yocto PROJECT

It's not an embedded Linux distribution –
It creates a custom one for you.

Yocto BSP Summit
BSP format, Kernels, and Tools

Tom Zanussi
Intel Corporation
April 2, 2012

These slides are available now on the Yocto website. Go to Documentation | Yocto Wiki | Projects | BSPs | Yocto BSP Summit Presentation

yocto PROJECT · THE LINUX FOUNDATION

Agenda

- **The Yocto BSP Format**
  - Background and motivation
  - Current Yocto BSP format and contents
- **Kernels**
  - Available kernel options
  - What's been useful for meta-intel
  - Some new changes
- **Yocto BSP and Kernel Tools**
  - What they are and how they work
  - BSP creation and kernel patching/config examples
- **Questions and Discussion**
  - What's missing? What else would people like to see, if anything?

My Background

- **Joined Intel (and Yocto) in Sept 2010**
- **Author and maintainer of several meta-intel BSPs**
- **Co-author of the current 'Yocto BSP Guide'**
- **Author of the new 'Yocto BSP Tools'**
- **Previously worked in the kernel mainly on tracing**
  - Author of kernel/relay.c (relayfs), perf scripting interface and Perl/Python bindings, perf 'live mode', kernel event filters (kernel/trace/trace_events_filter.c)
  - Major contributor to blktrace, LTT, and systemtap
- **Created systemtap, blktrace, sysprof recipes**
- **Other odds and ends related to BSPs and tracing**

Yocto BSP Format

- **What is a Yocto BSP?**
- **Logically speaking, a Yocto BSP is:**
  - A bitbake layer enabling a specific machine or machines
  - Contains hardware-specific components only
  - Exists as a single independent directory of metadata
    - Can be directly tar'ed, distributed, and built
    - Can be independently maintained inside an external repository
  - This is what developers see
- **When packaged and distributed, a BSP is:**
  - Binary part: deployed binaries from the build, for example, a runnable Linux image which can be booted on the hardware
  - Source part: the set of recipes and other metadata that were used to generate the bundled binaries
  - This is what we make available on the 'Yocto BSP Downloads' page

Yocto BSP Format (Why?)

- **Why do we need a BSP format?**
- **The main reason is packaging:**
  - We want to be able to point users to a tarball for BSP X
    - At 'Yocto | Download | Board Support Package (BSP) Downloads'
  - Everything needed to build the BSP is in the tarball
    - Minus of course the distro metadata, build system, toolchain, etc
  - And nothing unrelated to the BSP is in the tarball
    - Like other BSPs
- **Also the standard reasons:**
  - Consistency
  - Familiarity
  - Standardization
  - Collaboration

'Source' Part of a Yocto BSP

- **Machine configuration**
  - Metadata defining architecture-specific tuning and options
  - Bootloader configuration
- **Kernel configuration**
  - Patches against a Linux kernel version
  - Kernel config options
- **Device firmware**
- **Supporting user space**
  - Hardware-specific applications
  - Additional vertical use-specific applications
- **Toolchain and build system not part of BSP**
  - Defined in other layers

Binary part of a Yocto BSP

- **Provided under <meta-bsp>/binary**
- **Complete bootable disk image**
  - Or whatever build artifacts are needed to boot on hardware
- **A README for how to boot the image**
  - Part of the README in the top-level of the BSP directory
- **Functionality may be limited**
  - You may not want the binaries to be redistributed
    - In other words, you're not creating a distro, just a test image
  - For example, a single terminal or time-limited kernel
- **Should allow a developer to see if the board comes up**
  - Typically both a minimal and a graphical image are bundled

A Brief History of the BSP Format

- **Details and examples are available in the 'Yocto BSP Developer's Guide'**
  - Before Yocto the Guide was part of the 'Poky Handbook'
  - The BSP section first appeared in the Poky 'green' release (Poky 3.3, July 2010)
- **The format has evolved over time**
  - Historically the layout has mirrored the structure of Poky
- **Both the current format and Guide have been around since 'bernard' (Poky 5.0/Yocto 1.0, March 2011)**
  - Made to match the current Poky/Yocto layout
  - At that point, there was one meta-intel BSP, meta-emenlow
  - It was moved out of Poky and into a separate meta-intel repo
  - And it was updated to use the new BSP Format
  - Since then all the meta-intel BSPs have followed the format

