Embedded Alley

Solutions for Intelligent Devices

OpenEmbedded for Product Development

Matthew Locke

Embedded Linux Conference

April 2008

Agenda

- Background
- Introduction to OE
- Example product description
- OE setup
- Create configurations
- Define tasks
- Root file system
- Application development
- Deployment images
- Summary

Background / History

- Started in the OpenZarus project to be able to easily build applications for the Zaurus PDAs
- Build system was redesigned and rewritten to be more generic breaking out the metadata and build tool into two separate projects
- The build tool, bitbake, is based on concepts in Gentoo/ portage
- Adopted by many open source projects that provide distributions for handhelds.org, Linksys routers, motorola phones, mythTV hardware and many more
- Latest project to use OpenEmbedded is OpenMoko a complete and open mobile phone software stack.

Introduction to OpenEmbedded (OE)

- A **self contained** cross build system for embedded devices
- Collection of recipes (metadata) that describe how to build:
  - Thousands of packages including bootloaders, libraries, and applications
  - For ~60 target machines including the a780, N770 and x86
  - Over 40 package/machine configurations (distributions)
- Does not include source code. Fetches source using instructions in metadata.
- Take any number of components, build, create images- Components can be any source type, SVN, tarball
- Output is individual packages and filesystem images (jffs2, ext3, etc).

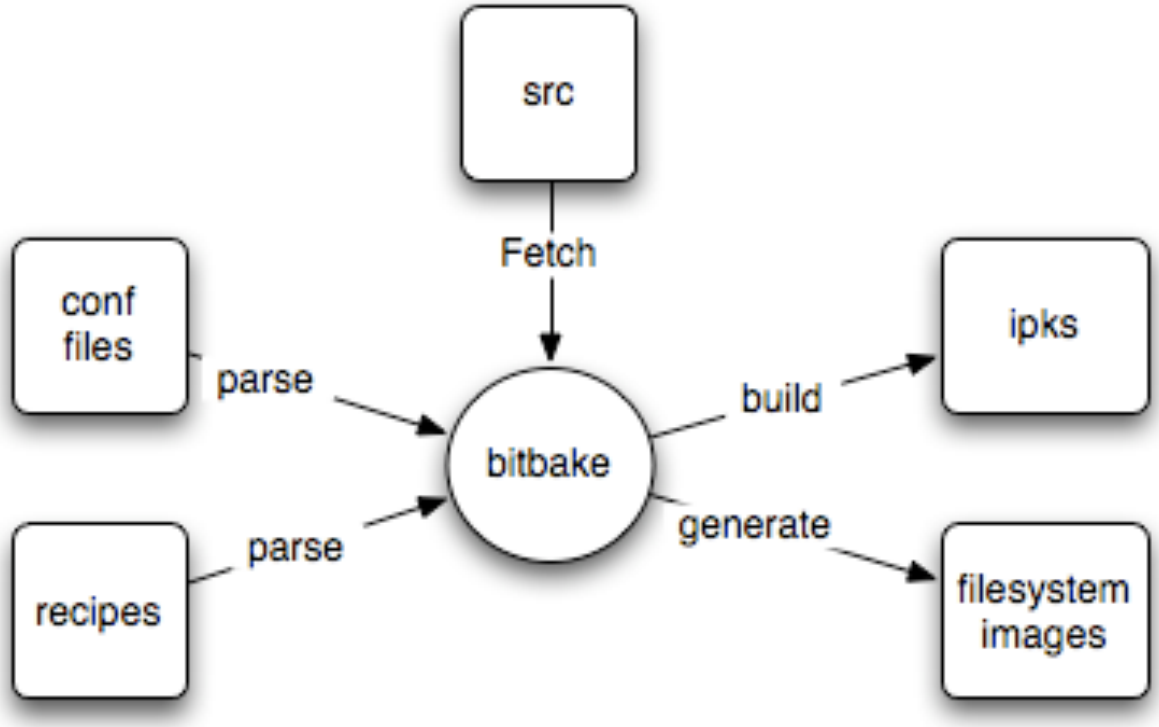
Embedded Alley

OE behavior - build from scratch

- Builds **self contained** build environment from source
- Builds **latest version** of all components unless specificied
- **Downloads** source code from location specified in recipe file (typically from project server on internet)
- Most open source packages built from tarball+patches

OE Metadata and BitBake

OE is powered by **BitBake,** which parses the OE metadata to build the system.



