Managing kernel modules with kmod

Lucas De Marchi
ProFUSION Embedded Systems

Embedded Linux Conference 2012
Redwood Shores, CA

Who am I?

- Software Engineer at ProFUSION
- Contributor to some open source projects: Kernel, BlueZ, oFono, ConnMan, EFL, WebKit
- Creator of others: dietsplash, codespell, and... **kmod**
- http://www.politreco.com/
- IRC: demarchi

Outline

Introduction

About

What's kmod?

"The goal of the new library libkmod is to offer to other programs the needed flexibility and fine grained control over insertion, removal, configuration and listing of kernel modules."

Lucas De Marchi -- announcement of kmod 1

About - Why?

What's wrong with module-init-tools (m-i-t)?

"module-init-tools: provide a proper libmodprobe.
so from module-init-tools:
Early boot tools, installers, driver install disks
want to access information about available
modules to optimize bootup handling."

Plumber's Wish List for Linux - October, 2011

That means: **udev**, **systemd**, initrd tools and others

About - Why?

What's different from m-i-t?

- Library is designed first

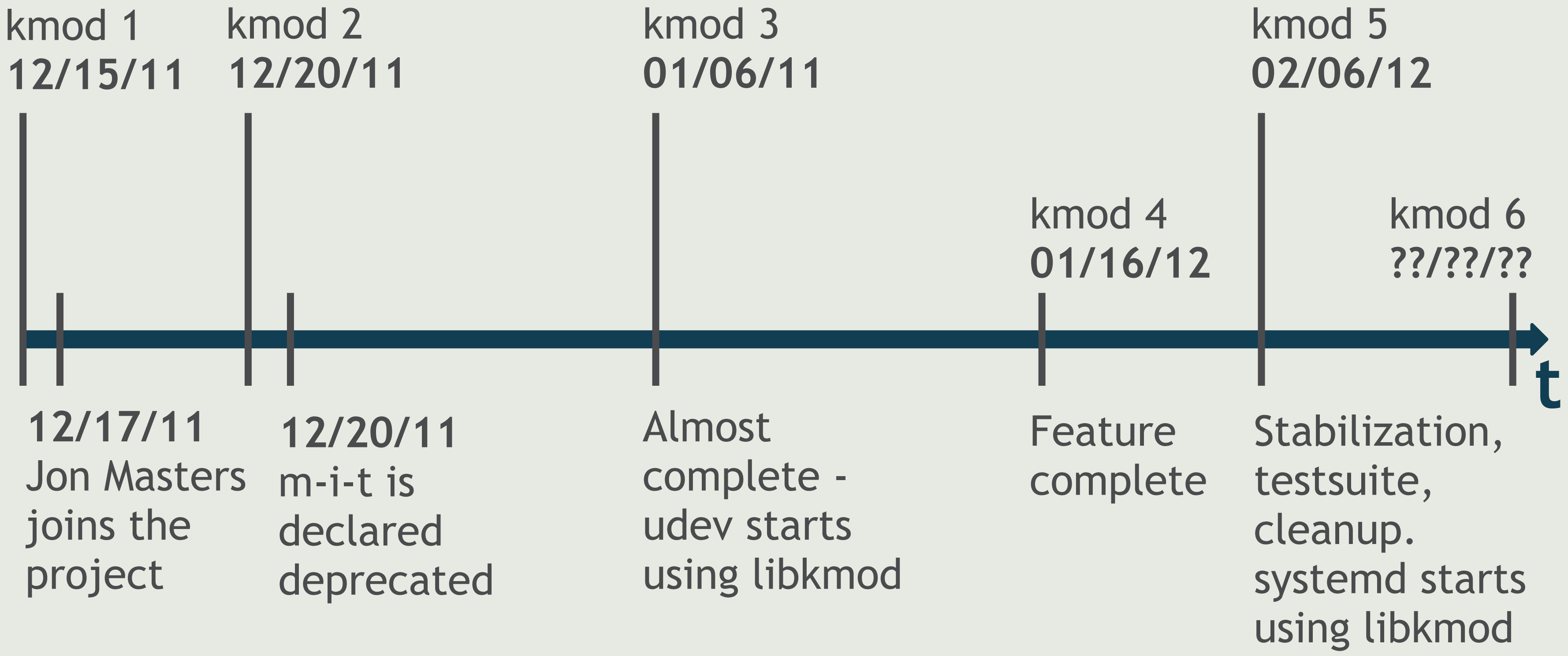
  Initial goal was to export only part of the needed functions **(libmodprobe.so)**, later we decided to export all of them **(libkmod.so)**

  Based on **libabc** (See Kay's e Lennart's talk at Kernel Summit 2011)