BSP file system layout (laverne (Poky 4.0), Oct 2010)

```
meta-bsp/
meta-bsp/binary/zImage
meta-bsp/binary/poky-image-minimal.directdisk
meta-bsp/conf/layer.conf
meta-bsp/conf/machine/*.conf
meta-bsp/conf/machine/include/tune-*.inc
meta-bsp/packages/bootloader/bootloader_0.1.bb
meta-bsp/packages/linux/linux-bsp-2.6.50/*.patch
meta-bsp/packages/linux/linux-bsp-2.6.50/defconfig-bsp
meta-bsp/packages/linux/linux-bsp_2.6.50.bb
meta-bsp/packages/modem/modem-driver_0.1.bb
meta-bsp/packages/modem/modem-daemon_0.1.bb
meta-bsp/packages/image-creator/image-creator-native_0.1.bb
```

BSP file system layout (edison (Poky 6.0/Yocto 1.1), Oct 2011)

```
meta-crownbay/COPYING.MIT
meta-crownbay/README
meta-crownbay/README.sources
meta-crownbay/binary
meta-crownbay/conf/layer.conf
meta-crownbay/conf/machine/crownbay.conf
meta-crownbay/conf/machine/crownbay-noemgd.conf
meta-crownbay/recipes-bsp/formfactor/formfactor_0.0.bbappend
meta-crownbay/recipes-bsp/formfactor/formfactor/crownbay/
meta-crownbay/recipes-bsp/formfactor/formfactor/crownbay/machconfig
meta-crownbay/recipes-bsp/formfactor/formfactor/crownbay-noemgd/
meta-crownbay/recipes-bsp/formfactor/formfactor/crownbay-noemgd/machconfig
meta-crownbay/recipes-graphics/xorg-xserver/xserver-xf86-config_0.1.bbappend
meta-crownbay/recipes-graphics/xorg-xserver/xserver-xf86-config/
meta-crownbay/recipes-graphics/xorg-xserver/xserver-xf86-config/crownbay/
meta-crownbay/recipes-graphics/xorg-xserver/xserver-xf86-config/crownbay/xorg.conf
meta-crownbay/recipes-graphics/xorg-xserver/xserver-xf86-config/crownbay-noemgd/
meta-crownbay/recipes-graphics/xorg-xserver/xserver-xf86-config/crownbay-
  noemgd/xorg.conf
meta-crownbay/recipes-kernel/linux/linux-yocto-rt_3.0.bbappend
meta-crownbay/recipes-kernel/linux/linux-yocto_2.6.37.bbappend
meta-crownbay/recipes-kernel/linux/linux-yocto_3.0.bbappend
```

BSP Components

- **License file(s):**
  - meta-<bsp_name>/<bsp_license_file>
  - Examples: COPYING.MIT, COPYING (GPLv2)
- **README file**
  - meta-<bsp_name>/README
  - How to build and boot the BSP, hardware/software details
- **README.sources file (for packaged BSPs)**
  - meta-<bsp_name>/README.sources
  - Location of the sources that generated the bundled image(s)
- **Pre-built binaries (for packaged BSPs)**
  - meta-<bsp_name>/binary/<bootable_images>
  - Bootable target kernel/rootfs, typically minimal and sato
- **Layer configuration file**
  - meta-<bsp_name>/conf/layer.conf
  - Defines the BSP as a Yocto layer

BSP Components (cont'd)

