

# KDE & Science

## A Talk of Two Halves

Marcus D. Hanwell  
cryos@gentoo.org

Gentoo Linux

Gentoo UK 2007  
14 July, 2007



What I am going to talk about today,

- ▶ KDE today and the road to KDE 4
  - ▶ The herd
  - ▶ The applications and libraries
  - ▶ Issues faced
  - ▶ Interaction with others
  - ▶ The road to KDE 4
- ▶ The scientific applications in Gentoo
  - ▶ The herd in the beginning
  - ▶ Splitting the category and the herd
  - ▶ The applications and libraries
  - ▶ Issues faced—specialised applications
  - ▶ Interaction with upstream



## The KDE Herd

- ▶ The KDE herd takes care of KDE applications and libraries
- ▶ Gentoo and the KDE herd is run by volunteers
- ▶ Small team managing a large number of ebuilds
- ▶ Gentoo is desktop environment agnostic—can be tough striking the right balance
- ▶ Always on the look out for volunteers



## Core Applications & Libraries

- ▶ The core applications are kept in kde-base
- ▶ Comprise all applications and libraries part of the main release
- ▶ Available in monolithic or split ebuilds
- ▶ All are built from the main KDE release tarballs
- ▶ Maintain minimal patch sets and aim to get patches included upstream where possible
- ▶ Stay as close to upstream sources as possible



## Extra KDE Applications

- ▶ Lots of other very important KDE applications

**k3b** killer burning application

**amarok** awesome music application

**digikam** amazing photo editing/management

**kile** for those  $\text{\LaTeX}$  junkies out there

- ▶ And lots more. . .



## Extra KDE Applications continued...

- ▶ Present some unique challenges to effectively package
- ▶ Go into the main FHS hierarchy whereas KDE is outside it
- ▶ Can be difficult to get linking right at times
- ▶ Binary compatibility in KDE helps a lot
- ▶ Is this the optimal way to handle things?



# Splitting Up KDE

- ▶ A few years ago split ebuilds were introduced to Gentoo
- ▶ Individual KDE core applications such as konsole could be compiled
- ▶ Improved granularity for dependencies—konsole instead of kdebase!
- ▶ So far we have maintained old monolithic and new split
- ▶ Many other distros only have the equivalent to split
- ▶ Should we continue supporting both?



# DBus & HAL—Conflict Between Gnome & KDE

- ▶ Dbus and HAL are used by Gnome and KDE
  - ▶ The API has not always been stable
  - ▶ This has led to big issues maintaining Gnome & KDE
  - ▶ Led to delayed stabilisation and forced patching
- ▶ As more is shared between DEs these types of problem are likely to become more of a issue
- ▶ Can be avoided by only using stable APIs and/or coordination
- ▶ Led to some interesting interaction between our herds!



# Upstream KDE Developers

- ▶ How do we currently interact?
  - ▶ IRC
  - ▶ Mailing lists
  - ▶ Bugzilla
  - ▶ Blogs
  - ▶ Conferences
- ▶ Do we need to improve communication?
- ▶ What is the optimal way to request urgent patches or ask questions?



## Other Linux Distributions

- ▶ Could we share patches more effectively?
- ▶ Shouldn't patches be pushed upstream anyway?
- ▶ Could we better coordinate security and other bug fixes?
- ▶ Does LSB makes any sense for a source distribution?
- ▶ Could we be more open?



# Distribution Applied Patches/Customisations

- ▶ Traditionally Gentoo has always had a philosophy of using minimal patch sets
- ▶ We have struggled with the branding question
- ▶ More and more packages are getting branding
- ▶ KDE is following this trend
  - ▶ Gentoo is about choice
  - ▶ Branding is controlled by a USE flag
- ▶ How much should we patch and customise KDE?



# How Can We Improve?

- ▶ Going forward how could we improve interactions?
- ▶ The open source world is filled with passionate people
  - ▶ Don't let it get personal
  - ▶ Disagreements should try to remain technical
  - ▶ Most of us are volunteers—time is in short supply
- ▶ Interaction can be tough to get right
  - ▶ Don't be an island



## Getting To Grips With CMake

- ▶ We have a lot of experience with autotools
- ▶ Docs for CMake aren't always what they could be
- ▶ Assignment of lib dir isn't a built-in anymore
- ▶ CMake ships a lot of built in libs which is bad for security
- ▶ Hopefully all these things will improve as CMake matures
- ▶ Coloured output and progress indications are nice
- ▶ Syntax seems nicer albeit less well documented



# Automagic Is Bad

If there is one thing you remember about my talk:

- ▶ Automagic detection of optional deps is bad!
  - ▶ Deterministic build systems are essential
  - ▶ Enable and disable flags should be provided
  - ▶ Automagic detection and inclusion breaks the dependency tree
  - ▶ This affects Gentoo quite badly but all binary distros have similar issues with automagic detection—unpredictable results
- ▶ Still getting to know CMake, autotools had `-with-*` and `-without-*` or `-enable-*` `-disable-*`
- ▶ One of the most difficult problems for us to fix with builds—it makes Larry grumpy!



## Splits & Monolithic

We find ourselves asking the question whether we should maintain both the split ebuilds and the old style monolithic ebuilds. As far as we can tell both are popular. Traditionally we kept everything the same as upstream—one source package = one ebuild.

