Introduction to Gentoo Linux and the Portage Package Manager

Guilherme Amadio

Agenda

- Introduction to Gentoo

- Portage package manager

  - Main features and concepts

  - Ebuild scripts

- Gentoo Prefix project

- Gentoo and HEP software

- Conclusions

Introduction and brief history of Gentoo

▸ Gentoo is the name of a penguin species
▸ Founded by Daniel Robbins as Enoch Linux (12/1999)
▸ Gentoo Linux 1.0 was released on 31/03/2002
▸ Gentoo Foundation created in 2004, holds all copyright
▸ Council of 7 elected members for technical oversight and policy
▸ Source-based distribution with wide support (x86, ARM, Mac OS X, etc)
▸ Website: www.gentoo.org

Gentoo penguin (source: Wikipedia)

Portage's main features

▸ Written in Python, based on FreeBSD's ports system

▸ Packages are special shell scripts called ebuilds

▸ Extensive options for dependency management

▸ Highly flexible configuration/customization

▸ Parallel and distributed builds (with distcc)

▸ Multiple installed versions of the same package

▸ Easy to support live packaging from git/svn/hg repos

Portage's advantages

- Portage is a mature solution (15+ years development)
  - Formal, versioned Package Manager Specification (PMS)
  - Used by Google's Chrome OS and CoreOS
- More than 19,000 currently available packages, including many HEP packages (ROOT, Geant4, etc)
- Leverage work done by other volunteer developers
- Extensive documentation at devmanual.gentoo.org
- Support for many different hardware architectures

Portage's disadvantages

▸ Portage is not designed to install many different independent package trees, although it is possible

▸ Performance for dependency calculation is not great

▸ Other operating systems (e.g., Mac OS X, etc) are not as well supported as Linux

▸ It can be cumbersome to support compilers other than GCC, ICC, and LLVM/Clang.

Important Gentoo and Portage Concepts

▸ Package manager specification (EAPI)

▸ Package tree (holds ebuild scripts with build recipes)

▸ Package tree overlay (tree with add-on packages)

▸ *SLOTS* and *USE flags*

▸ Virtual packages

▸ Package Keywords

▸ Arch Profiles

Distribution Models for HEP

▸ Full OS, Virtual Machines, Containers

- Base images with common HEP packages
- Binary package servers with pre-compiled add-ons
- Automated image build process with Catalyst

▸ Gentoo Prefix Environments

- Packages installed within a prefix by non-root users
- Good solution for CVMFS or HPC environments
- Support for Mac OS X and other systems (users' laptops)

Ebuild scripts

- ▸ Shell script with several phase functions
    - For example: src_prepare(), src_configure(), src_install(), etc
- ▸ Short detour: [Quickstart Ebuild Guide](#)
- ▸ Dependencies express how to rebuild if needed
- ▸ Easy to create package for newer versions after first
- ▸ emerge/ebuild command line tools used to install (similar to yum/rpm, but for source-based packages)

Gentoo Prefix Project

- Uses Portage to install packages within a prefix

- Uses host OS's kernel and C library

- Optionally, can share only the kernel on Linux

- Support for Linux, Solaris, Mac OS X, and other UNIX systems

- Downside: best if used with single tree, as many separate trees take space

Gentoo and HEP software

▸ Gentoo is popular among scientists

- Several developers are physicists

▸ Some HEP software already available:

- ROOT, CLHEP, Geant3/4, Pythia, HepMC, herwig(++), cernlib, PAW, lhapdf, looptools, yoda, etc
- Other interesting tools available:

  AFS, CVMFS, Nvidia CUDA, ICC, Vc, etc

▸ New software packages easy to add

Conclusions

- ▶ Gentoo's Portage is a mature solution for packaging
  - ● Many available packages, including HEP software
  - ● Possibility to automate image builds
- ▶ Not intended as only solution
  - ● Not ideal for highly combinatorial installations, although it is still possible to use it for this case
- ▶ ChromeOS / CoreOS model may make sense for HEP
- ▶ Example: automated builds into CVMFS and base VM and container images + binary package hosts for users

Live Demo and Q&A

Thank you!