- **Machine configuration file(s)**
  - meta-<bsp_name>/conf/machine/*.conf
  - Machine-specific parameters: kernel choice, machine and kernel features, bootloader info, image format info, compiler tuning options
- **Recipe files and recipe extensions (.bb and .bbappends)**
  - meta-<bsp_name>/recipes-bsp/*
  - meta-<bsp_name>/recipes-core/*
  - meta-<bsp_name>/recipes-graphics/*
- **Kernel recipes and extensions (.bb and .bbappends)**
  - meta-<bsp_name>/recipes-kernel/linux/linux-x_y*.bb
  - meta-<bsp_name>/recipes-kernel/linux/linux-korg.bb
  - meta-<bsp_name>/recipes-kernel/linux/linux-yocto*.bbappend
- **Kernel patches and configuration**
  - meta-<bsp_name>/recipes-kernel/linux/linux-x_y/*.patch | *defconfig
  - meta-<bsp_name>/recipes-kernel/linux/linux-korg/*.patch | *.cfg
  - meta-<bsp_name>/recipes-kernel/linux/linux-yocto/*.patch | *.cfg

BSP Layout

- **If you look closely at the BSP Guide, you see:**
  - Mostly everything is optional
  - But what you do include should fit the format
- **Advantages of the current format:**
  - The layout fits naturally with the poky/oe-core layout
  - Listing standard files such as README help overall quality
  - Including binaries, instructions, and metadata in a standard way is convenient for new users
  - The standard format makes it easier to create BSP tooling

BSP Release Process

- **BSPs can be made available on the Yocto site**
  - http://www.yoctoproject.org/download/all?keys=&download_type=1&download_version=
- **Here's the process:**
  - https://wiki.yoctoproject.org/wiki/Third_Party_BSP_Release_Process
  - Submit the BSP for review on the Yocto mailing list
  - Agree to be the maintainer of the BSP
  - Work with Yocto release engineer on release logistics
  - The BSP will then be hosted (and announced if desired)
  - You can also host it yourself and link from the Yocto page

Kernels

- **The kernel is just another package, right?**
  - Yes and no
  - Yes, it's represented by a recipe just like everything else
  - But it's so central and has so many configurable options that need continual tweaking that it's in fact very different
  - Most recipes are 'set it and forget it'
  - The kernel is not
  - So an accordingly powerful means of interacting with it is necessary
- **Yocto has several kernel options:**
  - User-defined kernel recipe (kernel.org tarball for example)
  - Recipe to 'yoctoize' kernel.org git (or your own git kernel)
  - 'Yocto' kernels

User-defined Kernel Recipe

- **A Yocto recipe just like any other**
  - The SRC_URI points to a tarball like any other recipe does
  - It's patched via the SRC_URI just like any other recipe is

```
$ cat linux_3.0.18.bb

DESCRIPTION = "Mainline Linux Kernel"
SECTION = "kernel"
LICENSE = "GPLv2"

LIC_FILES_CHKSUM = "file://COPYING;md5=d7810fab7487fb0aad327b76f1be7cd7"

inherit kernel

SRC_URI = "${KERNELORG_MIRROR}/linux/kernel/v3.0/linux-$
  {PV}.tar.bz2;name=kernel \
           file://defconfig "
SRC_URI += "file://yocto-testmod.patch"

SRC_URI[kernel.md5sum] = "67252770d7009eabe8bac7c26e074f9d"
```

User-defined Kernel Recipe (cont'd)

- **Suppose we have a patch and couple config items**
  - drivers/misc/yocto-testmod.patch
    - A test module that prints a silly message on module_init()
  - A couple options to turn it on
    - CONFIG_MISC_DEVICES = y and CONFIG_YOCTO_TESTMOD = y
- **To apply the patch and turn it on:**
  - We need to add the patch to the SRC_URI
    - `SRC_URI += "file://yocto-testmod.patch"`
  - And add the kernel options directly to the defconfig
    - `SRC_URI += "file://defconfig"`
- **'config fragments' not available**
  - defconfig is a simple list of config items
  - May be difficult to separate what was configured automatically by kbuild versus what was set by a user

