Tools and workflows for multi kernel version juggling of short term fixes, long term support, board enablement and features with the upstream kernel

Bruce Ashfield - Xilinx

About me

- Development background
  - Kernel drivers through tools to container runtimes
- Maintained product/distro kernels since 2005
  - Wind River, Yocto Project, Xilinx

Problem: Support the following

- Multiple (but not all) “active” kernel versions
  - 3+ supported streams (varying lengths)
- All major architectures
- Many different boards
- Development / Extension capabilities
- Thousands of changes versus mainline
  - Semi features, out of tree functionality, in house development, etc
- Flexible for fundamentally different features
  - -rt, footprint, Carrier Grade, Industrial, container, virtualization, etc
- Small team

Solution: Goals

- Changes are visible
  - tracked over time
  - Patches / features are carried forward continually (and released twice a year)
- Encourage open development and mainlining of changes
- Common feature set
- Common configuration
- Different feature enablements
- Developers and end users are equally supported
- Standard tools / workflows are fully supported
  - Write as few custom tools as possible
- Predictable release cadence

Observations

- Common goals, but very few identical tools/workflows/maintenance models
  - Plumbing is the exception (i.e. git)
- A lot of little known tools, frameworks
- Everyone is doing very similar work and duplicated effort
  - But yet still hard to collaborate/unify/”standardize”
- Supporting the developer, the distro build and the end user is challenging
- Complexity creeps in very easily
- Even a small amount of overhead turns some users away
- Timing LTS kernels, release dates and customers is ‘interesting’

“Solutions”

- The Yocto kernel management solution
  - config fragments, patch tracking repository, generated tree(s)
  - Branched repository with integrated changes (no patching at build time)
  - custom change management tools / workflows …
- Hierarchy of development trees / contribution points
  - Reduce the number of trees (we can’t agree on one, but maybe a few ?)
  - Project kernel trees -> mainline
- Packaging of kernel source (with and without history)
- SDK / eSDK for developers
- Cross build and native build use the identical tools
- Integrate with common / standard CI/CD stacks

Thoughts

- Starting is hard
  - Huge problem space
- Changing workflows is hard
  - Getting developers to change tools is nearly impossible
- Inertia of status quo is a challenge
  - “It’s not pretty, but it works”
- ….