Embedded Linux Conference

arm

Will it Boot?

Standards for embedded Linux

Grant Likely, Arm

#lfelc @glikely

THE LINUX FOUNDATION

Exactly what problem are you trying to solve?

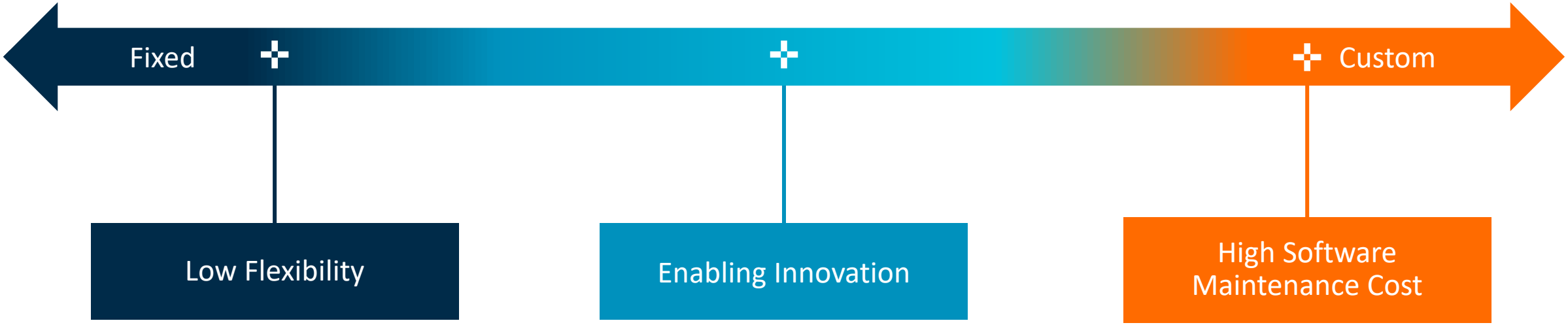
arm SystemReady

- Typically, embedded Linux is vertically integrated
  - Firmware, OS, and apps bound together in a single build
  - Every platform has slightly different behaviour
- Scale Problems
  - Distro must be customized to boot (change Devicetree, vendor kernel tree, special configuration)
  - Platform owner must maintain everything – problem for security updates
  - Distros cannot handle per-platform customization
- Ecosystem Problems
  - Extra engineering cost to get suitable OS booted – problem for ODM/OEM market
  - Embedded using different tools from other parts of Linux ecosystem

Standards are designed to reduce engineering effort, while increasing the flexibility of products