'Yoctoized' Arbitrary Kernel Recipe

- **linux-korg.bb is a kernel recipe in poky-extras**
  - It's not officially supported (but will be, discussed later)
  - All the yocto-specific variables have been removed

```
$ cat poky-extras/meta-kernel-dev/recipes-kernel/linux/linux-korg.bb

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

KMACHINE = "yocto/standard/auto-bsp"
YOCTO_KERNEL_EXTERNAL_BRANCH ?= "yocto/standard/auto-bsp"

KBRANCH = ${KMACHINE}
KMETA = meta

SRC_URI = "git:///home/kernellab/lab1/linux;protocol=file;nocheckout=1"
SRC_URI += " file://defconfig file://yocto-testmod.patch \
        file://yocto-testmod.cfg"
SRCREV=${AUTOREV}

LINUX_VERSION_EXTENSION ?= "-yoctized-${LINUX_KERNEL_TYPE}"
# Functionality flags
KERNEL_REVISION_CHECKING=
YOCTO_KERNEL_META_DATA=

require recipes-kernel/linux/linux-tools.inc
```

'Yoctoized' Arbitrary Kernel Recipe (cont'd)

- **linux-korg.bb 'yoctoizes' any arbitrary kernel**
  - Defaults to Linus' git tree:
    - SRC_URI = "git://git.kernel.org/.../torvalds/linux.git;protocol=git"
  - But we can point it to any other kernel repo:
    - SRC_URI = "git:///home/kernellab/lab1/linux.git;protocol=file"
- **It can be customized via the SRC_URI**
  - SRC_URI += "file://defconfig file://yocto-testmod.patch \
    file://yocto-testmod.cfg"
- **We can use 'config fragments'**
  - yocto-testmod.cfg adds CONFIG_MISC_DEVICES and CONFIG_YOCTO_TESTMOD
- **We can create and use 'kernel features':**
  - Config fragments and kernel patches in one
    - KERNEL_FEATURES_append = "features/yocto-testmod"
- **Enables Yocto 'kernel tooling' for any git kernel**

linux-yocto_3.2.bb

```
$ cat meta/recipes-kernel/linux/linux-yocto_3.2.bb

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

KMACHINE = "common-pc"
KMACHINE_qemux86  = "common-pc"

KBRANCH = "standard/default/base"
KBRANCH_qemux86  = "standard/default/common-pc/base"

SRCREV_machine_qemuppc ?= "74364f1062a219eb242d7cb300a404516c297601"
SRCREV_machine ?= "6f164ae4ef5aeec2bef40a1b936ac1f9b9db46ba"
SRCREV_meta ?= "8295227f068f78ec3c433529e4012a38773a88c9"

SRC_URI = "git://git.yoctoproject.org/linux-yocto-
  3.2;protocol=git;bareclone=1;branch=${KBRANCH},meta;name=machine,meta"

KERNEL_FEATURES="features/netfilter"
KERNEL_FEATURES_append=" features/taskstats"
KERNEL_FEATURES_append_qemux86=" cfg/sound"

require linux-tools.inc
```

Yocto Kernel Recipe (cont'd)

- **A 'Yocto' kernel is just a kernel.org kernel**
  - Inside a repo with other branches
  - The starting point is a kernel.org kernel snapshot
    - This is the 'master' branch – it's a snapshot and never changes
  - Other branches inherit this branch and add commits
  - yocto/base inherits 'master'
    - All other branches normally inherit this
    - This is where 'stable' is merged or anything global like security
  - yocto/standard/base inherits yocto/base
    - This adds really common stuff all BSPs normally want
    - It also inherits everything from yocto/base i.e stable, security
  - yocto/standard/common-pc inherits yocto/standard/base
    - This adds really common stuff all 'common pc's' normally want
    - It also inherits all of yocto/standard/base (and in turn yocto/base)
  - Finally, your BSP branch can inherit from any of the above

Yocto Kernel Recipe (cont'd)

- **Inherited branches get all updates for free**
  - Any BSP based on a Yocto kernel automatically gets stable updates for instance
- **The 'meta' branch represents configuration**
  - Groupings of common config settings as 'fragments'
    - These can be added as .cfg files to the kernel SRC_URI
  - Groupings of common config/patches as 'features'
    - These can be added via recipe-space KERNEL_FEATURE appends
    - See meta/kernel-cache/features and ../cfg for the available list
  - This allows these settings to be used as a group between BSPs and allows them to be independently added
  - Each BSP also has a specific starting configuration in meta
    - See meta/kernel-cache/bsp/<bsp-name>/<bsp>-<ktype>.scc
    - The tools find the .scc that matches MACHINE/KTYPE/parent branch
    - That starts the process of including the configs inherited up the chain
    - Also a place to hard-code features and config fragments, avoiding SRC_URI

