Yocto Project® : linux-yocto reference kernel maintenance and kernel workflows

Bruce Ashfield, Xilinx

Yocto Project *Virtual* Summit, May 2021

Agenda

- Why ?
- What ?
- When / Where ? (“A week in the life of ….”)
- Samples:
  - Custom -stable against linux yocto
  - Configuration option analysis and debugging

Why?

linux-yocto: purpose

- Multi-architecture, maintained reference
  - Userspace / kernel stacks
  - Identical to mainline linux in many branches
- Support tightly coupled kernel packages
  - libc-headers, perf, lttng, systemtap, etc
- Drive workflow changes / improvements
- Cadence / Inertia
  - Predictable / shared kernel versions
- Collaboration / Leverage collective testing
- Support BSP and assist upstreaming
- Kernel configuration meta-data and tooling
  - More important than you think!

linux-yocto: what it is not!

- Replace, fork or otherwise supercede the mainline kernel
- “must use”
  - Extended tasks are separate from core kernel support
- Intended to support all BSPs and use cases
  - Vendor kernels, OSVs
  - Use some, none or all
- Production complete
  - It is production ‘ready’

What?

kern-tools

- Developed over 10 years of maintaining distro kernels
  - Yes, almost everything has been tried
  - No, we can't equally maintain it with directories full of patches
    - User should never see a patch fail to apply for supported features
    - You spend your life fixing fuzz and patch failures
- Scales to maintain ~6 kernel versions, 6 architectures, 14 boards (in oe-core/meta-yocto-bsp alone), 3 kernel types
  - 30+ invasive / non-invasive features
  - Hundreds of patches at any given time
    - ... may or may not conflict, used in many combinations

kern-tools (continued)

- Uses for the same tools:
  - Day to day management
  - Configuration audit
  - OE core build support
    - kernel-yocto “style”
    - standard / core kernel “style”
    - tgz and non-git patches
  - External (non-OE) build support
  - Full tree reconstruction and configuration maintenance
- Repositories:
  - Branched and versioned kernel source git (linux-yocto, -dev)
  - Kernel configuration and meta data (yocto-kernel-cache)

kern-tools: in the build

- Preparation: do_kernel_metadata()
  - Inputs (SRC_URI, KERNEL_FEATURES, etc)
    - .scc, .cfg, .patch, kernel meta-data
    - tools: spp, scc
  - Outputs:
    - bsp entry point, config.queue, patch.queue, merge.queue
- Application: do_patch(), do_kernel_configme()
  - Patches: kgit-s2q (wraps git am)
  - Merges: git (note: rarely used)
  - Configuration: merge_config.sh
- Audit: do_kernel_configcheck()
  - symbol_why.py

Versioning: linux-yocto-dev

- -dev tracks (almost) all korg versions
  - In progress (active): standard/*
  - Archived: <version>/standard/*
  - May rebase (standard/*)
  - Note: may fold into linux-yocto

Versioning: linux-yocto

- Single repository tracks all released / maintained versions
- Branching:
  - master: unmodified tracking branch for tip of linus' tree
  - v<version>/base: unmodified upstream release. tracks -stable
  - v<version>/standard/base: small, OE specific changes
    - i.e. perf build fixes, boot quirks, etc
  - v<version>/standard/<BSP or topic name>: larger, more invasive changes

linux-yocto: recipes

- oe-core master has the active release recipes
  - dev, LTS and a ‘latest’ kernel (when LTS != latest)
  - ‘rt’ and ‘tiny’ variants for the same version (although -rt is not always possible)
- Building against any of the ‘unreleased’ versions is simple
  - See reference section for v5.7 example

When / Where ? (a week in the life of ...)

maintenance

- Daily / As required:
  - Sources: linux-yocto mailing list, pull requests, lkml, IRC, etc
  - Patch/merges: Bug fixes (CVE or ‘standard’), BSP merges, feature merges, configuration changes
  - Testing: local + AB
- Weekly (sometimes multiple times per week):
  - Current release: -stable merges
  - Testing: local + AB
- Bi-weekly:
  - Previous (non-EOL) release(s): -stable merges
  - Testing: local

maintenance

- Monthly:
  - Older release: -stable merges (build test only)
- Bi-Monthly:
  - linux-yocto-dev generation and bring up
  - Feature merges, configuration audit, etc
  - Basic / Core testing
- ~Quarterly:
  - Generation of “named/versioned” linux-yocto recipes for release
    - Based on current linux-yocto-dev at the time
    - Full configuration audit and support matrix testing
- Periodic:
  - Bug fixes for “unsupported” kernels (if not part of monthly)