Tension between standardization and flexibility

arm SystemReady



arm

Major changes to embedded Linux aren't realistic

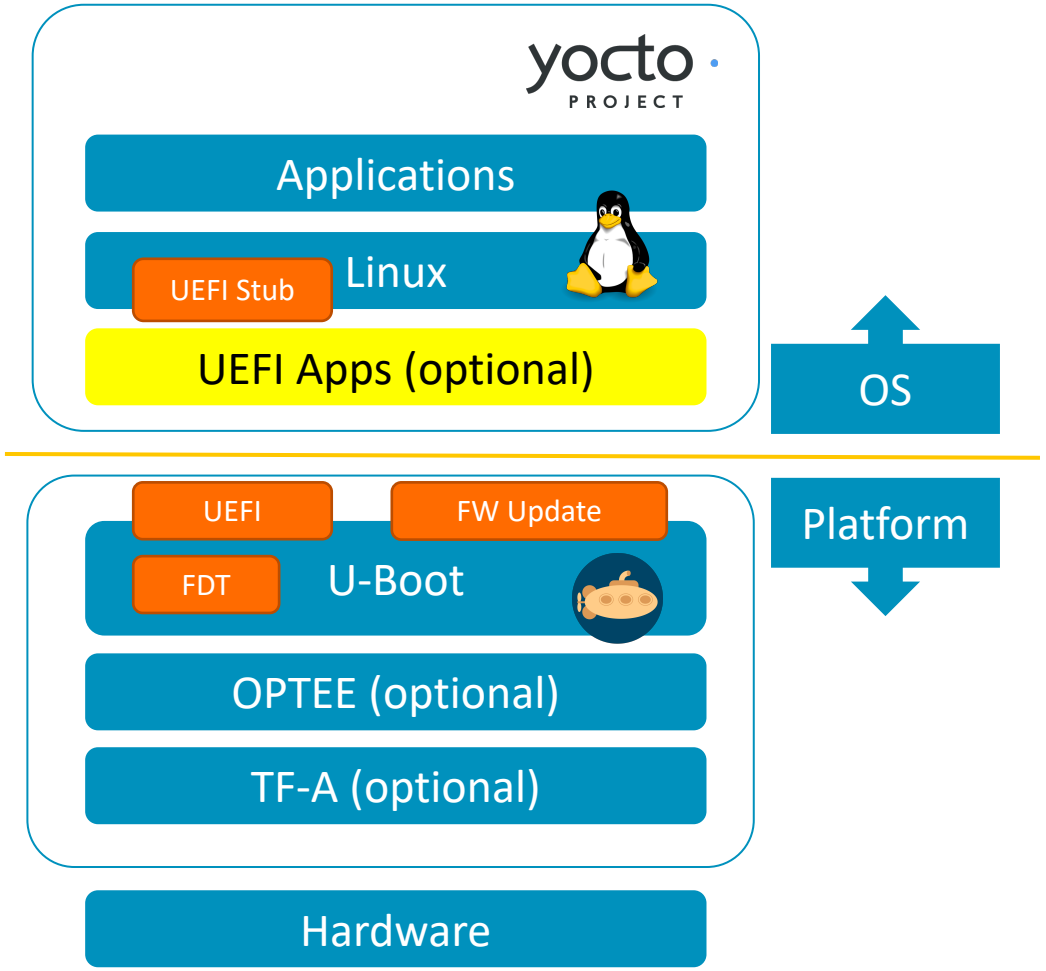
arm SystemReady

- Embedded Linux has well established, mature technology and tools
  - Projects: U-Boot, TF-A, OPTEE
  - Meta-distros: Yocto, OpenWRT
  - Devicetree
- Rearchitecting from ground up is not an option
  - Need migration path
  - Cannot be disruptive
  - Minimal engineering effort
- To be relevant, standards must work with existing tools and techniques

Distros need consistency

arm SystemReady

- Embedded Linux doesn’t mean custom OS
  - Distros can provide better support options
- Custom board enablement is not supported
  - Generic image must boot
  - Platform drivers must not cause conflicts
  - Ideally use same technologies as non-embedded platforms
- Mainline first policy
- Need standards for:
  - Boot order and control
  - Pre-boot execution environment (e.g., for Shim, Grub, systemd-boot, etc)
  - Platform description
  - Firmware Update
  - Verified boot


yocto PROJECT
Applications
Linux
UEFI Stub
UEFI Apps (optional)
OS
UEFI
FW Update
FDT
U-Boot
Platform
OPTEE (optional)
TF-A (optional)
Hardware

UEFI is specification, not implementation

arm SystemReady

- Multiple implementations
- Tianocore EDK2 is the reference open source implementation
  - https://github.com/tianocore/edk2
- U-Boot also implements UEFI
  - CONFIG_EFI_LOADER

Very little in UEFI is required

arm SystemReady

- The subset of what OSes require is nicely contained
- EBBR project defines a UEFI subset suitable for embedded Linux
  - Required features to support Linux Distros
  - Implemented in U-Boot
  - Works with embedded hardware
  - Uses Devicetree
- Problematic runtime features are optional
  - SetVariable()
    - not easy to implement if OS owns the storage device
  - SetTime()
    - OS owns the RTC device
  - UpdateCapsule()
    - Again, OS owns flash device

