Best Practice for Tizen Platform from Code to Device

Zhang, Qiang  Chen, Gui
(Intel Open Source Technology Center)
Agenda

- Tizen Development Tools
- Workflow Overview
- Downloading Tizen source code
- Building Package with GBS
- Creating image with MIC
- Flash to device with Ithor
- Experimental data
Tizen Development Tools
GBS – Git Build System

• **What is GBS**
  • Command line tool that supports developing Tizen package maintained with git.

• **Features**
  • build/remotebuild
  • submit
  • import/export
  • clone/pull
  • chroot
  • changelog
MIC – Mic Image Creator

• What is MIC
  • An image creator to create images for Tizen.

• Features
  • supported image types: loop, raw, fs, livecd, liveusb
  • create (cr): create an image
  • convert: convert image format from one type to another
  • chroot: chroot to an image
Others Tools

• **git**
  • An open-source version-control system
  • Handle large projects that are distributed over multiple repositories

• **repo**
  • First introduced in Android from Google
  • A repository management tool that we built on top of Git
  • Make it easier to work with multiple Git projects, sync / uploads codes.

• **Ithor**
  • Flash tool for Tizen Phone
  • Support Linux and Windows
How to Install

- **Tizen Development Tools repo**

- **Supported Distributions**
  - Ubuntu 12.04, 12.10, 13.04
  - Fedora 17, 18
  - openSUSE 12.1, 12.2, 12.3
  - CentOS 6
Workflow Overview

Remote Gerrit

Local Tizen Source Code

Manifest

Remote Repository

Local repository

GSB

MIC

Lthor

KS file
Download Tizen Source Code
Git Hosting

• Gerrit:
  • A web-based code review system for projects that use git
  • Tizen Gerrit URL: https://review.tizen.org/gerrit
  • Register: https://www.tizen.org
  • Follow guide to upload ssh public key

• Code clone
  • git clone
  • repo & manifest (recommended)

• Examples:
  • $ ssh review.tizen.org -p 29418 gerrit ls-projects
  • $ git clone ssh://review.tizen.org:29418/apps/home/memo
Prepare Manifest

• Sample manifest file

```xml
<?xml version="1.0" ?>
<manifest>
  <remote name="tizen.org" fetch="ssh://tizen/" review="https://review.tizen.org/gerrit" />
  <default revision="tizen_2.1" remote="tizen.org" sync-j="4" />
  <project name="apps/home/app-selector" path="apps/home/app-selector" />
  <project name="apps/home/calculator" path="apps/home/calculator" />
...
</manifest>
```

• Get full released manifest.xml
  • http://download.tizen.org/releases/2.1/tizen-2.1/builddata/manifest

• Prepare local manifest git repo

```
$ mkdir ~/tizen-manifest
$ git init
$ cp path/to/tizen2.1.xml .
$ git add tizen2.1.xml && git commit -a -m "initial version"
```
Downloading Tizen source

• **Download repo script**
  - [https://dl-ssl.google.com/dl/googlesource/git-repo/repo](https://dl-ssl.google.com/dl/googlesource/git-repo/repo)

• **Download Tizen source code with repo**

```bash
$ repo init -u ~/manifest/ -m <manifest file>
$ repo sync
```
Build Tizen Source using GBS
Prepare gbs conf file

- Create a new profile if needed
- Add Tizen repo to configuration file, Example:

```
[general]
profile = profile.tizen2.1
...
[repo.tizen2.1]
url= http://download.tizen.org/releases/2.1/tizen-2.1
# Comments out the following two lines if auth needed
# user=
# passwd=

[profile.tizen2.1]
repos=repo.tizen2.1
```

- use `--profile|--P` to use the profile above, like
  - $ gbs build -P tizen2.1 ...
  - -P option is not required if the profile is the default profile in [general] section
Build packages with GBS

• Break dependencies circles
  • 'gbs build' can report circles if exist, for example:
    cmake->curl->c-ares->cmake
    gettext->gcc->eglibc->gettext
  • use --exclude option to break circle: --exclude=cmake,gettext

• Other useful options:
  • --threads, --arch, --overwrite, -dist|-D, ...
  • More: gbs build --help