Yocto Kernel Branches and Updates

Branch	Commit message	Author
master	Merge git://git.kernel.org/pub/scm/linux/kernel/git/davem/sparc	Linus Torvalds
meta	meta: bumping kver to v3.0.24	Bruce Ashfield
yocto/base	Merge commit 'v3.0.24' into yocto/base	Bruce Ashfield
yocto/eg20t	WR Linux baseline	Bruce Ashfield
yocto/emgd	yocto/emgd: 3.0 fixes	Tom Zanussi
yocto/emgd-1.10	yocto/emgd: initial build fixups	Tom Zanussi
yocto/gma500	WR Linux baseline	Bruce Ashfield
yocto/pvr	yocto/pvr: add hdmi/acpi definitions	Kishore Bodke
yocto/standard/arm-versatile-926ejs	Merge branch 'yocto/standard/base' into yocto/standard/arm-versatile-926ejs	Bruce Ashfield
yocto/standard/base	Merge branch 'yocto/base' into yocto/standard/base	Bruce Ashfield
yocto/standard/beagleboard	Merge branch 'yocto/standard/base' into yocto/standard/beagleboard	Bruce Ashfield
yocto/standard/cedartrail	Merge branch 'yocto/standard/base' into yocto/standard/cedartrail	Bruce Ashfield
yocto/standard/common-pc-64/base	Merge branch 'yocto/standard/base' into yocto/standard/common-pc-64/base	Bruce Ashfield
yocto/standard/common-pc-64/jasperforest	Merge branch 'yocto/standard/base' into yocto/standard/common-pc-64/jasperforest	Bruce Ashfield
yocto/standard/common-pc-64/romley	Merge branch 'yocto/standard/base' into yocto/standard/common-pc-64/romley	Bruce Ashfield
yocto/standard/common-pc-64/sugarbay	Merge branch 'yocto/standard/base' into yocto/standard/common-pc-64/sugarbay	Bruce Ashfield
yocto/standard/common-pc/atom-pc	Merge branch 'yocto/standard/base' into yocto/standard/common-pc/atom-pc	Bruce Ashfield
yocto/standard/common-pc/base	Merge branch 'yocto/standard/base' into yocto/standard/common-pc/base	Bruce Ashfield
yocto/standard/crownbay	Merge branch 'yocto/standard/base' into yocto/standard/crownbay	Bruce Ashfield
yocto/standard/emenlow	Merge branch 'yocto/standard/base' into yocto/standard/emenlow	Bruce Ashfield
yocto/standard/fishriver	Merge branch 'yocto/standard/base' into yocto/standard/fishriver	Bruce Ashfield
yocto/standard/fri2	Merge branch 'yocto/standard/base' into yocto/standard/fri2	Bruce Ashfield
yocto/standard/fsl-mpc8315e-rdb	Merge branch 'yocto/standard/base' into yocto/standard/fsl-mpc8315e-rdb	Bruce Ashfield
yocto/standard/mti-malta32-be	Merge branch 'yocto/standard/base' into yocto/standard/mti-malta32-be	Bruce Ashfield
yocto/standard/mti-malta32-le	Merge branch 'yocto/standard/base' into yocto/standard/mti-malta32-le	Bruce Ashfield
yocto/standard/pandaboard	v3.0.24 -> pandaboard merge fixup	Bruce Ashfield

Some new Yocto kernel Changes

- **New linux-yocto-custom.bb recipe**
  - An easy-to-use entry point for taking advantage of Yocto kernel tooling
  - Provides a Yocto-jargon-free mechanism to use your own kernel
  - Well-documented
- **Ability to use an 'externalsrc' kernel**
  - Point the kernel recipe to an existing local kernel tree
  - The build system will compile and deploy it but 'keep its hands off' otherwise
  - Allows you to do kernel development 'the old fashioned way'
- **'master' is the default when creating new BSPs using linux-yocto**
  - The 'master' branch will point to the current 'stable' Linux kernel
  - So when you create a new BSP, you're simply basing on current 'stable'
  - You'll have to opt-in to base on Yocto branches like 'standard/base'
- **General branch cleanup**
  - 'meta' no longer has confusing upstream kernel commits
- **Tool to generate patchset from git branches**
  - To make it easier to see changes in patch form

