Top 3 pains in professional use of bitbake

Klaas van Gend
Senior Solutions & Services Architect
Distribution License

- This work (except the cartoons) is licensed under the Creative Commons Attribution-Noncommercial -Share Alike 3.0 Unported License.

To view a copy of this license:
- Visit http://creativecommons.org/licenses/by-nc-sa/3.0/
- Send a letter to:
  Creative Commons
  171 Second Street, Suite 300
  San Francisco, California, 94105
  USA

Images, cartoons and logos are © the respective artists.
Who’s Klaas???

(this slide intentionally left empty)
Who are You and why are you here?

• You have been using (Embedded) Linux for years
• You have used OpenEmbedded (or derivatives) and were not entirely satisfied or worse
• You are an OpenEmbedded developer

If you do not qualify for more than 60% of the statements, this talk may make little sense to you!
Development Life Cycle – “utopia”

Time progresses
Development went smoothly, until…

And we’re going to use embedded Linux…

integrate
Embedded Linux?
The Three B’s

• **Buy?**
  Go to a commercial Linux vendor

• **Build?**
  Develop yourself

• **Borrow?**
  Use Open Source
What software packages to select?  
From which source?

- System Software is usually more than one application
- Everything has been implemented before
  - Where to find them? How to build?
- Division between kernel space and user space
  - Driver design – debugging, performance, licensing
- Software Licensing
What package versions to use? And what are dependencies?

• Nearly every open source project uses SCM
  • Stable / unstable / CVS / alpha / beta?
  • Is the project alive? Is that version community supported?
  • When to freeze your platform?

• Packages require features from other packages…
  • “Dependency Hell”
  • 12 different programming languages on a system?

• How to compare packages?
The Solution:

Bitbake + OpenEmbedded?
“Everybody” uses it:

- OpenZaurus
- OpenMoko
- PokyLinux
- Ångström
- KOAN software
- Beagleboard.org

So it must be good, right?
How does it work (in 1 slide)?

1. Figures out build order

2. Downloads & patches source code

3. Cross-compiles source & builds packages

4. Builds image(s)
The pain
Yawn…

And then… the long wait…

827 tasks…

Build host support:
- **native** perl, curl, python, perl, OpenSSL, GTK-doc, bzip2, libxm2, ncurses
- **cross** gcc, binutils
- But the sanity check required diffstat and help2man on my host?

On target FS:
- hello-world (statically linked)
- sysvinit
- udev
- libc + supporting libs
- libcurl
- Opkg

Much more than expected…
Fetching or compiling

Bitbake:
• Sequentially
• Per package

System is idle during fetch
• Unless (undocumented?):
  • BB_NUMBER_THREADS
  • PARALLEL_MAKE="-jX"
But that’s not for the newbie…

“404 errors”:
• Very annoying if a download fails just after you went home!
• Why 5x re-try down mirrors
• Why try all Debian local mirrors?
What does a newbie want?

- It must work the first time
  - no errors – newbie cannot solve them yet!

- Usually time pressure
  - Has to deliver a first working platform to team
  - And has to prove himself 😊

- Contains “all he needs”
  - For a first image, helloworld-image is a good start!
  - It contains much more – but no ssh client or shell 😞

Building from source?

- Just putting binaries together works for newbie!
- Koen’s Angstrom on-line builder:
  http://amethyst.openembedded.net/~koen/narcissus/
• The resulting image for `helloworld-image` is 5.9MB...
• And we don’t have a shell nor ssh yet
• 85 `*–image.bb` files in `recipes/images`
  • Reading other image recipes and `classes/image.bbclass` really helps understanding the mechanism
  • Reading the (fine!) OpenEmbedded manual helps

• Now poor newbie will have to start creating his own image file and build it.
What does a newbie want? – part II

- Size not so relevant during prototyping
- NFS mount ! ?
- But easily reduce image size later in the process
  - No man, info, doc
  - Debugging symbols / strip
  - Remove packages
    - Dependency hell
    - Graphviz ?
  - Reduce/Remove locales (IMAGE_LINGUAS)

- Good tools would really help here
  - List all dependencies
  - List what is in an image
  - List what needs to built on the host
Reproducability

`git clone
   git://git.openembedded.net/openembedded`

- Try to rebuild exact platform a year later?
- Try to rebuild a day later on another machine?

- Solution:
  (keep monitoring for bugfixes!)
Upstream communities move forward, too
• Not every project keeps all releases around
  • Example: fakeroot
• Repositories move

Same solution applies:
• Locally mirror all “pristine sources” + patches
• Do this for the whole repository!
• Not trivial!
  • svn: cvs: http: https: ftp:
• Talk to your SCM administrator before committing many MBs of tar.gz files!
More reproducibility worries

- **Rebuild a single package**
  - Useful if you are modifying that package’s configuration
  - (OE manual page 35):
    - `bitbake -b <bb-file> -c clean`
    - `bitbake -b <bb-file>`
  - Does not inherit everything previously built

- **Does it matter?**
  - Not if you have nightly builds ?!?!
  - Don’t tell your boss…
Dependencies

In recipes:

- **DEPENDS** - must be present at *compile time*
- **RDEPENDS** - must be present at *run time*
  - Usual shared libraries are detected automatically: “shlibs”
  - Only dlopen()’ed libs in RDEPENDS
  - Loose coupling: `wvdial` depends on `ppp`
  - Unrelated packages, e.g. font files
- **RRECOMMENDS**
  - lists recommendations that can be overridden by the user
- Also: **RCONFLICTS and RREPLACES**

Problem:

- Full dependency information only available *after* building of the image…
Community direction?

- Where is the community going?
- Which community?
Back to 10,000 ft…
Conclusions?

• In many projects, embedded Linux is not an thorough engineering decision
  • Though it fits nicely!

• OpenEmbedded + Bitbake fits professional use better than “Linux from Scratch”
  • Re-use of community knowledge
  • Wide variety of packages available
  • Community is large and active

• Bitbake has steep learning curve
  • Less of a problem if your boss pays for time and you do not have deadline tomorrow
## Summary: The Top 3 Pains

<table>
<thead>
<tr>
<th>#</th>
<th>Pain</th>
<th>Cause</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>404 on repositories and/or recipes</td>
<td>“Community”</td>
<td>• Local mirrors</td>
</tr>
<tr>
<td>2</td>
<td>Dependency hell</td>
<td>Community</td>
<td>• Know your components</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Start small</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• A graphical tool would help</td>
</tr>
<tr>
<td>3</td>
<td>Reproducability, speed and caching</td>
<td>Bitbake</td>
<td>• Keep local repositories</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Cache builds</td>
</tr>
<tr>
<td>4</td>
<td>Divergent communities</td>
<td>Community</td>
<td>• Switch to Windows CE?</td>
</tr>
</tbody>
</table>

Final statement: bitbake is “great”, but everybody else is making it a pain???