Building GNOME

What is this talk about?

- Meta-build systems (jhbuild, rpm/dpkg, Yocto)
- Improving new GNOME developer experience
- Improving long-time GNOME developer experience
- Changing what we're building and how (with an eye to improving the GNOME user experience)
- If we have time, GNOME OS thoughts

What is this talk NOT about?

- Module-internal build systems (autotools vs cmake vs ...)
  - (my opinion: autotools is mostly good)
- Dpkg vs rpm vs...
  - (my opinion: they are all pretty much just the same uninteresting wrappers for tar/wget; an intern could write a basic one in a week)
- How to build third party applications for GNOME/Linux
  - This is an entirely separate and complex topic

Why are we here in this talk?

(context: lots of refactoring of the Linux build)

“So there's a very significant 2.1% reduction of work for the compiler to do, very nice!” -- Ingo Molnar, linux-kernel mailing list, 2011-05-28

Ingo is a very talented developer, paid (probably well) to do what he does. Why does he care about build times?

GNOME

Competent build benefits

- Edit, Compile, Debug
- Continuous integration
- Fast, reliable builds means more testing

Edit, Compile, Debug

- The length of this cycle directly impacts developer productivity
- It is worth taking shortcuts to speed things up
- Developer time spent on meta-tasks like builds can pay off over time
- There is a big difference between < 10 seconds and >= 10 seconds. At > 10 seconds, people will lose attention and switch to reading their email

Continuous Integration, part 1

- A basic prerequisite of competent development
  - Knowing who broke the tests and when allows us to revert changes for a constantly usable tree
- Constantly producing means there is no “magic dance” for a release
  - Changing anything at the last minute implies the possibility of breaking changes, even if the change is just incrementing a version in configure.ac
- I will either bring this to GNOME, or die trying

Continuous Integration, part 2

- http://tinderbox.mozilla.org/showbuilds.cgi?tree=Mo
  - Performance tracking over time
- All of this can combine to deliver higher quality software to users, and help new developers
- I will either bring this to GNOME, or die trying

Integration Testing

- Difference between unit and integration testing
  - Unit tests have low external dependencies and usually test internal API
- Most of our current “make check” are really integration
  - Almost all the glib tests should be split out so we can run them without building glib

jhbuild

- In GNOME, jhbuild is our meta-build tool
- Designed for contribution
  - Git checkouts
- Designed for speed (sorta)
  - A lot of shortcut commands offered like jhbuild buildone, won't complain
- Designed to do partial builds
  - Work from a presumably stable base
- Does not (currently) help distribute binaries

What about “distros”?

- (Debian/Fedora style, not Gentoo)
- Self-hosting
- Reproducible/Reliable
  - Be able to get a security update out
- Compliance with licenses (GPL)
- Encourage FOSS
- Also, about as slow as you can get without involving virtualization
  - I/O is expensive

GNOME

Yocto

- Designed for cross builds
- Designed for people making Linux-based products – deep control over OS image
- Optimized for building an entire OS image from source on one machine
- Fast (parallel, built in ccache usage, etc.)
- Handles GPL compliance
- Has ~full time Intel people on it
- Is the future replacement for jhbuild

GNOME

Continuous Integration, part 2

- http://tinderbox.mozilla.org/showbuilds.cgi?tree=Mo
  - Performance tracking over time
- All of this can combine to deliver higher quality software to users, and help new developers
- I will either bring this to GNOME, or die trying

Integration Testing

- Difference between unit and integration testing
  - Unit tests have low external dependencies and usually test internal API
- Most of our current “make check” are really integration
  - Almost all the glib tests should be split out so we can run them without building glib

Improving jhbuild: Build failure

- It's ok for builds to fail, if it's somewhat obvious
  - Example: module A uses API just added in module B
- “Undebuggable” build failures can cost hours or even days
  - Stale files left over in build tree
  - API that changes semantics
- Jhbuild will delete no-longer-installed files
- Jhbuild will do “git clean -dfx” before building

Improving jhbuild: Misc

- We will use parallel make by default
- srcdir != builddir; improve reliability and also support both gcc -O2 and gcc -O0 builds
- Add logging
  - Track compiler warnings
  - pastebin
- Better terminal interaction

GNOME

Improving jhbuild: New Developers

- Fresh VM
- jhbuild sysdeps
  - Reuse system packages
- jhbuild sysdeps –install
  - Find what we need, install it on system

GNOME

Continuously produced binaries implies...

- ...shipping them.
- Many users want to be able to see/try the latest without compiling things
- Binaries could be used as a quickstart for new GNOME developers

GNOME OS thoughts, part 1

- GNOME is the right place for development
- GNOME currently holds 1/2 of the application API (glib, gtk+, themes)
- GNOME should get more credit for things
  - Plugging in USB devices
- Should not 100% compete with distributions
  - Need to add value where they lack focus
  - Refer people who want distributions to them
- Should sync up with distributions explicitly
  - How do desktop features impact server etc.

GNOME

GNOME OS thoughts, part 2

Controversial/hard issues:

- Security updates
- Support for proprietary software
  - What is stable application API
- $#@!$ packages

  State of the art is patching Firefox to say “I'm broken now”
- Non-application stuff
  - Codecs, proprietary drivers, antivirus

GNOME OS thoughts, part 3

- Corporate influence
  - Full time developers and system administrators massively influence project direction
  - Ideally GNOME is not dependent on one financial stream
    - Apache requires projects to have at least two independent income sources
- Branding is obviously tough...
  - But we want e.g. RHEL to credit GNOME in addition to Fedora

GNOME OS thoughts, part 4

- Mozilla is a successful project with a useful impact on society and the technology industry, that happens to also be FOSS
  - They have a revenue model which scales per user; GNOME does not
- Still though, anything we can do to be more like Mozilla, we should.

GNOME

Phases

- 3.2: less lame jhbuild
- 3.2-3.4: /opt/gnome binaries
- 3.2-3.4: Integration testing over /opt/gnome on build.gnome.org
- 3.4: Build sheriff
- 3.4-3.6: Yocto+jhbuild generated base image and SDK for use in VM
- 3.6: Installer and updater (OS and apps)

Questions?