Example: bugfix / CVE (simplified)

```
% cd linux-yocto.git
# checkout the base branch (depends on the type of the fix)
% git checkout v5.10/standard/base
% git am -s CVE.mbox

# merge the fix to all supported branches in the hierarchy
% kgit-merge -v -v --auto

# Stores the patch in the tracking repository
% kgit-commit-to-cache -o ../kernel-cache/patches/misc/ HEAD~..HEAD

# generate the SRCREV update to linux-yocto recipes
% kgit-branch-status --incremental HEAD~..HEAD -s "linux-yocto/5.10: CVE xyz" -o
~/poky/meta/recipes-kernel/linux :yocto-5.10

#  local testing, AB as required

# push changes to linux-yocto repository
% git push -a

# release fix via pull request to oe-core for merge
```

Example: -stable merge (simplified)

```
% cd linux-yocto.git; git checkout v5.10/standard/base
# checks tag, increments, fetches tag and merges to current branch
% kgit-get-tag-and-merge ../linux-stable

# merge the fix to all supported branches in the hierarchy
# potentially spend hours fixing merge conflicts, etc
% kgit-merge -v -v --auto

# Update the kernel version in tracking repository
% kgit-get-tag-and-merge ../kernel-cache

# generate the SRCREV update to linux-yocto recipes
% kgit-branch-status -v  --release -o ~/poky/meta/recipes-kernel/linux :yocto-5.10
% kgit-branch-status -v  --release -o ~/poky/meta-yocto-bsp/recipes-kernel/linux :yocto-5.10

# local testing, AB as required

# push changes to linux-yocto repository
% git push -a; git push --tags

# release -stable via pull request to oe-core for merge
```

Workflow: new development or release kernel

- Update meta-data (kernel-cache)
  - Create meta-data repository branch
  - Update kernel branching information (ktypes/base)
- Update target kernel repository (linux-yocto, linux-yocto-dev)
  - tools: spp/scc, kgit-scc, kgit-s2q
    - Creates patch queues for all supported BSPs
  - Applies queues to tree (creates branches, applies patches, etc)
    - Automatically carries forward tracked patches and features
      - Includes major features like -rt, aufs, yaffs2, etc.
  - Fix conflicts, refresh patches, drop obsolete patches, import new features, etc

Workflow: new development or release kernel

- Update kernel configuration data for new kernel
  - Used in the audit phase
  - Categorizations for hardware/non-hardware updated
  - Drop obsolete or invalid configs
- Synchronize linux-libc-headers
- Create “versioned” recipes
- Build test + Configuration audit cleanup
- Boot / Feature testing
  - Includes updating tightly coupled tools/recipes
- Push branches to linux-yocto
- Profit! (aka Release)
  - Update preferred versions, reference BSPs, etc

Sample ‘clean’ stable with linux-yocto

linux-yocto: ‘other’ versions

- The following slides show how a v5.7-stable can be built from linux-yocto
- Even though it was never released as part of a project release

Create a 5.7-stable recipe using linux-yocto

```
SUMMARY = "Linux kernel"
SECTION = "kernel"
LICENSE = "GPLv2"

inherit kernel
inherit kernel-yocto

LIC_FILES_CHKSUM = "file://COPYING;md5=6bc538ed5bd9a7fc9398086aedcd7e46"

KCONFIG_MODE="alldefconfig"
KBUILD_DEFCONFIG_qemuarm="multi_v7_defconfig"
KBUILD_DEFCONFIG_qemux86-64="x86_64_defconfig"
# optional: only needed if we are reusing a reference qemu machine
KERNEL_DANGLING_FEATURES_WARN_ONLY="t"

# just the code
SRC_URI = "git://git.yoctoproject.org/linux-yocto.git;name=machine;branch=v5.7/base"

# Example: add a fragment:
# SRC_URI += "file://virtio.cfg"
# Example: all the kernel config meta-data, use a branch 'close' to the v5.7 release
# SRC_URI += "git://git.yoctoproject.org/yocto-kernel-cache;type=kmeta;name=meta;branch=yocto-5.8;destsuffix=kernel-meta"

# required for newer kernels
DEPENDS += "${@bb.utils.contains('ARCH', 'x86', 'elfutils-native', '', d)}"
DEPENDS += "openssl-native util-linux-native"
DEPENDS += "gmp-native"
```