UEFI defines the ABI

arm SystemReady

OS uses firmware services during early boot, before device drivers get loaded

BootServices

Callable before EBS()

- Boot Services Table
  - Events and Notifications
  - Memory Management
  - Image Loading
  - Watchdog Timer
  - ExitBootServices()
- Protocols (ABI extensions)
  - Network
  - Console
  - Storage
  - Graphics

RuntimeServices

Callable before and after EBS()

- Runtime Services Table
  - Variables
  - Timers, Wakeup and RTC
  - System Reset
  - Firmware update

SystemTable

- Pointers to data about the system
- System Descriptions
  - Devicetree
  - SMBIOS
  - ACPI
- UEFI details
  - ESRT
  - RuntimeServicesSupported

UEFI defines the behaviour

arm SystemReady

- Boot device selection
- BootOrder and BootXXXX
- Removable device booting
- Handoff to OS

UEFI defines *limited* functions at runtime

arm SystemReady

- Set/Get RTC settings
- Set/Get Variables
- UpdateCapsule()

UEFI Secure Boot

arm SystemReady

- Authentication scheme for delegating authority
- Hierarchical key database
  - Platform Key (PK) – Root of trust for UEFI
  - Key Exchange Key (KEK) – signing authorities
  - DB & DBX – key & hash databases, allow and revoke respectively

Firmware Update

arm SystemReady

- U-Boot implements UpdateCapsule() ABI and ESRT
- Device Firmware Update (DFU) backend
- UEFI Capsule as wrapper around FIT image
  - Capsules can be managed by fwupd & LVFS
  - tools/mkimage -f capsule.its capsule.itb
  - tools/mkeficapsule capsule.itb capsule.bin
- CapsuleApp.efi for applying capsules
  - Use either self-built SCT, or Arm SystemReady ACS image
  - FS1:\> boot/efi/apps/capsuleapp.efi /path/to/capsule.bin

Devicetree tied to Linux kernel doesn’t scale

arm SystemReady

- Historically considered tied to the Linux kernel build
- Doesn’t work for the distros
- Isn’t great for Yocto either
- Firmware needs to provide copy of Devicetree by default
  - Enables OS portability

Mainline Linux

arm SystemReady

- Distros have a mainline-first policy
  - If board support isn’t in mainline, the distros will not support
- Mainline support helps Yocto too
  - Less reliance on custom vendor trees
  - Uprev kernel or apply security patches
- Requiring mainline solves many support problems

Testing

arm SystemReady

- UEFI Self Certification Test (SCT)
  - UEFI ABI test utility
  - https://github.com/tianocore/edk2-test
  - https://gitlab.arm.com/systemready/edk2-test-manifest
- U-Boot UEFI Self Tests
  - CONFIG_CMD_EFI_SELFTEST=y
  - => bootefi selftest
- FWTS
- Arm's SystemReady ACS-IR
  - Prebuild disk images – work on SD and USB
  - https://github.com/arm-software/arm-systemready
  - Includes: SCT, Arm BSA & BBR tests, FWTS

Formal Compliance Testing

arm SystemReady

- Arm launched SystemReady program in October 2020
- First certifications in June 2021
- Requirements
  - Conform to EBBR
  - Pass SystemReady ACS-IR test suite
  - Boot 2 unmodified Linux distros
  - Support UpdateCapsule()
- Using 3rd party testing labs
- More information
  - https://developer.arm.com/systemready
  - Contact email: systemready@arm.com

Work to be done

arm SystemReady

- Boot Device Selection (BDS) improvements
  - BDS currently implemented as hush scripts
  - Boot list or menu
- Console device selection
  - console= kernel argument still needed too frequently
  - Enabling framebuffer should not break console
- Measured Boot
- Devicetree validation and stability

Certification program is running now

See www.arm.com/systemready-certification-program for list of certified platforms and expect more announcements this year

Join us on the EBBR community project at https://github.com/arm-software/ebbr

Contact us about getting your platform certified systemready@arm.com