Improving the Embedded Linux Development Workflow

Paul Eggleton
Intel Open Source Technology Centre

ELC 2015 ● San Jose ● 25 Mar 2015

Yocto Project

- **Make embedded Linux easier**

- **Support the OpenEmbedded build system**
  - Cross-compiling build system supporting all of the major arches
  - Can build a Linux-based OS from source (images, packages)
  - Can produce a companion SDK for application development

Users

- **Different roles:**
  - System developer
  - Kernel developer
  - Application developer
  - QA engineer
  - Release engineer
- **System developers are our traditional audience**
- **We don't hear much from application developers**
- **We can do more to help kernel developers**

Current OpenEmbedded SDK

- **Tarball installer**
- **Toolchain (compiler, debugger, misc tools)**
- **Libraries, headers & symbols to match image**
- **Environment setup script**

… and that's pretty much it

To do more, have to install the full build system

OE Build system (for development)

- **Powerful**
  - Great at building a custom OS
  - In-built knowledge of how to build for the target
- **A lot to deal with just for building a single component**
  - Configuration
  - Build time
  - Not very friendly to external source trees (until 1.8)

User research

- Needed some real-world usage information
- User research (and design in general) is not just applicable to software that has a GUI
- Semi-structured interviews

User research – feedback received

- Hard to add new libraries to the SDK
- Hard to keep the SDK on developer machine up-to-date with the rest of the OS
- Sometimes developers do have to deal with the build system, though they would rather avoid it
- Perception that OpenEmbedded is great for system developers, not so much other developers
- SDK isn't much help when you want to modify an existing component
- We really need a basic workflow defined (but a rigid solution won't work)

Aims

- Shorten the code->test->debug cycle
- Install SDK once, update as needed
- Extend SDK on-demand
- Allow developers to work together more closely
- Provide tools to ease integration into final image

New: Extensible SDK

- **All of the capabilities of the existing SDK**
- **Additional tooling to:**
  - Ease addition of new apps & libraries
  - Allow modifying source of an existing component
  - Test changes on the target
  - Ease integration into the rest of the build system

Adding a new application

1. **Add application to workspace:**
   `devtool add myapp /path/to/source`
2. **Build it:**
   `devtool build myapp`
3. **Write to target device (w/network access):**
   `devtool deploy-target myapp device`
4. **Edit source code & repeat steps 2-3 as necessary**

Demo – add a component

How does it work?

- **Pre-packaged adaptation of the build system**
  - Preconfigured, locked down, prebuilt artifacts
  - Wrapped in a friendly tool rather than using directly
  - All of the collected build intelligence
- **“devtool add” creates a recipe**
  - May need additional editing, hopefully not too much
  - Basis for integration
  - Further work to be done in this area
- **Improvements on the build system side as well**
  - Better external source tree support

We can help system developers too

- **Some of the same kinds of tasks:**
  - Also need to build new software
  - Need to modify existing components - add a patch, fix a bug, etc.
  - Need to update the recipe with changes
- **devtool also available next to the build system**

Modifying a component

1. **Extract source and add recipe to workspace:**
   `devtool modify -x` *`recipename /path/for/source`*
2. **Edit source code**
3. **Build it:**
   `devtool build` *`recipename`*
4. **Write to device (w/network access):**
   `devtool deploy-target` *`recipename device`*
5. **Repeat steps 2-4 as necessary**
6. **Either push source code changes, or write as patches on top of recipe:**
   `devtool update-recipe` *`recipename`*

Demo – modify a component

Target deployment

- Currently done via SSH
- Deploys everything installed at the do_install step
- Knows what files to deploy & what was deployed previously
- Don't need package management on target

Recipe creation

- **Used by `devtool add`**
- **Or standalone with `recipetool create`**
- **Looks at source, currently without building**
  - Scans for licenses
  - Determines build system (autotools, cmake, …)
  - Extra support for creating Python module recipes
- **Easy to extend**

Kernel development

- **Want to be able to work on the kernel as with other recipes**
- **devtool can extract the kernel source and build it**
  - linux-yocto has its own patch process
  - Some tweaks to the external source tree support that help use with the kernel
  - Can't generate configuration yet
- **Kernel build performance improvements in 1.8**

External source tree support

- do_compile now runs every time, no need to force/clean
- Recipes with local files in SRC_URI (e.g. config files) now work
- Other minor fixes

Current status (upcoming 1.8 release)

- **Initial version of command-line tools**
  - devtool: add, modify, update-recipe, deploy-target...
  - recipetool: basics; python
- **Proof-of-concept extensible SDK**
  - No update capability yet
- **Basic kernel support**
  - Missing configuration

Future (1.9+)

- Full support for kernel development
- SDK publishing / update capability
- make/make install integration (no recipe)
- Eclipse integration
- Wizard-based recipe creation, look at build output
- Support for submitting changes
- Further research and discussion

Conclusions

- **We are listening**
- **We're developing tools for a broader user base**
- **Try the tools, send us feedback**
  - Tell us about your workflow and any stumbling blocks

Questions?

Thanks!

Additional thanks to:
Belen Barros Pena, Richard Purdie, Randy Witt, Chris Larson, Chen Qi, Trevor Woerner, Junchun Guan