YOCTO PROJECT* EXTENSIBLE SDK
SIMPLIFYING THE WORKFLOW FOR APPLICATION DEVELOPERS

Henry Bruce
Intel Open Source Technology Center
WHAT WE’LL BE COVERING

Why do we need the Extensible SDK?
How do distro maintainers create one?
How does the Extensible SDK improve workflow?
Not a devtool deep dive
WHAT IS THE EXTENSIBLE SDK
THE STANDARD SDK

Develop code to run on target machine

Delivered as a shell script installer

• Cross-Development toolchain
• Target libraries, headers, and symbols
• Environment setup script
WHAT COULD BE IMPROVED?

Updateable
Extensible
Packages
Images
Teamwork
MEET THE EXTENSIBLE SDK

Major design change
  Shrink-wrapped distro maintainer environment
Allow compact installer (as small as 35MB)
Updateable and extensible – lazy install
Simplifies team-work
First appeared in Jethro, usable in Krogoth
USING THE EXTENSIBLE SDK
INSTALL

Installer created by distro maintainer

Shell script installer

Completely self contained

Can be installed to an arbitrary location
INSTALL EXAMPLE

poky-glibc-x86_64-core-image-minimal-i586-toolchain-ext-2.2.1.sh
$ chmod +x poky-glibc-x86_64-core-image-minimal-i586-toolchain-ext-2.2.1.sh
$ ./poky-glibc-x86_64-core-image-minimal-i586-toolchain-ext-2.2.1.sh -d mysdk

Poky (Yocto Project Reference Distro) Extensible SDK installer version 2.2.1
You are about to install the SDK to "/home/hbruce/mysdk". Proceed[Y/n]? Y
Extracting SDK.... done
Extracting buildtools...
Preparing build system...
Parsing recipes: 100% |###########################################| Time: 0:00:17
SDK has been successfully set up and is ready to be used.
Each time you wish to use the SDK in a new shell session, you need to source the environment setup script e.g.
   . /home/hbruce/mysdk/environment-setup-i586-poky-linux
Assumes SDK contains toolchain

$ . /home/hbruce/mysdk/environment-setup-i586-poky-linux

$ $(CC) hello.c -o hello

Also enables Eclipse* integration

Keep updatability but act like standard SDK

*Other names and brands may be claimed as the property of others.
ACCESS TO NEW TOOLS

Most of the distro maintainer tools

• devtool
• recipetool
• wic
• ...

EXAMPLE DEVTOOL WORKFLOW

$ . /home/hbruce/mysdk/environment-setup-i586-poky-linux
$ devtool add https://github.com/whbruce/bbexample.git
$ devtool edit-recipe bbexample
$ devtool build bbexample
$ devtool deploy-target bbexample root@192.168.0.2

Run, fix, repeat.

$ devtool finish bbexample /path/to/bitbake/layer

Issue pull request

Maintainer builds and publishes updated eSDK with your work in it
UPDATING AND EXTENDING THE SDK

Update
   $ devtool sdk-update [url]

Extend
   $ devtool search package-name
   $ devtool sdk-install [-s] recipe-name

Example
   $ devtool sdk-install meta-extsdk-toolchain
RUN IN A CONTAINER

A Docker* container is available

https://hub.docker.com/r/crops/extsdk-container

Enables operation on macOS* and Windows*
CREATING AND MAINTAINING AN EXTENSIBLE SDK

Tasks for the distro maintainer
GENERATING SDK: BUILDING

Build

```
$ bitbake image-recipe -c populate_sdk_ext
```

Default contains everything for image recipe

SDK installer in tmp/deploy/sdk

Make installer available to your developers
CONFIGURE ESDK: INSTALLER AND CONTENT

Make a minimal installer (full by default)
SDK_EXT_TYPE = "minimal"

Add toolchain to minimal installer
SDK_INCLUDE_TOOLCHAIN = "1"

Add all package info
SDK_INCLUDE_PKGDATA = "1"

Add specific tools or libraries
SDK_EXTRATOOLS += "nativesdk-package-name"
TOOLCHAIN_HOST_TASK += "package-name"
Define a URL for the updater, say http://mysite.com/sdk-updater

Set default update URL in your configuration

```
SDK_UPDATE_URL = "http://mysite.com/sdk-updater"
```

When you make change, re-build SDK and generate updater files

```
$ oe-publish-sdk tmp/deploy/sdk/installer.sh /path/to/sdk-updater
```

Let your developers know when updates are available
FASTER BUILDS THROUGH SHARED STATE

