Building for Heterogeneous Systems
Outline

- Heterogeneous Devices
- Linux Flow vs Baremetal Flow
- Baremetal / RTOS build
- Working with SDKs (Rpi Pico / Zephyr)
- Bitbake’s Solution (Multiconfig)
- Testing
Heterogeneous Devices
Heterogeneous Devices

- Multiple Architectures on the same device
  - Typically “Big” core runs Linux, “small” core runs RTOS/Baremetal
- Power Consumption
- Safety Critical
- Processing power
- Efficiency
Heterogeneous Builds

- Linux
- Linux + Baremetal
- Linux + RTOS
- Baremetal
- RTOS
- Linux + Linux (RT/Tiny)
Heterogeneous Builds

- Create cross toolchain for arch X
- Create cross toolchain for arch Y
- Cross compile package A for arch X
- Cross compile package B for arch Y
- Package Images / Binaries to be flashed
Developer Workflows
Linux Flow vs Baremetal Flow

Linux:
- Git repository
- Toolchain (host)
- IDE of your choice

Baremetal:
- Git repository
- Vendor provided toolchain
- Vendor provided SDK
- Vendor provided IDE

Devtool works
Baremetal Toolchain

ARM Embedded Toolchain

- GCC
- Binutils
- Newlib

Bitbake can provide it by using TCLIBC="newlib"
Baremetal / RTOS
Baremetal / RTOS

1.- Clone the required repositories (Use -b dunfell or -b kirkstone if you want a stable release)

```bash
$ git clone https://git.yoctoproject.org/git/poky
$ cd poky
```

2.- Source the build environment file

```bash
$ source oe-init-build-env
```

3.- Add the required variables to your local.conf (Supported MACHINES are qemuarm64,qemuarm,armuarmv5)

```bash
$ echo "MACHINE = "\"qemuarm64\"" >> ./conf/local.conf
$ echo "TCLIBC = "\"baremetal\"" >> ./conf/local.conf
```

4.- Build the application

```bash
$ bitbake baremetal-helloworld
```

https://github.com/ahcbb6/baremetal-helloworld
5. Run the baremetal application on QEMU:

```
$ qemu -nographic
```

Example output:

```
rqemu - INFO - Running bitbake -e ...
rqemu - INFO - Continuing with the following parameters:
KERNEL: [tmp/deploy/images/qemuarm64/baremetal-helloword-qemuarm64.bin]
MACHINE: [qemuarm64]
FSTYPE: [bin]
ROOTFS: [tmp/deploy/images/qemuarm64/baremetal-helloword-qemuarm64.bin]
CONFFILE: [tmp/deploy/images/qemuarm64/baremetal-helloqemuarm64.qemuboot.conf]

Hello OpenEmbedded!
```
Baremetal / RTOS

1.- Clone the required repositories

```
$ git clone https://git.yoctoproject.org/git/poky
$ cd poky
$ git clone https://github.com/ahcbb6/meta-freertos.git
```

2.- Add meta-freertos to your bblayers.conf

```
$ source oe-init-build-env
$ bitbake-layers add-layer ../meta-freertos
```

3.- Add the required variables to your local.conf

```
$ echo "DISTRO = \"freertos\"" >> ./conf/local.conf
# If building for QEMU use:
$ echo "MACHINE = \"qemuarmv5\"" >> ./conf/local.conf
# If, instead, building for STM32 use:
$ echo "MACHINE = \"stm32f446\"" >> ./conf/local.conf
```

https://github.com/ahcbb6/meta-freertos
Baremetal / RTOS

4.- Build a sample FreeRTOS standalone application:

# For QEMU:
$ bitbake freertos-demo
# For STM32:
$ bitbake freertos-demo-stm32

5.- Run the application on QEMU (or flash the .hex file on the deploy directory for STM32):

$ runqemu nographic
Baremetal / RTOS

A text may be entered using a keyboard.
It will be displayed when 'Enter' is pressed.

Periodic task 10 secs
Waiting For Notification - Blocked...
Task1
Task1
You entered: "HelloFreeRTOS"
Unblocked
Notification Received
Waiting For Notification - Blocked...
Working with SDKs
Working with SDKs (Raspberry Pi Pico / Zephyr)

Approach 1:

- Create a native recipe to provide an SDK
  - Fetch
  - Install
- DEPEND on it on application recipe
- Use the SDK from recipe-sysroot whilst building the application

https://git.yoctoproject.org/meta-zephyr/tree/meta-zephyr-core/recipes-devtools/zephyr-sdk/zephyr-sdk_0.16.3.bb
Working with SDKs (Raspberry PI Pico / Zephyr)

Approach 2:

- On application recipe
  - Fetch app source code
  - Fetch SDK
  - Wire and use SDK
Bitbake Multiconfig
Manually configure bitbake to parse an additional conf

Multiconfig dependencies allow bitbake to use the same build

Shared State can be reused across multiconfigs (native)
Bitbake Multiconfig Builds

local.conf

```
MACHINE="stm32f446"
DISTRO="freertos"
BBMULTICONFIG = "dummy-aarch64"
```

multiconfig/dummy-aarch64.conf

```
MACHINE="qemuarm64"
DISTRO="poky"
TMPDIR="$\{TOPDIR\}/tmp-$\{MACHINE\}-$\{TCLIBC\}"
```

core-image-minimal.bb

```
do_image[mcdepends] = "multiconfig:dummy-aarch64::freertos-demo-stm32::do_image"
```
Bitbake Multiconfig Builds

Build Configuration (mc:default):
- BB_VERSION = "2.6.0"
- BUILD_SYS = "x86_64-linux"
- NATIVESBSTRING = "universal"
- TARGET_SYS = "arm-oe-eabi"
- MACHINE = "stm32f446"
- DISTRO = "freertos"
- DISTRO_VERSION = "1.0"
- TUNE_FEATURES = "armv7em cortexm4 thumb callconvention-hard vfpv4spid16"
- TARGET_FPU = "hard"

Build Configuration (mc:dummy):
- BB_VERSION = "2.6.0"
- BUILD_SYS = "x86_64-linux"
- NATIVESBSTRING = "universal"
- TARGET_SYS = "aarch64-poky-linux"
- MACHINE = "qemuarm64"
- DISTRO = "poky"
- DISTRO_VERSION = "4.3+snapshot-a9befd527e173a0b8d7e684691e2b87e5c1587baa"
- TUNE_FEATURES = "aarch64 armv8a crc cortexa57"
- TARGET_FPU = ""
Bitbake Multiconfig Builds

NOTE: Executing Tasks
Setscene tasks: 1319 of 1319
Currently 2 running tasks (3707 of 3728) 99% |######################################
0: freertos-demo-FreeRTOSv10.5.1+gitAUTOINC+391c79958f_4b08f3a14b-r0 do_fetch - 4s (pid 3857779)
1: mc:dummy-aarch64:core-image-minimal-1.0-r0 do_rootfs - 1s (pid 3857915) 3% |##
Bitbake Multiconfig Builds

```
$ ls tmp-qemuarm64-glibc/deploy/images/qemuarm64/

  Image
  Image--6.5.10+git0+e4aaaaddfa_e769bc7ca8-r0-qemuarm64-20231114055859.bin
  Image-qemuarm64.bin
  core-image-minimal-qemuarm64.rootfs-20231114150137.ext4
  core-image-minimal-qemuarm64.rootfs-20231114150137.manifest
  core-image-minimal-qemuarm64.rootfs-20231114150137.qemuboot.conf
  core-image-minimal-qemuarm64.rootfs-20231114150137.spdx.tar.zst
  core-image-minimal-qemuarm64.rootfs-20231114150137.tar.bz2
  core-image-minimal-qemuarm64.rootfs-20231114150137.testdata.json
  core-image-minimal-qemuarm64.rootfs.ext4
  core-image-minimal-qemuarm64.rootfs.manifest
  core-image-minimal-qemuarm64.rootfs.qemuboot.conf
```

```
$ ls tmp/deploy/images/stm32f446/

  freertos-image-stm32f446-20231114151318.manifest
  freertos-image-stm32f446-20231114151318.qemuboot.conf
  freertos-image-stm32f446-20231114151318.testdata.json
  freertos-image-stm32f446-20231114151318.manifest
  freertos-image-stm32f446-20231114151318.qemuboot.conf
  freertos-image-stm32f446-20231114151318.testdata.json
  freertos-image-stm32f446-20231114151318.manifest
  freertos-image-stm32f446-20231114151318.qemuboot.conf
  freertos-image-stm32f446-20231114151318.testdata.json
```

Testing
Testing multiple OSs

- Testing different OS can be done using OpenEmbedded infrastructure (to some extent)
  - Designed for Linux
  - Emulate expectations from other OS
  - Treated as separate builds
    - `bitbake mc:big:core-image-minimal -c testimage`
    - `bitbake mc:small:baremetal-app -c testimage`
Future
Closing thoughts

- Building your own cross toolchain provides some advantages
  - Newer versions
  - Quickly fixed vs Waiting for fix to cascade
- Vendor IDE may not use upstream toolchain
- Multiconfig requires manual configuration
  - Describes the build, not the system/product.
  - Can we describe the system?
- Complicated to integrate into vendor workflows
  - How can we integrate better?