- parses recipes/confs
- Creates a database of how to fetch, configure, build, install and stage each package.
- Determines package dependencies and builds in correct order, parallel where possible
- Uses the IPK packaging format.

Embedded Alley

Digital Photo Frame (DPF)

- Digital Photo Frame (DPF)
  - Typical current embedded Linux application
  - Illustrates use of a varied set of FOSS components
  - Requirements are clear and concise
  - Many people are familiar with DPF device functionality

Embedded Alley

DPF platform

- Hardware assumptions
  - ARM SoC
    - DSP
    - PCM audio playback
    - LCD controller w/ 16-bit color support
    - MMC/SD controller
    - NAND controller
  - 800x600 LCD
  - Small number of navigation buttons
  - MMC/SD slot
  - NAND flash
  - Speakers

Embedded Alley

OE Setup

- Decide on OE metadata version (snapshot or latest)
- Install bitbake
- Setup a pristine OE directory
- Keep changes in an overlay
- Download directory
- Internal mirror
- Changes necessary for commercial product development versus open source project development

OE setup - Overlays

- Bitbake parses all conf and recipes files found in the BBPATH environment variable
- Setup an overlay directory that will hold:
  - specific conf files
  - internal package metadata
  - any overloads on pristine metadata for classes, bb files, conf files
- BBPATH should include the following directories:
  - *openembedded/* - pristine OE metadata
  - *<overlay>/* - custom metadata
- Overlay directory should look like
  - conf/ - custom and overloaded config files
  - packages/ - internal and overloaded package bb files

OE Configuration - Distro

- Configuration files define how the build environment is setup, package versions, information, global inheritance, target boards, final image configuration.
- Four types of configuration files
  - **Distro** - highest level configuration which defines:
    - Toolchain and package versions
    - Package configuration - xserver can be built in several configurations. Distro defines which configuration is built.
    - Sets Distro information variables
    - High level settings such as use udev for device nodes and final image format.

OE Configuration - Distro

```
# Make the most use of our build system
PARALLEL_MAKE = "-j 6"
DL_DIR = "${OEDIR}/sources"
BBFILES := "${OEDIR}/openembedded/packages/*/*.bb ${OEDIR}/ea-oe/
   packages/*/*.bb"

IMAGE_FSTYPES = "jffs2 squashfs tar.bz2"

 # Use EABI ready toolchain
PREFERRED_VERSION_gcc-cross        = "4.1.2"

PREFERRED_VERSION_glibc = "2.5"
PREFERRED_VERSION_uclibc = "0.9.29"
PREFERRED_PROVIDER_virtual/libc = "glibc"
```

Embedded Alley

OE Configuration - Machine

- Machine config files – defines board specific versions and features
  - Architecture
  - Compiler options and other architecture tunables
  - Kernel version and package provider
  - Board specific i/o that require drivers and lib

OE Configuration - Machine

TARGET_ARCH = "arm"

PACKAGE_EXTRA_ARCHS = "armv4t"