• Full gbs build command
  
  $ gbs build -A <i586|armv7l> --threads=4 --exclude=cmake,gettext
Output of gbs build

```
gbs output top dir # Default is ~/GBS-ROOT/
|-- local
 | |-- cache # repodata and RPMs from remote repositories
 | |-- repos # generated local repo top directory
 | | |-- tizen2.0 # repo for profile ivi
 | | | |-- armv71 # store armv71 RPM packages
 | | | |-- i586 # store i586 RPM packages
 | | |-- tizen2.1 # build for profile: tizen2.1
 | | `-- i586 # the same as above
 | |-- BUILD-ROOTS # top dir to store all kinds of build roots
 | | |-- scratch.i586.0 # first build root for x86 build
 | | |-- scratch.i586.1 # second build root for x86 build
 | | |-- scratch.i586.2 # third build root for x86 build
 | | `-- scratch.armv71.0 # first build root for armv71 build
 | `-- sources # The above build root dir can be used by gbs chroot
 | |-- tizen # sources generated for build, including tarball, spec, etc.
 | `-- tizen2.1
`-- meta # meta data used by gbs in the future
```
Create image using MIC
Customize KS file

- Get original ks file
  - Download from released remote repo:

- Customize ks file
  - Add local repo created by gbs
  - Add priority for repo if needed
  - Add extra packages/groups
  - Remove packages / groups
  - Update post scripts
Sample KS file

```bash
# -*-mic2-options-*- -f loop --pack-to=@NAME@.tar.gz -*-mic2-options-*-
lang en_US.UTF-8
keyboard us
timezone --utc America/Los_Angeles
# ROOT fs partition
part / --size=3000 --ondisk mmcblk0p --fstype=ext4 --label=platform
# DATA partition

# Add local repo generated by gbs
repo --name=local --baseurl=file:///home/<user>/GBS-ROOT/local/repos/tizen2.1/i586 --priority=1
repo --name=base \n   --baseurl=https://download.tizen.org/releases/2.1/tizen-2.1/repos/tizen-base/ia32/packages/ \n   --save --ssl_verify=no
repo --name=main \n   --baseurl=https://download.tizen.org/releases/2.1/tizen-2.1/repos/tizen-main/ia32/packages/ \n   --save --ssl_verify=no
%packages
...
%end
%prepackages
...
%end
%post
...
%end
```
Create image using mic

- **Options for image creation**
  - `mic cr --help` or `mic cr <image type> --help`
  - **Image types supported:** loop, raw, fs, livecd, liveusb
  - **special type:** auto
- **Basic usage of mic**
  - `$ mic cr auto <tizen.ks>`
  - `$ mic cr loop --pack-to=@NAME@-rs.zip <tizen.ks>`
Flashing image with Ithor

- **Boot into download mode**
  - Power off
  - Press `<volume down>` + `<power>` keys
- **Flash system (kernel, u-boot) image**
  - Get me here: [http://download.tizen.org/releases/system/](http://download.tizen.org/releases/system/)
  - `$lthor <system.tar>`
- **Flash rootfs image**
  - rootfs: `platform.img`, `data.img`, `ums.img`
  - `lthor <tizen.tar.gz>`
## Experimental work

<table>
<thead>
<tr>
<th>Test Machine</th>
<th>Test Data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong></td>
<td><strong>i7, 3.40GHz, 8 cores</strong></td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td><strong>8G</strong></td>
</tr>
<tr>
<td><strong>Swap</strong></td>
<td><strong>8G</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Build Time</th>
<th><strong>i586</strong></th>
<th><strong>armv7l</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>gbs build</td>
<td>export package</td>
<td>10 m 30 s</td>
</tr>
<tr>
<td></td>
<td>build package</td>
<td>5 h 10 m</td>
</tr>
<tr>
<td>Create image</td>
<td></td>
<td>38 m 29 s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Build Time (optimized)</th>
<th><strong>i586</strong></th>
<th><strong>armv7l</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>gbs build</td>
<td>export package</td>
<td>10 m 30 s</td>
</tr>
<tr>
<td></td>
<td>build package</td>
<td>2 h 30 m</td>
</tr>
<tr>
<td>Create image</td>
<td></td>
<td>4 m 39 s</td>
</tr>
</tbody>
</table>

---

**GBS:** mount gbs build root dir as tmpfs

```
$ mount -t tmpfs -o size=16G tmpfs ~/GBS-ROOT/local/BUILD-ROOTS
```

**MIC:** add `--tmpfs` option

```
$ mic cr --tmpfs ...
```
References

• [1] https://source.tizen.org/documentation/articles/creating-tizen-platform-image-scratch
Q&A