The new 'Yoctoized' Arbitrary Kernel Recipe

- **linux-yocto-custom.bb**
  - Similar to linux-korg.bb but an official recipe
  - All the Yocto-specific variables have been removed
  - Defaults to the kernel.org kernel (so replaces linux-korg.bb)
  - Defaults to arch defconfig if no defconfig specified

```
$ cat poky-extras/meta-kernel-dev/recipes-kernel/linux-yocto-custom.bb

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

# point this to the git repository of choice
SRC_URI = "git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git;protocol=git;nocheckout=1"

SRCREV="${AUTOREV}"
LINUX_VERSION ?= "3.3"
LINUX_VERSION_EXTENSION = "-custom"

PR = "r0"
PV = "${LINUX_VERSION}+git${SRCPV}"

COMPATIBLE_MACHINE = "(qemuarm|qemux86|qemuppc|qemumips|qemux86-64)"

require recipes-kernel/linux/linux-tools.inc
```

The new 'Yoctoized' Arbitrary Kernel Recipe

- **Select the custom kernel as the BSP's kernel**
  - PREFERRED_PROVIDER_virtual/kernel = "linux-yocto-custom"
- **Customize it via .bbappend**
  - Point to your own kernel repo
  - Add defconfig or config fragments or kernel features
- **Enables any git-based kernel repo to be the upstream kernel**
  - Use your own repo but gain ability to use fragments and features
  - Work towards something that could be sent to linux-yocto
  - Or just as a local collection of features and metadata in your own repo

```
$ cat meta-mybsp/recipes-kernel/linux/linux-yocto-custom.bbappend

SRC_URI = "git://myrepo.github.org/linux.git;protocol=git;nocheckout=1"

SRC_URI += " file://defconfig \
             file://yocto-testmod.patch \
             file://yocto-testmod.cfg"

KERNEL_FEATURES_append=" features/testmod"


SRCREV="c16fa4f2ad19908a47c63d8fa436a1178438c7e7"
```

A few examples from meta-intel

- **EMGD**
  - Large kernel patch (5Mb), will never be upstreamed
  - Used by many meta-intel BSPs
  - We don't want to add this patch to every recipe
  - With a 'user-defined' kernel recipe each BSP would duplicate the patch
  - With linux-yocto, we don't have to. Instead we can:
    - Put the EMGD driver into a topic branch (emgd-1.10)
    - Use 'git merge' to merge it into a given BSP

```
$ cat linux-yocto-3.2/meta/cfg/kernel-cache/bsp/crownbay/crownbay-standard.scc

define KMACHINE crownbay
define KTYPE standard
define KARCH i386

include ktypes/standard
branch crownbay

include features/emgd/emgd-1.10.scc
git merge emgd-1.10

include crownbay.scc
```

A few examples from meta-intel

- **Intel power settings**
  - Most Intel BSPs share a bunch of common power settings
  - It's very convenient to have them visible as a group
  - Any BSP that wants to use them simply includes them:
    - include features/power/intel.scc (includes intel.cfg)

```
$ cat linux-yocto-3.2/meta/cfg/kernel-cache/features/power/intel.cfg

# use the native intel cpuidle driver for recent Intel processors
CONFIG_INTEL_IDLE=y

# cut out the top source of unnecessary wakeups
CONFIG_NO_HZ=y

# enable apps to cut down on polling
CONFIG_INOTIFY_USER=y

# enable cpu frequency scaling and stats for powertop
# enable power management debugging for tools like powertop
...

# turn on run-time power management
CONFIG_PM_RUNTIME=y

# allow usb runtime power management
CONFIG_USB_SUSPEND=y
```

