The meta-virtualization layer of OpenEmbedded

Bruce Ashfield
Principal Technologist
Linux Products Group
Agenda

- Introduction
- Brief OpenEmbedded introduction / history
- How Wind River uses OE
- meta-virtualization
- OE + meta-virtualization + security
- Future / Questions
A Brief Introduction …
Enterprise vs Embedded

- The world is not limited to enterprise vs embedded
  - It’s really more a continuum, from the pre-defined to the fully customized

- Many users have requirements that are between those of the Enterprise Linux and Embedded Linux
  - Some enterprise like systems are source based
  - Some embedded like systems are based on preconfigured binaries

- One size does not fit all in the Linux ecosystem
OpenEmbedded

- Includes a cross-compile build environment
- User is required to configure and define their environment before compiling
- Created a custom binary Linux distribution based on configuration
- Output includes ‘packages’, like an enterprise OS, filesystem images, SDKs

- All software is downloaded from the original provider as source code
- Designed to be expanded/extended
- Commercial and Community support
The Yocto Project

[ yoc-to ]

The smallest unit of measure, equal to one septillionth (10⁻²⁴).
What is the Yocto Project?

- The Yocto Project is an Open Source project with a strong community.
- It is based on a collection of embedded projects, tooling, and procedures:
  - OpenEmbedded
  - Application Development
  - Quality Assurance testing
  - Commercial Ecosystem
- The Yocto Project is designed to provided an ecosystem to the Operating System developer.

It’s not an embedded Linux distribution – it helps you creates the custom one for you.
Who is the Yocto Project

- Founded under the Linux Foundation
- Members include numerous companies and projects spanning Silicon vendors, Board vendors, OSVs, ISVs, and end users
- Lead by Advisory Board and Technical Leadership
- Advisory board is responsible for ecosystem, marketing, etc.
- Technical Leadership is a meritocracy based group that leads various projects and makes technical contributions
Why was the Yocto Project started?

- The industry needed a common build system and core technology
  - Bitbake and OpenEmbedded build system

- A place for Commercial Interests to work together to avoid duplicating effort
  - Why should each company have a competing build system?
  - Why is each organization integrating the same components in different ways?
  - Why are we all duplicating effort, duplicating bugs, and duplicating solutions?

- Less time spent on things which don’t add value
- More time spent on things which do add value
Comparisons ...
Embedded Linux Requirements

- **Goal:**
  - Build upon the existing Linux ecosystem and goals
  - Build a complete, customized, Linux system for a specific device
  - Include Bootloaders, Linux Kernel, Root Filesystems

- **Build from scratch from source**
  - Reproducibility, IP compliance reasons, customization

- **Use cross-compilation to build software**
  - Often developer/build machine will be faster or more plentiful than target hardware

- **Need a vibrant community**
  - Documentation, support, training
Alternatives / Options

- OE / Yocto project
- Enterprise Linux
  - IoT variants
- Buildroot
- Roll-your-own
Enterprise Linux

- Easy entry level
- Often used for prototyping
- Customization or Support – not both
- Maintenance
- IoT focused systems
  - Project Atomic
  - Ubuntu Core
- Not cross-compiled
- Not source code based
Buildroot

- Allows simple entry into Embedded Linux
- Limited built in extension points
- No binary packages
- Most users end up having to create their own forks
Roll-your-own

- Enterprise based
- Silicon Vendor/Board Vendor SDK
- Completely custom
With the alternatives, why OpenEmbedded?

- OE may not be right for all situations!
- PC like usage model? enterprise Linux or variants
- One-time use board bring up? OE might be too complex
- Foot-print, long-term maintenance, commercial ecosystem, IP, etc concerns?
  - OE/Yocto Project is probably what you want
Layers and the Ecosystem...
Layers

- Layers are a way to manage extensions, and customizations to the system
  - Layers can extend, add, replace or modify recipes
  - Layers can add or replace bbclass files
  - Layers can add or modify configuration settings
  - Layers are added via BBLAYERS variable in build/conf/bblayers.conf

- Best Practice: Layers should be grouped by functionality
  - Distribution configurations
  - BSP/Machine
  - Functional groups
  - Project/Product specific components
Layers

LEGO is a trademark of the LEGO Group
Layers

- Developer-Specific Layer
- Commercial Layer (from OSV)
- UI-Specific Layer
- Hardware-Specific BSP
- Yocto-Specific Layer Metadata (meta-yocto)
- OpenEmbedded Core Metadata (oe-core)
Ecosystem

- The ecosystem is formed by the collection of layers and projects
  - Broad and active
  - Content of layers is a work in progress
- Various levels of maintenance and ‘quality’
  - Yocto project compatibility and layer index
  - Tools to support layers and recipe creation/maintenance
- If you have layers: test for compatibility and publish!
Build System Workflow
How (and why) is Wind River using the Yocto Project?
What about commercial Linux?