Create a 5.7-stable recipe using linux-yocto

```
LINUX_VERSION ?= "5.7.19"
PV = "${LINUX_VERSION}+git${SRCPV}"

# matches upstream v5.7-stable release
SRCREV_machine ?= "6b9830fecd4a87d7ebb4d93484fef00f46d0fa0f"
# optional: only relevant if yocto-kernel-cache is used
SRCREV_meta ?= "7b9ba93dfc39efa90056eed0b572e86909127aaa"

# some basic compatibility
COMPATIBLE_MACHINE = "(qemux86|qemux86-64|qemuarm)"
```

k.org using kernel-yocto classes

```
inherit kernel
require recipes-kernel/linux/linux-yocto.inc

# korg
SRCREV="d07f6ca923ea0927a1024dfccafc5b53b61cfecc"
SRC_URI = "git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git;protocol=git;nocheckout=1"
LINUX_VERSION ?= "5.13-rc2"
LINUX_VERSION_EXTENSION = "-yoctized-${LINUX_KERNEL_TYPE}"
PV = "${LINUX_VERSION}"

KBUILD_DEFCONFIG = "x86_64_defconfig"
# or: SRC_URI += "file://defconfig"

KERNEL_DANGLING_FEATURES_WARN_ONLY = "1"
COMPATIBLE_MACHINE = "(qemuarm|qemux86|qemuppc|qemumips|qemux86-64)"
LIC_FILES_CHKSUM = "file://COPYING;md5=6bc538ed5bd9a7fc9398086aedcd7e46"

DEPENDS += "${@bb.utils.contains('ARCH', 'x86', 'elfutils-native', '', d)}"
DEPENDS += "openssl-native util-linux-native"
```

Thanks for your time

Extra/Reference: Kernel option debugging

What is kernel config analysis ?

- Commonly asked question:
  - “The kernel configuration phase ignores option ‘x’”
- The ‘truth’:
  - option_ignore=$(($RANDOM%2))
- The reality:
  - The option is likely missing a dependency, is not reachable or is disabled by another fragment

How to determine why ‘that option was ignored’

- Previously
  - Manual inspection of merge_config and audit logs
  - Other tricks ..
- Analysis tools were completely rewritten for gatesgarth
  - More information is available
  - More flexibility to ignore options that are not a problem
  - Full .config or option specific analysis available

When are kernel options checked ?

- Automatically for linux-yocto builds via kernel_configcheck()
- On demand via config_analysis task

kernel_configcheck (1/3)

- Runs after each linux-yocto configuration
- Has several levels / tweaks available:
  - KCONF_AUDIT_LEVEL (default: 1)
  - KCONF_BSP_AUDIT_LEVEL (default: 1 or 2)
  - KCONF_AUDIT_WERROR (default: off)

kernel_configcheck (2/3)

- KCONF_AUDIT_LEVEL > 0
  - option mismatches are reported
    - Options that are in the .config, but don't match the last value set in a fragment
    - Only "hardware" options if audit level == 1

kernel_configcheck (3/3)

- KCONF_BSP_AUDIT_LEVEL > 0
  - Reports fragments with errors or warnings
- KCONFIG_BSP_AUDIT_LEVEL > 1
  - Only reports for the ARCH being built
- KCONFIG_BSP_AUDIT_LEVEL > 2
  - Reports if any fragments redefine options (can be verbose)