Pre-built objects on a server
eSDK checks if you can re-use objects
Qemu86 core-image-minimal build times
  • Without: 35mins
  • With: 1 min
**CONFIGURE SDK: SHARED STATE MIRROR**

Define URL of the shared state mirror server.
http://mysite.com/sstate-cache

Set in `conf/sdk-extra.conf`

```conf
SSTATE_MIRRORS = "file://.*_http://mysite.com/sstate-cache/PATH"
```

Ensure sstate-cache is served up at this location
Bubba makes eSDK
Bubba makes eSDK

1. Bubba makes eSDK
2. Joe updates eSDK
1. Bubba makes eSDK
2. Joe updates eSDK
3. Uses “devtool add” to generate recipe
1. Bubba makes eSDK
2. Joe updates eSDK
3. Uses "devtool add" to generate recipe
4. Build and debug sensor Lib
1. Bubba makes eSDK
2. Joe updates eSDK
3. Uses “devtool add” to generate recipe
4. Build and debug sensor Lib
5. When Lib is ready “devtool finish” finalizes recipe (and patches), and sends recipe to Bubba
1. Bubba makes eSDK
2. Joe updates eSDK
3. Uses “devtool add” to generate recipe
4. Build and debug sensor Lib
5. When Lib is ready “devtool finish” finalizes recipe (and patches), and sends recipe to Bubba
6. Bubba “turns the crank” for continuous integration process
1. Bubba makes eSDK
2. Joe updates eSDK
3. Uses “devtool add” to generate recipe
4. Build and debug sensor Lib
5. When Lib is ready “devtool finish” finalizes recipe (and patches), and sends recipe to Bubba
6. Bubba “turns the crank” for continuous integration process
7. Bubba publishes “revised” eSDK with Joe’s sensor Lib
1. Bubba makes eSDK
2. Joe updates eSDK
3. Uses “devtool add” to generate recipe
4. Build and debug sensor Lib
5. When Lib is ready “devtool finish” finalizes recipe (and patches), and sends recipe to Bubba
6. Bubba “turns the crank” for continuous integration process
7. Bubba publishes “revised” eSDK with Joe’s sensor Lib
8. Laurie and Betsy update their eSDKs
1. Bubba makes eSDK
2. Joe updates eSDK
3. Uses "devtool add" to generate recipe
4. Build and debug sensor Lib
5. When Lib is ready "devtool finish" finalizes recipe (and patches), and sends recipe to Bubba
6. Bubba "turns the crank" for continuous integration process
7. Bubba publishes "revised" eSDK with Joe’s sensor Lib
8. Laurie and Betsy update their eSDKs
9. They use "devtool add" to generate recipes; Laurie for a Python app, Betsy for a Node.js app
1. Bubba makes eSDK
2. Joe updates eSDK
3. Uses “devtool add” to generate recipe
4. Build and debug sensor Lib
5. When Lib is ready “devtool finish” finalizes recipe (and patches), and sends recipe to Bubba
6. Bubba “turns the crank” for continuous integration process
7. Bubba publishes “revised” eSDK with Joe’s sensor Lib
8. Laurie and Betsy update their eSDKs
9. They use “devtool add” to generate recipes; Laurie for a Python app, Betsy for a Node.js app
10. They build and debug their apps
When apps are working, they use “devtool finish” and send recipes to Bubba.
1. Bubba makes eSDK
2. Joe updates eSDK
3. Uses “devtool add” to generate recipe
4. Build and debug sensor Lib
5. When Lib is ready “devtool finish” finalizes recipe (and patches), and sends recipe to Bubba
6. Bubba “turns the crank” for continuous integration process
7. Bubba publishes “revised” eSDK with Joe’s sensor Lib
8. Laurie and Betsy update their eSDKs
   They use “devtool add” to generate recipes; Laurie for a Python app, Betsy for a Node.js app
9. They build and debug their apps
10. When apps are working, they use “devtool finish” and send recipes to Bubba
11. Bubba “turns the crank” and ships the product image
WRAP UP
EXTENSIBLE SDK BENEFITS

Shared development environment
Power of devtool
Improved team workflow
Leverages shared state for faster builds
Runs on a range of host operating systems
THANKS

Paul Eggleton
Randy Witt
Brian Avery
Doug Martin
CALL TO ACTION

Use the Extensible SDK
Use it on Windows and Mac

https://wiki.yoctoproject.org/wiki/Extensible_SDK
Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
© Intel Corporation