Hands-on cross compile Gentoo from scratch for RISC-V

樊其轩
Alex Fan
alexfanqi@yahoo.com

Thanks PLCT/ISCAS

- Offer great learning opportunity of the emerging RISC-V platform
- Sponsor motivated interns and excellent mentors

Objective of this presentation

- Introduction to Gentoo and its package/source management system
- Hands-on guide to build from scratch (ie. bootstrap) a Gentoo stage3 image for RISC-V

Watch also: Linux Distro on RISC-V – status update by 傅炜 Redhat

Bootstrap

- “Pull yourself up by your bootstrap”

- RISC-V has different combinations of instruction sets and ABIs

  - rv64g(imafd)c, -rv32i

  - lp64d, lp64

  - need to build on one platform targeting the other

Introduction to Gentoo

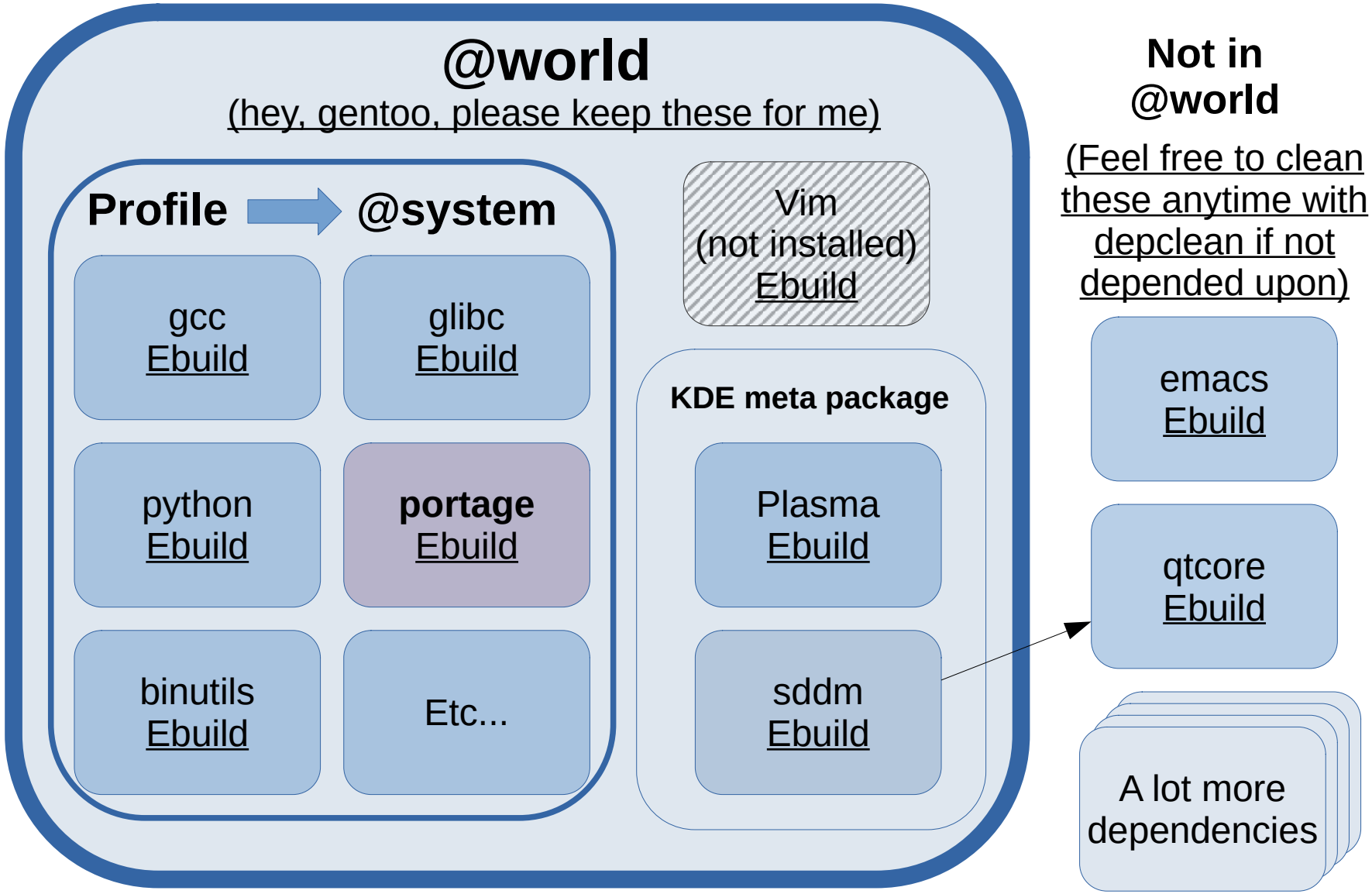
- Packages are in general built **directly from source**
- Source is fetched and compiled in temporary directory (optionally generate binary package)
- Packages then get installed & **merged** into the system root, usually recorded in @world as well
- The whole process is automated and managed per package essentially by plain script called **ebuild files**.