- ▶ Should we keep monolithic ebuilds
- ▶ Are the split ebuilds still possible?
  - ▶ Initial work has taken place in overlays
  - ▶ New build system
  - ▶ Lots of changes
- ▶ Very much a new code base and new build system



## Split The Source Too?

- ▶ Split ebuilds effectively build a custom source tree at unpack
- ▶ Building split ebuilds from monolithic source
  - ▶ Multiple monolithic source tarballs required
  - ▶ To install kuickshow all of kdegraphics is needed
  - ▶ Quite wasteful of resources
  - ▶ Binary distros split their packages
- ▶ Could be reduced by improving portage—ranged source
- ▶ Haven't yet conducted tests to see if this approach would work as expected



# The Scientific Herd

- ▶ The scientific herd takes care of most scientific applications and libraries
- ▶ Quite a large area with many applications and upstreams
- ▶ Extremely varied upstreams with a large array of licences
- ▶ It would appear that build systems are not a scientist's forte
- ▶ Specialised applications from chemistry, biology, physics, mathematics and electronics



## Splitting the Category

- ▶ As the scientific herd grew the number of applications and libraries grew very quickly
- ▶ Originally all kept in the app-sci category
- ▶ The category was split into many sci-\* categories



# Splitting the Herd

- ▶ Quite some time after the category split
- ▶ Some developers were put off by the number of applications maintained by the sci herd
- ▶ The decision was taken to split the herd
- ▶ Improved granularity of bugzilla mail
- ▶ Made the split sci herds more approachable
- ▶ Most people only interested in one or two areas
  - ▶ Biology
  - ▶ Chemistry
  - ▶ Electronics
  - ▶ Mathematics
  - ▶ Physics
  - ▶ Geosciences



# Applications

- ▶ Many applications and libraries concerned with different fields
- ▶ Written in a large array of languages encompassing libraries, command line apps and GUIs
- ▶ Many applications require specialist knowledge to test even basic functionality
- ▶ Requires a diverse range of developers, herd testers and users in order to effectively package
- ▶ There is quite a lot of cross over with other herds
- ▶ Stabilisation can be a difficult question



# Applications

**abaKus** is a great calculator package

**marble** is a new 3d globe that doesn't need OpenGL

**qucs** is a circuit design tool that is very easy to use

**ePiX** Graphs for  $\text{\LaTeX}$  loving C++ graphers

**R** Statistical language with a large array of its own packages

**grace** Publication quality graphs with ugly GUI

- ▶ Many more already packaged as well as lots waiting
- ▶ Maintenance burden can be very high for sci packages



## Scientific Overlay

- ▶ Introduced before many overlays existed
- ▶ Intended as a testing ground for new packages
- ▶ Also a place for packages that didn't necessarily have maintainer resources
- ▶ Gives herd testers a place to experiment with packages
- ▶ Should access be given more openly?
- ▶ How else can we increase the speed of package addition?
- ▶ Some packages simply will not make it in
  - ▶ Limited developer resources
  - ▶ Difficult upstream
  - ▶ Unreliable build system or code
- ▶ Gives easy access to these packages



## Issues—Specialised Applications

- ▶ Require developers with specialised skills in scientific fields
- ▶ How best to facilitate package testing by arch testers?
- ▶ Can be hard to judge what applications are really needed
- ▶ Some applications are not very portable
- ▶ As mentioned build systems are not the forte in this area
- ▶ Some applications need very unusual hardware



# Herd Testers

- ▶ Lower the barrier to entry
- ▶ Introduce more people with specialised knowledge
- ▶ Good option for power users who do not have the time to become a developer (or feel they don't)
- ▶ Herd testers can go on to become developers if they wish
- ▶ Increases the number of people examining bugs and helping with ebuids
- ▶ Could be better managed and advertised



# Interaction With Upstream

- ▶ Very varied upstreams from school students working in their spare time to large research groups and collaborations
- ▶ Some upstreams do not make proper releases
- ▶ Some are very tied to a particular distribution
- ▶ Some don't even like us packaging their work
- ▶ Some think hand coded Makefiles are preferable to autotools!
- ▶ Can be tough interacting with such varied upstreams
- ▶ Many of these problems are quite generic issues



## Conclusions

- ▶ I was recruited by Olivier (ribosome) into the sci herd
- ▶ Have done work mainly in the sci, KDE and amd64 herds
- ▶ There are many similarities between the work in the sci and KDE herds
  - ▶ Interaction with upstream
  - ▶ Varied build systems
  - ▶ Complex libraries and applications
- ▶ There are also quite a few differences
  - ▶ Sci herd deals with lots of very different upstreams
  - ▶ Large array of different languages used
  - ▶ Scientists more pragmatic a lot of time
  - ▶ Some extremely specialised applications
- ▶ Both are fun—I am now working on a scientific KDE application for the summer thanks to Google!



# The End

- ▶ I would like to thank you all for listening
- ▶ This presentation was created in  $\text{\LaTeX}$  using the beamer class
- ▶ Gentoo is all about choice, we put a lot of work into offering as much as we can to all of our users. . .
- ▶ I welcome your input
- ▶ Questions?