- Prior to the Yocto Project, there were many commercial Linux products
  - Each was incompatible with the others, even if they shared a common core
- In many ways each commercial vendor had created their own ‘Roll-your-own’ system and tried to share the costs among their customers
- This lead to many limited ecosystems:
  - Limited ISV support
  - Limited semiconductor support
  - Limited BSP support
- Vendor lock-in was a problem for customers
Innovation / Differentiation

- Linux is now a commodity operating system
- Nobody is going to buy a new kernel
- People will pay for service, new development, features, etc.
Software Lifecycle Management

- Open source software lifecycle is short
  - New versions are released constantly, but not on any fixed cycle
  - Days, weeks, months or years
  - Old versions are often abandoned as soon as new versions are released

- Commercial product lifecycles vary
  - Developed in 6 months, only sold for 6 months
  - Developed for 6 months and sold for years...
  - Developed over years and sold for years...

- It is Wind River’s job to help the customer manage the commercial lifecycle vs the fast changing open source lifecycles
Software Integration

- Roll-your-own or the Yocto Project?
- It’s easy to do something once...
- It’s not to bad to do it twice...
- But supporting something for a long time takes process, planning and expertise
- Carrying costs, including maintenance, updates, etc add up quickly!
- Continuous Integration of the Yocto Project
What that looks like ...

- **Core product**
  - Closely based on OE core
  - Selected / curated layers
  - github / community editions
  - CI/CD stream

- **WR BSPs**

- **Vertical specific ‘products’ (distros)**
  - WR core + additional layers and configuration
  - Networking, industrial
  - Technology horizontals: virtualization /containers, security ..
meta-virtualization
meta-virtualization overview

- From openhub:
  - has had 773 commits made by 104 contributors
  - with a very well-commented source code
  - has a well established, mature codebase
  - maintained by a very large development team
  - starting with its first commit in June, 2012

- Current maintainer(s): Bruce Ashfield (Wind River)

- Contributors: OSVs (Wind River, Mentor, Monta Vista, Enea, ...), distros, individual users
meta-virtualization goals

- **Goals:**
  - Single point of integration for virtualization technologies
    - VMs and containers
    - Core technology + support software
    - Many audiences: Bleeding edge and established tech
    - Well tested and stable
    - Baseline for creating OE derived virtualization solutions
  - Recipes migrate over time
meta-virtualization components

- Technology
  - virtualization: guests/hosts, containers, management, utilities / support, configuration(s): images, kernel

- ~98 recipes (some are variants)
  - recipes-containers: Kubernetes, runc, docker/moby, OCI, LXC, containerd
  - recipes-core: system init, runv
  - recipes-devtools: support recipes for core/containers
  - recipes-extended: libvirt, hyperstart, kvmtool, image definitions, dev86 ..
  - recipes-kernel: configuration fragments to support VMs/Container features
  - recipes-networking: CNI, OpenVSwitch, netns
meta-virtualization use cases

- **Virtualization: Xen / KVM**
  - Small, secure, etc

- **Containers: docker, LXC, runc, moby**
  - Lightweight, micro-services, serverless, etc
  - Standards based: OCI

- **VMs and container co-existence**
  - Single image, nested, runv ...

- **Management and control**
  - CLI: libvirt
  - Orchestration: kubernetes, CNI, etc
How Wind River uses meta-virtualization

- Core hypervisor support
  - realtime + security variants

- Containers
  - Core container support: docker, lxc, runc
  - Container OS: OverC
Secure boot + meta-virt @ Wind River ...
Secure boot requirements

- As little as possible is BSP specific
  - leverage hardware when possible
  - Avoid one-offs
- Multiple layers of security
- key management
- Multi-architecture
Wind River Linux Security

Security technology

- Security policy
  - SE Linux MLS/MCS
  - Login and remote access
- Access controls
- Memory protection
- PKI
- Secure & measured boot
- Linux IMA
- TPM 1.2 and TrouSerS
- TPM 2.0 and TPM2-TSS

Virtualization technology

- Least privilege/privilege controls
- Access controls
- Resource utilization protection
- Memory protection

UEFI, TXT

Virtualization technology

- Remote attestation
- File system integrity monitoring
- Backup/restore
OverC / WRL Security Reference Image

- Wind River Linux secure kernel
- Secure Boot
- Linux IMA
- Secure backup/restore
- SELinux
- TPM 2.0 and TPM2-TSS
- IPsec/L2TP/PPTP VPN
- Storage encryption
- Secure update
- OpenSCAP configuration guidance
- AES-NI, QAT
- PTT
- UEFI, TXT
What’s next ?…
What the future may hold …

- New technology in meta-virtualization
  - Hypervisors (ACRN ...)
  - container / sandbox techniques (gvisor? pouch? kata containers)
- Improved system level use cases / tests, not just buckets of packages
  - security 'toolkit' / core components
  - See Richard Purdie's 2.6 planning email
- Update mechanisms (OTA or not), reference binary feeds
- Developer experience
- More ... we need help!