Introduction to Gentoo

USE Flags

make.conf

Package.use



Introduction to Gentoo

- Ebuild system, EAPI, portage

  check out /var/db/repos/gentoo/

- Profiles & Use Flags to control global & per-package features and components

  make.conf & package.use

- RISC-V: two level lib64/lp64d vs one level profile?

- *@system, @world*

  /var/lib/portage/world

- Overlays, user repos

Install a package on Gentoo

> emerge -av vim

-a --ask :
confirm yes/no before installation

-v --verbose:
show additional information

Exercise 1:

- Now that you have learned how to install packages with portage, it is time to build the whole system on your own!

Overview of bootstrap steps

1. Generate cross compile toolchains: Crossdev
2. Build the seed tar ball
   - Build from sources with binary package option enabled
   - Install essential binary packages into a new SYSROOT location
3. Setup qemu-user and binfmt
4. Use Catalyst to generate further stages & final image

1. Cross build toolchain: crossdev

- Obtain pre-built toolchain

  https://toolchains.bootlin.com/
- Or build it on your own?
- Why not let Gentoo handle things for you?

  - Create a custom repo for sanity and isolation

  > *crossdev -t riscv64-unknown-linux-gnu*

  - setup CHOST, CBUILD, SYSROOT, flags etc

  - automate building cross compiler, headers and portage

  - create quick symbolic links

2. Build the seed tar ball

- As minimal as possible, contains necessary tools for building later stages.
- Choose the profile
- Compile sources with binary package option enabled in one root directory

  > riscv64-unknown-linux-gnu-emerge -1auDNb --keep-going @system

- Install binary packages into another new root directory

  > env SYSROOT=$new_root ROOT=$new_root riscv64-unknown-linux-gnu-emerge -1aek --keep-going @system

2. Build the seed tar ball

- Why two root directories? File collision errors
- What if failed to build certain packages?

  - choose another version or disable certain flags in make.conf or package.use

  - change to a even smaller profile, ie. embedded

  - chroot into the root directory with qemu-user & binfmt

  - modify ebuild, not likely necessary

3. Setup qemu-user and binfmt

- Need a virtual environment to run RISC-V binaries on amd64

  - full system emulation? qemu system, virtualbox, etc

  - user emulation. binfmt with chroot or systemd-nspawn

  1). install qemu on host system with correct use flag

  2). register binfmt, /etc/binfmt.d/

  3). copy/install qemu-static to $new_root

  4). chroot/systemd-nspawn into $new_root to check

  https://wiki.gentoo.org/wiki/Embedded_Handbook/General/Compiling_with_qemu_user_chroot

4. Use Catalyst to generate final image

- Catalyst: Gentoo's release building tool. Configurable with spec files and catalystrc

  > catalyst -f stage1.spec && catalyst -f stage2.spec && catalyst -f stage3.spec

- Feed in our seed tar ball, and generate stage3 tar ball after 3 passes. Will chroot into the tar ball for each pass.
- Need to copy qemu-static into tar ball for each pass.

  - manually copy into working directory

  - or use the catalyst spec option (undocumented):

    interpreter: opt/qemu-riscv64

  https://wiki.gentoo.org/wiki/Catalyst#Specs_files

Further way beyond

- Speed up the compilation time.

  - tmpfs, distcc, ccache, icecream
- Test 1 level library profile
- Additional resources

  https://riscv.org/wp-content/uploads/2015/02/riscv-software-stack-tutorial-hpca2015.pdf

  https://wiki.gentoo.org/wiki/Porting

  Linux Distro on RISC-V – status update by 傅炜 Redhat

Thanks for your time!

Try it out yourself and have fun!

Questions?