- Tools are created on top of the library

  Project is renamed to **kmod**

History



kmod 1
12/15/11

12/17/11
Jon Masters joins the project

kmod 2
12/20/11

12/20/11
m-i-t is declared deprecated

kmod 3
01/06/11
Almost complete - udev starts using libkmod

kmod 4
01/16/12
Feature complete

kmod 5
02/06/12
Stabilization, testsuite, cleanup. systemd starts using libkmod

kmod 6
??/??/??

t

How module management works on Linux

Module management

module insertion:

```
long init_module(const void *mem,
                 unsigned long len,
                 const char *args)
```

module removal:

```
long delete_module(const char *name,
                   unsigned int flags)
```

module list, params, state:

```
/sys/module
```

Module management

- Pretty simple interface with kernel, but...
- Much more complicated when all use cases must be handled:
  - Hotplug (resolving aliases)
  - Blacklist
  - Dependencies and soft-dependencies
  - Install and remove commands
  - ELF tweaking

Dependencies

- A module may depend on symbols from another module
- Too heavy to check dependencies at insertion time
  - Offload calculation: **depmod**
  - Read dependencies info and do TheRightThing®

Dependencies - depmod

- Read .symtab and .ksymtab sections of each module
- Match who provides a symbol with who requires a symbol
- Calculate dependencies (topological sort) and write modules.dep.

```
    kernel/drivers/bluetooth/btusb.ko: \
                        kernel/net/bluetooth/bluetooth.ko
```

- modules.dep.bin: same information, but stored in a Trie

Dependencies - depmod

- Actually it does a bit more. Indexes:
    - `modules.alias{,.bin}`
    - `modules.dep{,.bin}`
    - `modules.devname`
    - `modules.softdep`
    - `modules.symbols{,.bin}`
- All indexes are saved per-kernel:
    - `/lib/modules/$(uname -r)`

Dependencies - modprobe

- Basically it reads dependencies and load modules in the right order
- Configurations:
  - Blacklist
  - Alias
  - Install and remove commands
  - Options
  - Softdeps

Dependencies - modprobe

- --force-modversion, --force-vermagic, -f

  These are the bad guys
- Kernel refuses to load modules with mismatching versions. It checks the `.modinfo` section (the same read by modinfo)
- Solution: remove that information from module before handing over to kernel