A few examples from meta-intel

- **In general, each BSP is different**
  - Some BSPs have local groupings they like to keep straight
  - Each BSP might have its own set of one-off config settings
  - Most BSPs also include some non-hardware features

```
$ cat linux-yocto-3.2/meta/cfg/kernel-cache/bsp/emenlow/emenlow.scc
kconf non-hardware reboot-quirk.cfg

$ cat linux-yocto-3.2/meta/cfg/kernel-cache/bsp/emenlow/reboot-quirk.cfg
CONFIG_CMDLINE_BOOL=y
CONFIG_CMDLINE="reboot=pci"

$ cat linux-yocto-3.2/meta/cfg/kernel-cache/bsp/jasperforest/jasperforest.cfg
# NUMA support
CONFIG_NUMA=y
CONFIG_X86_64_ACPI_NUMA=y
CONFIG_NODES_SPAN_OTHER_NODES=y
CONFIG_USE_PERCPU_NUMA_NODE_ID=y
CONFIG_ACPI_NUMA=y

$ cat linux-yocto-3.2/meta/cfg/kernel-cache/bsp/fri2/fri2-standard.scc
include features/latencytop/latencytop.scc
include features/profiling/profiling.scc
include cfg/efi-ext.scc
```

Yocto BSP Tools

- **A new set of tools to help users:**
  - Start a new BSP
  - Manage kernel patches and config options
- **'yocto-bsp' creates an initial Yocto BSP**
  - Creates an initial buildable image that may or may not boot
  - Just a starting point, ultimately the user must make it work
  - With a standardized BSP format, tooling is easier
- **'yocto-kernel' allows users to add and remove patches and config items from the command line**
  - Lots of context required to deal directly with linux-yocto metadata
  - Managing patches and .cfg items is tedious and error-prone
  - Dealing with multiple branches adds to the confusion
  - 'yocto-kernel' abstracts the details with a friendly guided interface
    - Users never have to know anything about the internals of the kernel recipe

Create a BSP using 'yocto-bsp'

```
trz@elmorro:/usr/local/dev/Yocto$ yocto-bsp create myintelbsp x86_64

Would you like to use the default (3.2) kernel? (y/n) [default: y]
Do you need a new machine branch for this BSP (the alternative is to re-use an existing
  branch)? [y/n] [default: y]
Getting branches from remote repo git://git.yoctoproject.org/linux-yocto-3.2...
Please choose a machine branch to base this BSP on: [default: standard/default/common-pc-
  64]
3) standard/default/arm-versatile-926ejs
4) standard/default/base
8) standard/default/common-pc-64/jasperforest
17) standard/default/fsl-mpc8315e-rdb
20) standard/default/preempt-rt

Do you need SMP support? (y/n) [default: y]
Do you need support for X? (y/n) [default: y]
Please select an xserver for this machine: [default: xserver_i915]
1) VESA xserver support
2) i915 xserver support

Does your BSP have a touchscreen? (y/n) [default: n]
Does your BSP have a keyboard? (y/n) [default: y]

New x86_64 BSP created in meta-myintelbsp
```

yocto-bsp

- **yocto-bsp and yocto-kernel are Python scripts**
  - They live under top-level yocto repo scripts/ dir
  - The main implementation is under scripts/lib/bsp
  - BSP template files live under scripts/lib/bsp/substrate/target/arch/XXX
  - There's a subdirectory for each Yocto arch, plus 'common' and 'qemu'

```
$ find scripts

scripts/yocto-bsp
scripts/yocto-kernel
scripts/lib/bsp/engine.py
scripts/lib/bsp/substrate/target/arch/common/README
scripts/lib/bsp/substrate/target/arch/common/conf/layer.conf
scripts/lib/bsp/substrate/target/arch/qemu/recipes-kernel/linux/files/{{=machine}}-
  standard.scc
scripts/lib/bsp/substrate/target/arch/arm/conf/machine/{{=machine}}.conf
scripts/lib/bsp/substrate/target/arch/powerpc/recipes-kernel/linux/{{ if
  kernel_choice == "linux-yocto_3.2": }} linux-yocto_3.2.bbappend
scripts/lib/bsp/substrate/target/arch/i386/recipes-graphics/xorg-xserver/xserver-
  xf86-config/{{=machine}}/{{ if xserver_choice == "xserver_i915": }} xorg.conf
```