Kernel .config analysis

```
% bitbake -f -c config_analysis linux-yocto
WARNING: linux-yocto-5.8.16+gitAUTOINC+7b9ba93dfc_4e5f265cee-r0
do_config_analysis: Configuration analysis executed,
see: <snip>/config-analysis.txt for details
WARNING: linux-yocto-5.8.16+gitAUTOINC+7b9ba93dfc_4e5f265cee-r0 do_config_analysis:
Configuration audit executed,
see: <snip>/config-audit.txt for details
```

- config-analysis.txt
  - Option by option 'config blame'
- config-audit.txt:
  - Debug and additional information about options (errors, references, etc)

config-analysis.txt

```
[NOTE]: symbol blame for .config

  CONFIG_CC_VERSION_TEXT : "arm-poky-linux-gnueabi-gcc (GCC) 10.2.0" ## .config: 4 :
  CONFIG_CC_IS_GCC : y ## .config: 5 :
  CONFIG_GCC_VERSION : 100200 ## .config: 6 :

<snip>

  CONFIG_LOCALVERSION : "-yocto-standard" ## .config: 19 :configs/v5.8/ktypes/base/base.cfg ("")
    [NOTE]: 'CONFIG_LOCALVERSION' last val ("") and .config val ("-yocto-standard") do not match
    [INFO]: raw config text:

        config LOCALVERSION
                string "Local version - append to kernel release"
                help
                  Append an extra string to the end of your kernel version.
                  This will show up when you type uname, for example.
                  The string you set here will be appended after the contents of
                  any files with a filename matching localversion* in your
                  object and source tree, in that order.  Your total string can
                  be a maximum of 64 characters.

    [INFO]: config 'CONFIG_LOCALVERSION' was set, but it wasn't assignable, check (parent)
dependencies
```

config-analysis.txt (continued)

```
CONFIG_CC_OPTIMIZE_FOR_SIZE : y ## .config: 170 :configs/v5.8/standard/ktypes/standard/standard.cfg (n)
                                                 configs/v5.8/standard/arch/arm/arm.cfg (y)


<snip>

    [NOTE]: 'CONFIG_IP_NF_TARGET_MASQUERADE' last val (y) and .config val (m) do not match
    [INFO]: raw config text:

        config IP_NF_TARGET_MASQUERADE
                tristate "MASQUERADE target support"
                select NETFILTER_XT_TARGET_MASQUERADE
                depends on IP_NF_NAT && IP_NF_IPTABLES && INET && NETFILTER && NETFILTER && NET
                help
                  This is a backwards-compat option for the user's convenience
                  (e.g. when running oldconfig). It selects NETFILTER_XT_TARGET_MASQUERADE.

        Config 'IP_NF_TARGET_MASQUERADE' has the following Direct dependencies (IP_NF_TARGET_MASQUERADE=m):
                IP_NF_NAT(=m) && IP_NF_IPTABLES(=m) && INET(=y) && NETFILTER(=y) && NET(=y)
        Parent dependencies are:
             NETFILTER [y] NET [y] INET [y] IP_NF_IPTABLES [m] IP_NF_NAT [m]
```

config-audit.txt

```
[INFO]: redefined configuration option report:

    - option CONFIG_MODULES is defined more than once
       - configs/v5.8/ktypes/base/base.cfg (y)
       - configs/v5.8/standard/features/systemtap/systemtap.cfg (y)
    - option CONFIG_MODULE_UNLOAD is defined more than once
       - configs/v5.8/ktypes/base/base.cfg (y)
       - configs/v5.8/standard/features/systemtap/systemtap.cfg (y)

<snip>

[NOTE]: no specific option defined, full config analysis follows
[NOTE]: user selected configs:

   - 'MODULES' is a user set value, which resolved to: 2 (y)
       - defined at init/Kconfig:2009
   - 'CC_VERSION_TEXT' is a user set value, which resolved to: x86_64-poky-linux-gcc (GCC) 10.2.0 ()
       - defined at init/Kconfig:11
   - 'CC_IS_GCC' is a user set value, which resolved to: 2 (n)
<snip>
   - 'NET' is a user set value, which resolved to: 2 (y)
       - defined at net/Kconfig:6
       - selected values:
             - NLATTR
             - GENERIC_NET_UTILS
             - BPF
```

config-audit.txt (continued)

```
 - 'CPU_FREQ' is a user set value, which resolved to: 2 (y)
      - defined at drivers/cpufreq/Kconfig:4
      - selected values:
            - SRCU
      - 'CPU_FREQ' is selected

<snip>


[INFO]: the following symbols were not found in the active configuration:
     - CONFIG_DRM_I915_PRELIMINARY_HW_SUPPORT
```