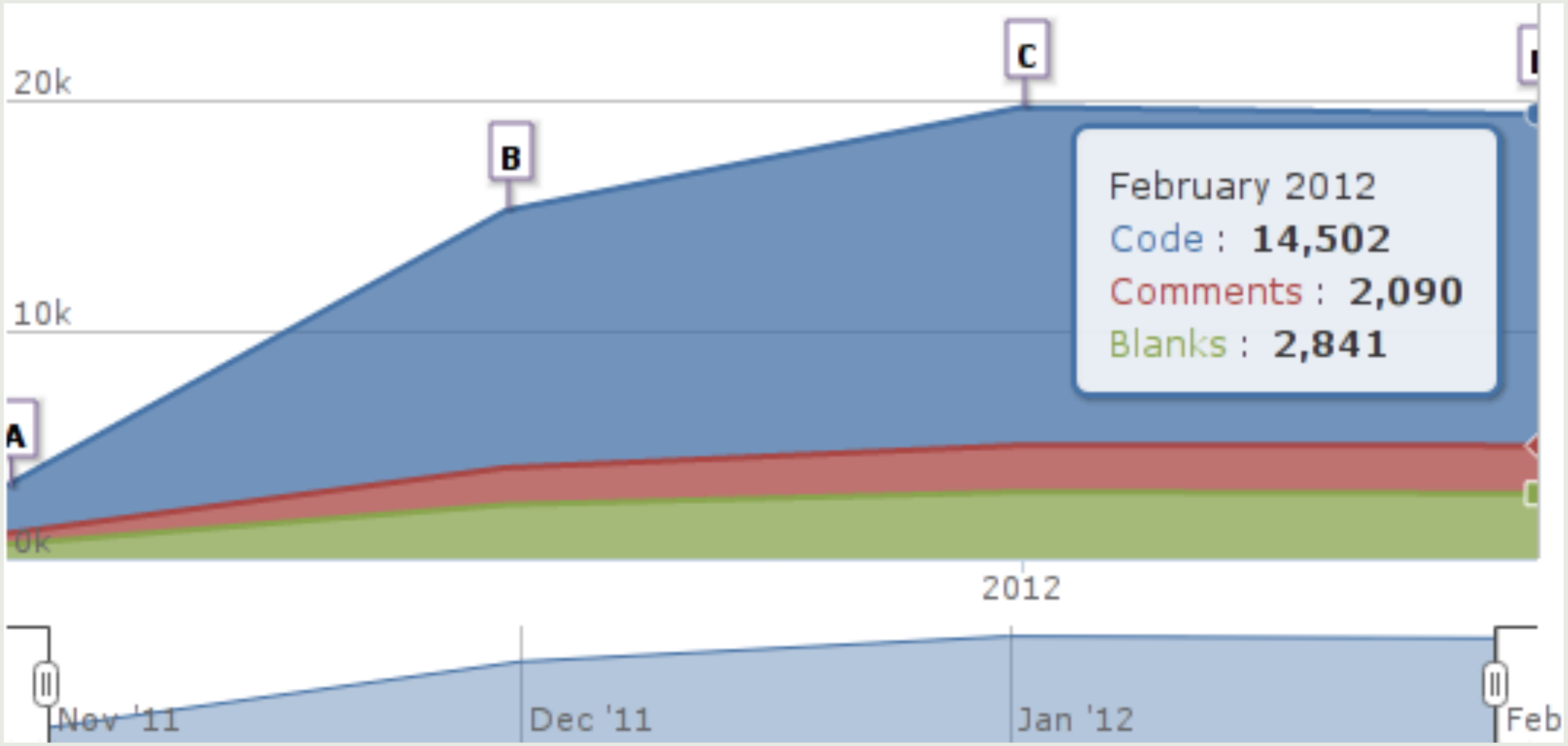
Current status

m-i-t phase out plan

1. Put all (part) of the features inside a library
2. Port all (part of the) tools to use the library
3. Allow library to be installed in parallel to m-i-t
4. Allow kmod to completely replace m-i-t
5. Eventually get rid of all tools and use only 'kmod' tool (a la git, systemctl, udevadm and others)

Status

- ~ 14.5 KLOC (libkmod and tools - insmod, rmmod, modprobe, depmod, modinfo and lsmod)



Status

- Close to release v6 (waiting some pending bugs and repository on kernel.org)
- Udev, systemd and other initrd tools already depend on libkmod
- Architectures supported: **x86**, **x86-64**, **ARM**, PPC, PPC64, SH4, MIPS, SPARCv9, SPARC64, HPPA, S390
- libc: known to work with glibc, eglibc, uClibc and dietlibc (with some patches)

Status - Desktop distros

- Major distros adopting kmod

  - Archlinux: replaced m-i-t with kmod 5

  - Fedora (F17): replaced m-i-t with kmod 5

  - Opensuse: replaced m-i-t with kmod 5

  - Debian: package in Experimental

  - Ubuntu: ??

  - Mageia, Openmamba and others reported to be using

Status - embedded

- Angstrom: using libkmod 3
- Buildroot: using libkmod 5
- Poky, Yocto: ~~???~~ (Darren Hart said there are patches pending to add kmod)
- Android: ... more later

Status - embedded

- Why embedded should care about kmod?

  - Allow module loading / hotplug

  - Link init/udev/mdev/your-home-made-solution directly to libkmod:

    >> avoid several fork/exec calls during boot

    >> having configurations and indexes in memory, we can be faster

Module loading on Android

- Very primitive module handling - the equivalent of insmod/rmmod
  - Used by toolbox (adb shell)
  - Used by init (it's a command available for init.rc file)
- Vendors don't allow module loading -> no external devices. See talk at **ABS 2012**: "USB Device Support Workshop", Bernie Thompson - Plugable Technologies

Module loading on Android

- Linking Android's init to libkmod

  - Very few code to add support for module loading with all the necessary goodies for hotplug

Packaging, coding, testing

Packaging

- 2 ways of using kmod
  - Only as a library
  - As a replacement to m-i-t
- `./configure [ --enable-tools ] && make && make install`

Packaging

- Create symlinks (there's only 1 tool, named kmod)

  Typical configuration:

```
/usr/bin
    insmod -> kmod
    kmod
    lsmod -> kmod
    modinfo -> kmod
    rmmod -> kmod
/sbin/
    modprobe -> ../usr/bin/kmod
    depmod -> ../usr/bin/kmod
```

Coding

- How to use libkmod?
- Steps:
    1. Init library: grab context object, setup logging function, pre-load indexes, etc
    2. Create module object by path, name or through index lookup
    3. Operate on that module: insert, insert with dependency handling, remove, get info, etc

Coding - example

Hands on - udev or systemd

Automated testing

Testsuite

- Testsuite added on kmod 5
- Need to address regression reports that were being received from different architectures and different distributions

Testsuite - features

- Each test runs isolated on a separate process
- Trap calls to libc functions, modifying the result:

  - All functions dealing with path: `open()`, `fopen()`, `opendir()`, `stat()`, etc

  - uname()

  - init_module()

  - delete_module()

Testsuite - features

- Goal of function traps: allow each test to have a fake rootfs and don't touch current state of the system
- Test both library API and tools:

  - Inline tests in test definition

  - Exec built binaries: modprobe, insmod, modinfo, depmod, etc

Testsuite - anatomy

Hands on - Anatomy of a test

Thanks

ProFUSION embedded systems

THE LINUX FOUNDATION

Thank you for your attention
Questions?

Repository: git://git.profusion.mobi/kmod.git
Mailing list: linux-modules@vger.kernel.org
IRC: #kmod at freenode