Why Would Anyone Develop on Debian/Ubuntu

Wookey

19th September 2012

Outline

1. A little History

2. Debian Derivatives ecosystem

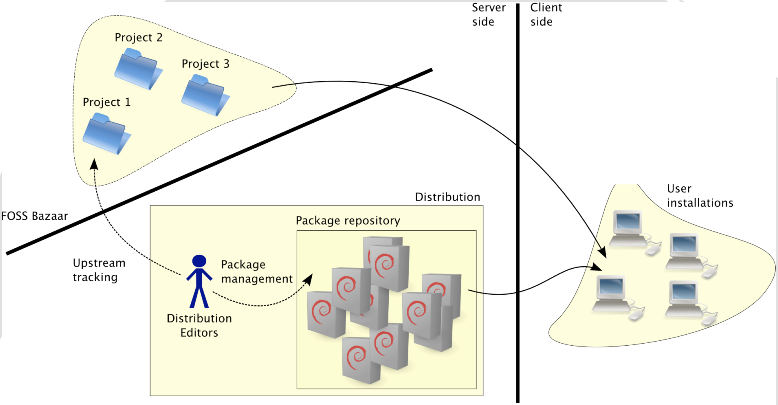
3. Developing the Debian way

4. Some new stuff

Outline

1. A little History
2. Debian Derivatives ecosystem
3. Developing the Debian way
4. Some new stuff

Prelude — the notion of “distribution”



- distributions are meant to ease software management
- key notion: the abstraction of package
- offer coherent collections of software
- killer application: package managers

Debian: once upon a time

> *Fellow Linuxers,*
>
> *This is just to announce the imminent completion of a* **brand-new Linux release**, *which I'm calling the* **Debian Linux Release**. *[...]*
>
> *Ian A Murdock, 16/08/1993*
>
> *`comp.os.linux.development`*

- entirely independent - no one commercial sponsor
- Free software, tools and open process
- built collaboratively by experts

Debian: one of a kind?

1993 — not many distros back then

19 years later — *lots* of other distros

openSUSE, Linux Mint, PCLinuxOS, Slackware, Gentoo Linux, CentOS, FreeBSD, Arch, Sabayon, Puppy, Lubuntu, MEPIS, Ultimate, NetBSD, Tiny Core, Zenwalk, CrunchBang, Dreamlinux, Vector, Kubuntu, Maemo, Red Hat, aptosid, Peppermint, PC-BSD, Chakra, Salix, ClearOS, KNOPPIX, Xubuntu, Super OS, BackTrack, gOS, TinyMe, Zentyal, EasyPeasy, Frugalware, Clonezilla, Pardus, Meego, OpenBSD, Quirky, PC/OS, Zorin, **Debian**, SystemRescue, Element, Unity, SliTaz, Macpup, wattOS, Scientific, Mythbuntu, Slax, DragonFLY, Elive, linux-gamers, 64 Studio, Ubuntu, mageia, Nexenta, Parisx, NuTyX, GhostBSD, Kongoni, moonOS, LFS, Lunar, Imagineos, Untangle, Fedora, Yellow Dog, aLinux, Yoper, IPFire, BlankOn, Mandriva, PureOS, FreeNAS, Moblin, Linpus, TurboLinux, blackPanther, . . .

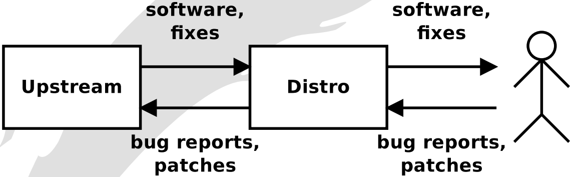
with many differences:

- technical choices
- release management
- release schedule
- target users/purpose
- community
- support
- packaging system
- user base
- look & feel
- package set

How is Debian different?