Yocto-bsp (cont'd)

- **Each file in 'substrate/target/arch/*' is a template file**
  - The files are copied into the target BSP
    - Tags of the form {{=var}} are replaced by that variable
    - Conditional inclusion, looping, etc are accomplished by in-line Python inside {{ tags }}
  - yocto-bsp creates a BSP-generating Python program that when executed writes the BSP files with variable substitution and logic
  - Tags of the form {{ input ... }} generate user input elements

```
$ cat scripts/lib/bsp/substrate/target/arch/i386/conf/machine/{{=machine}}.conf

#@TYPE: Machine
#@NAME: {{=machine}}

#@DESCRIPTION: Machine configuration for {{=machine}} systems

{{ preferred_kernel_version = kernel_choice.split('_')[1] }}
PREFERRED_VERSION_{{=preferred_kernel}} ?= "{{=preferred_kernel_version}}%"

{{ input type:"boolean" name:"xserver" prio:"50" msg:"Do you need support for X?
  (Y/n)" default:"y" }}
{{ if xserver == "y": }}
XSERVER ?= "${XSERVER_IA32_BASE} ${XSERVER_IA32_EXT}”
```

Mod the kernel with 'yocto-kernel'

```
$ yocto-kernel patch add myqemuarm /home/trz/newpatches/yocto-testmod.patch
Added patches:
yocto-testmod.patch

$ yocto-kernel config add myqemuarm CONFIG_MISC_DEVICES=y
Added items:
CONFIG_MISC_DEVICES=y

$ yocto-kernel patch rm myqemuarm
Specify the patches to remove:
1) yocto-testmod.patch
1
Removed patches:
yocto-testmod.patch

$ yocto-kernel config rm myqemuarm
Specify the kernel config items to remove:
1) CONFIG_MISC_DEVICES=y
2) CONFIG_YOCTO_TESTMOD=y
1
Removed items:
CONFIG_MISC_DEVICES=y
```

yocto-kernel

- **Everything yocto-kernel does is visible in the SRC_URI of the BSP's kernel recipe .bbappend**
  - The items will appear either directly in the SRC_URI or in a file named in recipes-kernel/linux/files/

```
$ cat meta-foo/recipes-kernel/linux/linux-yocto_3.2.bbappend

SRC_URI += " \
    file://foo-standard.scc \
    file://foo.scc \
    file://foo.cfg \
    file://user-config.cfg \
    file://user-patches.scc \
    file://yocto-testmod.patch \
"
$ cat meta-foo/recipes-kernel/linux/files/user-config.cfg

CONFIG_MISC_DEVICES=y
CONFIG_YOCTO_TESTMOD=y
```

Discussion

Resources

- **The Yocto Project BSP Developer's Guide**
  - http://www.yoctoproject.org/docs/current/bsp-guide/bsp-guide.html
- **The Yocto Project BSP Tools Documentation**
  - https://wiki.yoctoproject.org/wiki/Yocto_BSP_Tools_Documentation
- **yocto-bsp qemu BSP Creation Walk-through**
  - https://wiki.yoctoproject.org/wiki/Transcript:_Using_the_Yocto_BSP_tools_to_create_a_qemu_BSP
- **yocto-kernel Patch and Config Item Walk-through**
  - https://wiki.yoctoproject.org/wiki/Transcript:_Using_the_Yocto_BSP_tools_to_manage_kernel_patches_and_config_items
- **Yocto-bsp meta-intel BSP Creation Walk-through**
  - https://wiki.yoctoproject.org/wiki/Transcript:_Using_the_Yocto_BSP_tools_to_create_a_meta-intel_BSP
- **Yocto Third Party BSP Release Process**
  - https://wiki.yoctoproject.org/wiki/Third_Party_BSP_Release_Process
- **yocto-testmod.patch and yocto-testmod.cfg**
  - https://wiki.yoctoproject.org/wiki/Yocto_BSP_Summit_Presentation


yocto
PROJECT