require conf/machine/include/tune-arm920t.inc

 # With this kernel version, we can use a newer udev

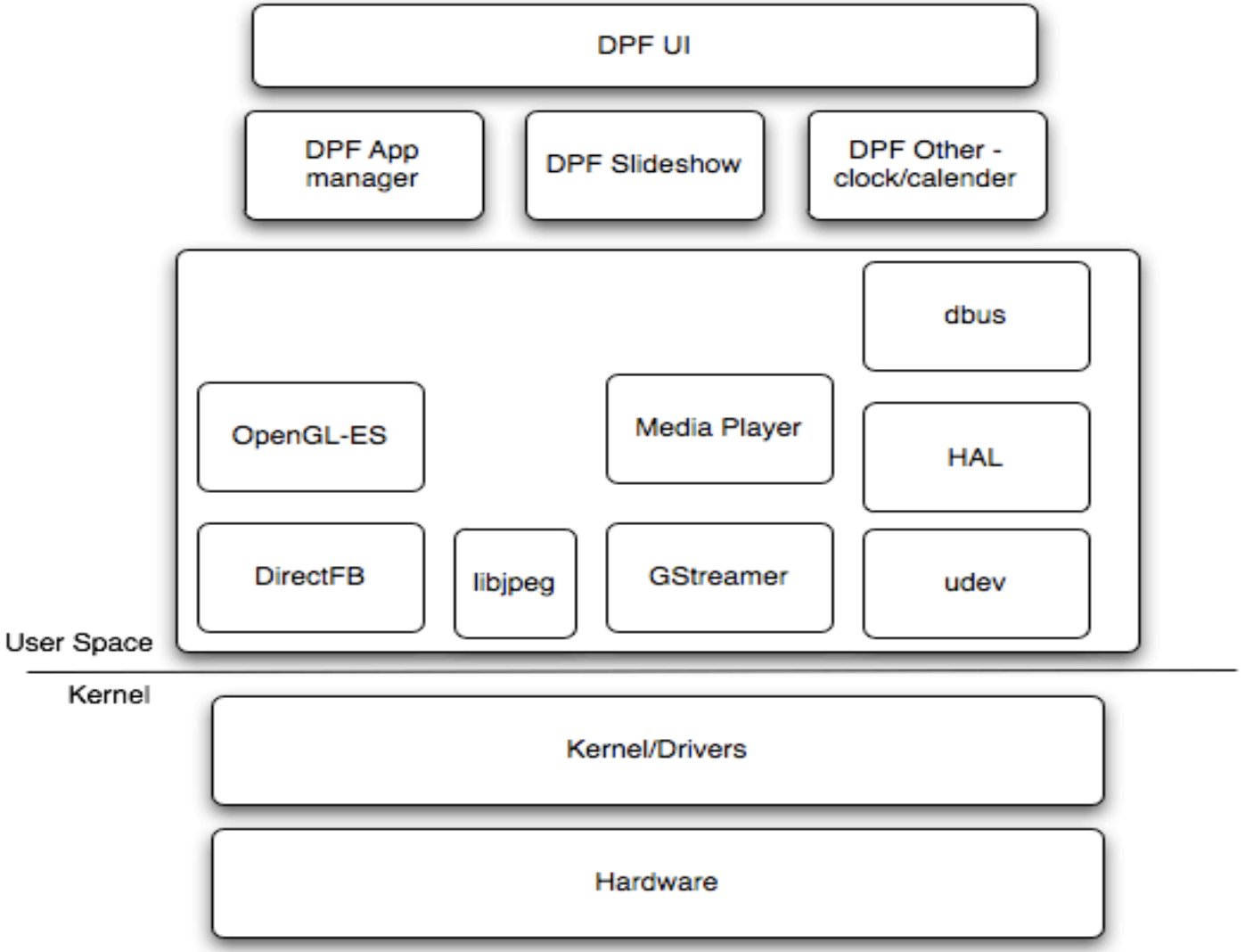
PREFERRED_VERSION_udev = "115"

PREFERRED_PROVIDER_virtual/kernel = "linux-dpfboard"

PREFERRED_VERSION_linux--dpfboard = "2.6.23.14"

DEVEL_FEATURES = "alsa host-usb gadget-usb mtd wifi"

DPF software stack



OE recipe (Overview)

- Bitbake recipe files (.bb)
  - Contain the necessary environment variables, cmds and steps need to build a package
  - Do_fetch, Do_stage(), do_configure(), do_compile(), do_install(), etc.
- Four types of bb files
  - **Classes** - contains common steps for a class of packages.
    - For example, all kernel builds have make, make install, make modules.
  - **Packages** - inherits classes and adds or overrides package specific settings and steps.
    - Defines version and fetcher used to retrieve source
  - **Tasks** - defines the collection of packages to be built
  - **Images** - creates filesystem images out of tasks

OE Tasks

- Divide packages into logical groups
  - enables developers to work on building blocks and easier to manage
  - Separate production and development
- Typical task definitions –
  - Base - the basic user space applications necessary to boot to a prompt. Used for initial debug of system. Glibc, busybox, initscripts, sshd
  - Core – core open source and/or custom applications necessary for the apps (middleware)
  - Apps - Product applications
  - UI – User Interface specific components. Themes, fonts, menu.