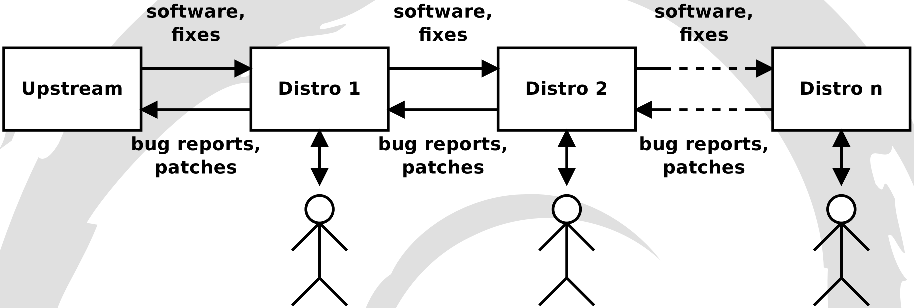
The distribution pipeline

yesterday . . .



The *new* distribution pipeline

. . . today



Upstream

software, fixes

bug reports, patches

Distro 1

software, fixes

bug reports, patches

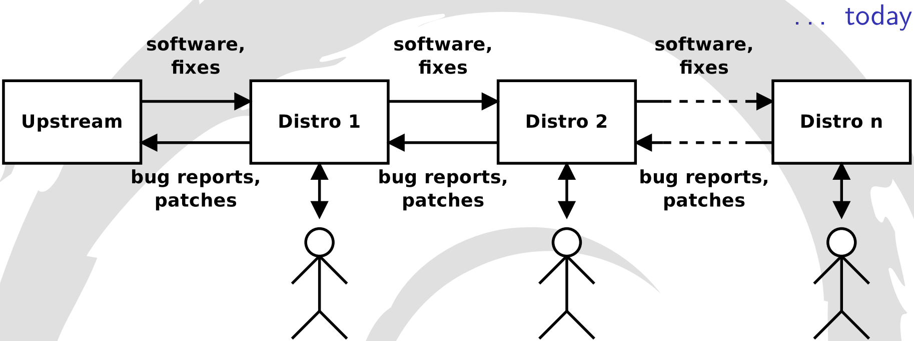
Distro 2

software, fixes

bug reports, patches

Distro n

The *new* distribution pipeline



This is a Good Thing

- wider FLOSS adoption
- more eyeballs swallow more bugs
- more potential contributors

Outline

1. A little History

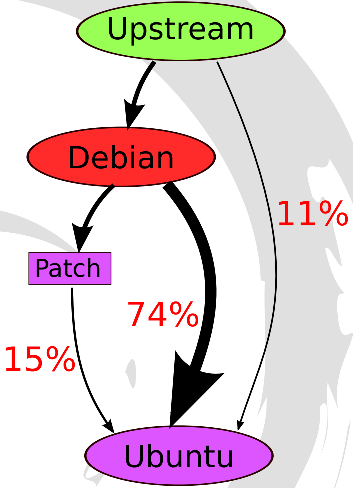
2. Debian Derivatives ecosystem

3. Developing the Debian way

4. Some new stuff

A Debian derivative example: Ubuntu

- started in 2004 by Canonical
  target: desktop
- Debian derivative
- very popular (15–20x Debian?)
- historical/past correlations
  - heavily customized/forked in main
  - very close to Debian elsewhere
- sprouting its own derivatives (≈80)
  - . . . as Debian *transitive* derivatives



Data for Oneiric Ocelot, main + universe

Another derivative example: Emdebian (grip)

- Only reprocessing tools - no upstream delta
- Identical binaries
- Reduced on-disc bloat
- Smaller package set
- Reduced metadata

Blends and remixes

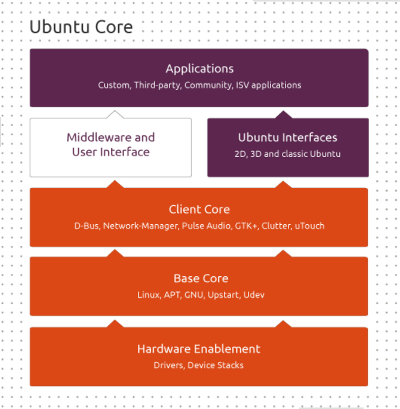
Debian Pure Blends

- Debian Edu
- Debian Med
- Debian Science
- Debian GIS
- DebiChem
- Freedombox
- Debian Automotive

