded infrastructure for Maemo (Chinook)
- Mamona 0.3
  - May, 15
  - Mamona distro for next Internet Tablet
  - Hildon and more applications for Mamona distro
- Mamona 0.4
  - August, 15
  - Mamona SDK for Maemo (Diablo)
  - OpenEmbedded infrastructure for Maemo (Diablo)

How can Mamona contribute to Nokia/Maemo?

- Creating a build and integration system allowing even better releases
- Improve the Maemo flexibility
- Independence of Scratchbox and others cross compilation tools

How can Nokia/Maemo contribute to Mamona?

- Releasing proprietary components as open source
- Sharing technical documentation (specs, datasheets, references, ...)
- Following our work

Demonstration - Let's see it working!

Why are you showing this? Show it working...

Discussion

Let's talk...

- Thanks
  - Nokia
  - INdT
- Links
  - http://mamona.garage.maemo.org
  - http://www.openembedded.org/