OE Tasks - Base

```
RDEPENDS = "\
    ${@base_contains("DEVEL_FEATURES", "alsa", "${ALSA_PKGS}",
     "",d)} \
    base-files base-passwd busybox-devel
    kernel kernel-modules \
    initscripts sysvinit udev \
    ${@base_contains("DEVEL_FEATURES", "mtd", "mtd-utils", "", d)} \
    ${@base_contains("DEVEL_FEATURES", "wifi", "wireless-tools", "",
     d)} \
    dropbear \
"
```

Embedded Alley

OE Tasks - Core

```
RDEPENDS = "\
  directfb \
  hal \
  dbus \
  gstreamer \
  vincent \
  libjpeg \
"
```

OE Tasks - UI

```
RDEPENDS = "\
  dpf-ui \
  dpf-themes \
 dpf-menu \
"
```

OE – Images

- The image file controls what goes into the root file system
- The image types are defined in the distro config file
  - Flash file system for burning to flash
  - Tarball for nfsroot, ramdisk or other development/debug uses
- Separate production and development image
- Root file system is created from packages
- Root file system class controls root file system creation
- Recipes have hooks for extra scripts
- Classes can be overridden

Setup OE for Commercial environment

- Cache copies of open source components as tarballs on a local server
  - OE will wget from a URL
- Lockdown open source component versions in a Bill of Materials conf file that is used by the distro conf file
  - PREFERRED_VERSION_<pkg name>=<version>
- Create internal component metadata to
  - fetch from source control (svn, git, cvs, perforce)
  - Setup variables to control building from tag, branch or head
  - Compile, install, stage and package
- Speed up build
  - Parallel make and multiple bitbake threads controlled by variables
  - Create and distribute prebuilt build environment (SDK)
- Reuse ipk's across machines of the same architecture

Application Development

- Open source applications now building and in a root filesystem
- What about developing the DPF custom applications?
- App developer model
  - Quickly rebuild source with local changes
  - Rebuild source from source control
  - Unit test in a development environment
  - Integrate with rest of the system

Application Development – Two options

- Use OE directly during application development
  - Create bb recipe files for application
  - Keep SCM updated with changes
  - Build using *bitbake <package name>*
  - Integrate by adding package into appropriate task file
- Export SDK from OE
  - Setup OE to export toolchain and libraries to an OE independent environment
  - Build applications from local or SCM sources
  - Integrate into OE when ready

Application Development – BB File

```
DESCRIPTION = "The DPF Media Player"
SECTION = ”dpf/applications"
DEPENDS += "alsa-lib dbus-glib id3lib"
PV = "0.0.1+svnr${SRCREV}"
PR = "r1”
SRC_URI := "${DPF_MIRROR}/src/${DPF_RELEASE}/${SUBDIR};module=${PN};proto=http"
S = "${WORKDIR}/${PN}"

FILES_${PN} += "${datadir}/icons"
```

Embedded Alley

Deployment – Build output

- Build output directories

  cache

  conf – build specific configuration files

  deploy – image and packages

  staging – intermediate install for libraries and headers

  work - build directory

  cross – host tools for target

  rootfs - expanded root filesystem

  stamps

Embedded Alley

Deployment – images and packages

- Deploy directory

  images/ - kernel, bootloader and rootfs images

  ipk/ - all components in a binary package format (ipk)

- Packages can be used to manage software updates or run time configuration changes

Test and Ship your DPF

Embedded Alley

OE Summary

- Very powerful metadata system
- Layered design allows easy customizations and additions
- Supports commercial software development use cases nicely
- Many, many packages already supported
- Can build anything from a complete mobile phone stack to a DVR to a wireless access point stack
  - Maemo, Angstrom, OpenMoko, MythTV, unSlung
- Metadata learning curve is high
- Fairly large open source community using it and maintaining it
- Finding a version of metadata that "just works" can be a challenge

Resources

- http://www.openembedded.org
- http://bitbake.berlios.de/manual/ - manual for the bitbake tool
- http://wiki.openmoko.org - great place to get familiar with how to build a complete software stack with OE

Embedded Alley

Solutions for Intelligent Devices

Contact for more information

Matthew Locke

mlocke@embeddedalley.com