`blends.alioth.debian.org`
debian-embedded@lists.debian.org

Ubuntu remixes

- Ubuntu Automotive remix



Ubuntu Core

Applications
Custom, Third-party, Community, ISV applications

Middleware and User Interface

Ubuntu Interfaces
2D, 3D and classic Ubuntu

Client Core
D-Bus, Network-Manager, Pulse Audio, GTK+, Clutter, uTouch

Base Core
Linux, APT, GNU, Upstart, Udev

Hardware Enablement
Drivers, Device Stacks

Outline

1. A little History

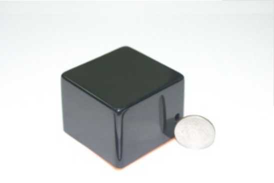
2. Debian Derivatives ecosystem

3. Developing the Debian way

4. Some new stuff

The Universal OS

- Wide architecture support (arm, i386, amd64, mips, powerpc . . .)
- Same environment on dev box and target



  - Easy debugging
  - Everything available on target

Everything is a package

- Reproducible build environment
- Installable and removable components
- Reliable OTA component updates
- Automatic release tracking
- Guaranteed corresponding source

- Most things are already packaged
  - media players
  - navigation tools
  - virtualisation
  - bluez, conman, ofono
  - GENIVI stack
  - ...

Open and Legal

- Open infrastructure - no secret sauce
- Huge array of tools

  rebuildd, reprepro, dpkg-dev, debrsign, dupload, sbuild, pbuilder, debdiff, debchange, debconf, svn-buildpackage, git-buildpackage, debcheckout, debcommit, debi, dose, dpkg-source, debsnap, apt-cache, grep-dctrl, devscripts, . . .
- Legal checks already done
  - All software meets the 4 freedoms
  - Linking checked for incompatibility
  - Nothing you can't distribute

Derivative Mechanisms

- Choose your subset
- dpkg-vendor - build-time variation
- Package and patch tracking systems
- Launchpad cross-distro references
- Rebuildable components

Focus on the delta

- Spend effort on the differentiation
- Working upstream saves time
- Mixed and prioritised repositories
- Automotive needs really long-term security support
- Fast boot choices/optimisations

Smaller delta→less work

Debian or Ubuntu

Fundamentally quite similar, but some differences

Debian

- 2 yr release cycle
- 3-3.5yr security support
- Do-ocracy
- More solid upgrades
- Distributed support
- sysvinit, systemd or upstart

Ubuntu

- 6 month release cycle
- 18-months security support - 5yrs for LTS releases
- Quicker to get things changed
- Commercial support
- Upstart only

Differences in binary drivers/firmware policy

Effective community working

- Propose ideas early
- Take and give feedback
- Write down specs for shared understanding
- Learn to work in the open
- Upstreaming saves a lot of work in the long run

Managing derivatives - Everyone has the same issues

- Tracking upstream
- Managing patches
- Running repositories, build daemons
- VCS integration, making releases

Share that work with standard tools

Outline

1. A little History
2. Debian Derivatives ecosystem
3. Developing the Debian way
4. Some new stuff

Multiarch

- Put libraries into architecture-specific paths
  - /usr/lib/libfoo (amd64)→/usr/lib/x86_64-linux-gnu/libfoo
  - /usr/lib/libfoo (armel)→/usr/lib/arm-linux-gnueabi/libfoo
  - /usr/lib/libfoo (i386)→/usr/lib/i386-linux-gnu/libfoo
- Install libraries side-by side i386/amd64, arm/arm64, amd64/arm64
- Run foreign binaries in-place with qemu
- 32/64 special casing goes away (/lib64, /emul/ia32-linux)
- Canonical locations
  - Files don't move when cross-bulding
- Build/host version lockstep

Example:

```
dpkg --add-architecture i386
apt-get install skype
```

Crossbuilding - library paths

Runtime is the same as build-time.

Old system (classic/dpkg-cross)

build-time library path:	/usr/arm-linux-gnueabi/lib/libfoo
runtime library path:	/usr/lib/libfoo

Multiarch

build-time library path:	/usr/lib/arm-linux-gnueabi/libfoo
runtime library path:	/usr/lib/arm-linux-gnueabi/libfoo

Much harder for libtool to screw it up

Example:

```
apt-get -aarmel build-dep acl
dpkg-buildpackage -aarmel
```

Bootstrapping

- dpkg build profiles
- Remove circular build-dependencies
- Cross-build from scratch
- Dose3 dependency analysis
- Ports possible without Yocto/OE

Thanks

Wookey
wookey@wookware.org
http://wookware.org