Specific option checking

- List options to check in CONFIG_ANALYSIS variable
- Limits analysis to just those options
- In a .bbappend or local.conf
  - `CONFIG_ANALYSIS_pn-linux-yocto = 'NF_CONNTRACK LOCALVERSION'`
- A per-option analysis and audit txt file are produced

Sample option analysis (blame)

```
WARNING: linux-yocto-5.8.16+gitAUTOINC+7b9ba93dfc_4e5f265cee-r0 do_config_analysis: Configuration analysis executed,
See:<snip>/NF_CONNTRACK-config-analysis.txt for details
WARNING: linux-yocto-5.8.16+gitAUTOINC+7b9ba93dfc_4e5f265cee-r0 do_config_analysis:
[NOTE]: symbol blame for NF_CONNTRACK

  CONFIG_NF_CONNTRACK : m ## .config: 1002 :configs/v5.8/standard/./docker.cfg (y)
                                           configs/v5.8/standard/features/netfilter/netfilter.cfg (m)
    [INFO]: raw config text:

        config NF_CONNTRACK
                tristate "Netfilter connection tracking support"
                default m if NETFILTER_ADVANCED = n
                select NF_DEFRAG_IPV4
                select NF_DEFRAG_IPV6 if IPV6 != n
                depends on NET && INET && NETFILTER && NETFILTER && NET
                help
                  Connection tracking keeps a record of what packets have passed
                  through your machine, in order to figure out how they are related
                  into connections.

                  This is required to do Masquerading or other kinds of Network
                  Address Translation.  It can also be used to enhance packet
                  filtering (see `Connection state match support' below).

                  To compile it as a module, choose M here.  If unsure, say N.

        Config 'NF_CONNTRACK' has the following Direct dependencies (NF_CONNTRACK=y):
                NET(=y) && INET(=y) && NETFILTER(=y)
        Parent dependencies are:
             NET [y] INET [y] NETFILTER_ADVANCED [y] IPV6 [y] NETFILTER [y]
```

Sample option audit

```
WARNING: linux-yocto-5.8.16+gitAUTOINC+7b9ba93dfc_4e5f265cee-r0 do_config_analysis: [INFO]: option 'NF_CONNTRACK' summary:
   - raw config:

        config NF_CONNTRACK
                tristate "Netfilter connection tracking support"
                default m if NETFILTER_ADVANCED = n
                select NF_DEFRAG_IPV4
                select NF_DEFRAG_IPV6 if IPV6 != n
                depends on NET && INET && NETFILTER && NETFILTER && NET
                help
                  Connection tracking keeps a record of what packets have passed
                  through your machine, in order to figure out how they are related
                  into connections.

                  This is required to do Masquerading or other kinds of Network
                  Address Translation.  It can also be used to enhance packet
                  filtering (see `Connection state match support' below).

                  To compile it as a module, choose M here.  If unsure, say N.

   - value in .config is: 'm'
   - visibility: y
   - currently assignable values: n, m, y
   - defined at: net/netfilter/Kconfig:58
   - referenced by the following fragments:
            configs/v5.8/standard/./docker.cfg
            configs/v5.8/standard/features/netfilter/netfilter.cfg
   - assignment details: configs/v5.8/standard/./docker.cfg [y] -> configs/v5.8/standard/features/netfilter/netfilter.cfg [m]
```

Sample option audit (continued)

```
[NOTE]: extended configuration analysis for: NF_CONNTRACK
[NOTE]: user selected configs:

   - 'NF_CONNTRACK' is a user set value, which resolved to: 1 (m)
       - defined at net/netfilter/Kconfig:58
   - 'NF_CONNTRACK' is referenced by (79) Kconfigs
       - selected values:
              - NF_DEFRAG_IPV4
              - NF_DEFRAG_IPV6
```

Thanks for your time

yocto PROJECT | THE LINUX FOUNDATION