Yocto Project® Size reduction techniques

KHEM RAJ

Yocto Project Virtual Summit Europe, October 29-30, 2020
Agenda

- Introducti on
- "poky- tiny" Distribution
- Image analysis tools
- Compiler and toolchain optimizations
- C/C++ this time
- Package refactoring
- Package selection
Introduction

• **Reference Distributions (meta-poky/conf/distro)**
  • **poky** – Standard default distribution
  • **poky-tiny** – Geared towards small size
    Uses musl, linux-yocto-tiny, mdev, custom init
  • **poky-altconfig** – Uses systemd, LTSS kernel

• **Most of examples here are based on yoe distribution**
  • Derives from poky distribution policies
poky-tiny

- Use to seed your custom distro

```sh
require conf/distro/poky-tiny.conf
DISTRO = "my-tiny-distro"
...
```
- Use musl C library
  - IMAGESIZE = 37668
  + IMAGESIZE = 35532

- Use busybox init system

- Use mdev instead of udev.
  - See if you can live without it

- IMAGESIZE = 35532
  + IMAGESIZE = 3120
Compiler Flags

- Check global optimization flags

**SELECTED_OPTIMIZATION**

```
TARGET_CFLAGS = "${TARGET_CPPFLAGS} ${SELECTED_OPTIMIZATION}"
```

```
DEBUG_FLAGS = "-g -feliminate-unused-debug-types ${DEBUG_PREFIX_MAP}"
```

```
# Disabled until the option works properly
FELIMINATE = "-O2 -pipe ${DEBUG_FLAGS}"
DEBUG_OPTIMIZATION = "-Og ${DEBUG_FLAGS} -pipe"
SELECTED_OPTIMIZATION = "${@d.getVar(oe.utils.vartrue('DEBUG_BUILD', 'DEBUG_OPTIMIZATION', 'FULL_OPTIMIZATION', d))}"
```

- Use `-Os` instead of `-O2`

```
-IMAGESIZE = 3120
+IMAGESIZE = 2992
```
Compiler Flags

- `fno-function-sections`
- `fno-unroll-loops`
- `fno-exceptions`
- `fno-jump-tables`
Code

- Libraries have functions defined in headers
- Use explicit template instantiations
  - Reduces code size
  - Faster to compile
- Commonly used functions can use attributes to avoid inlining
  ```c
  static int __attribute__((noinline)) foo(void *arg)
  ```
Code

• **Do not define functions in headers**
  • They will be candidates for inlining
  • Constructors/destructors
  • Operator overloading

• **Use pragmas/function attributes to help compiler**
  • Note: They may be compiler dependent
    ```c
    int foo (int) __attribute__ ((cold));
    int bar (int) __attribute__ ((optimize(0)));
    ```

• **Use cheaper data structs**
  • std::list instead of std::vector
Code

- Use compiler built-ins wherever possible
  - Your memcpy is not better than compiler’s

- Use standard C library functions
  - They are better optimized
Modern compilers support LTO (clang/gcc)

Enable `-flto` switch

```
LTO_pn-glibc = ""
LTO_pn-gcc-runtime = ""
LTO_pn-libgcc-initial = ""
LTO_pn-libgcc = ""
LTO_pn-libpam = ""
LTO_pn-elfutils = ""
LTO_pn-perl = ""
LTO_pn-busybox = ""
LTO_pn-libxcrypt = ""
LTO_pn-curl = ""
LTO_pn-libcap = ""
LTO_pn-python3 = ""
LTO_pn-libproxy = ""

LTO += "-flto"

SELECTED_OPTIMIZATION_append_class-target = " ${LTO}"
TARGET_LDFLAGS_append_class-target = " ${LTO}"

SELECTED_OPTIMIZATION[vardeps] += "LTO"

PACKAGE_DEBUG_SPLIT_STYLE = "debug-without-src"
```

- IMAGESIZE = 2992
+ IMAGESIZE = 2956
LTO

- Clang support (needs work for arm arch)
  - Thin LTO
    1. Faster to compile
  - Full LTO
    1. More optimizations
  - Needs gold linker
    1. Gold ld + LTO does not like “Os” (Use O2 or O3)

# Enable LTO based on global distro settings
TOOLCHAIN_OPTIONS_append_toolchain-clang = "${@bb.utils.contains('DISTRO_FEATURES', 'thin-lto', ' -flto=thin -fuse-ld=gold', '', d)}"
TOOLCHAIN_OPTIONS_append_toolchain-clang = "${@bb.utils.contains('DISTRO_FEATURES', 'full-lto', ' -flto=full -fuse-ld=gold', '', d)}"
Check Image

- **Enable buildhistory**

  ```
  USER_CLASSES ?= "buildhistory"
  BUILDHISTORY_COMMIT ?= "1"
  ```

- **Build Image**
Refine Image

- **IMAGE_LINGUAS**
  - “en-us”
  - Use “as needed”, prefer empty

- **IMAGE_FEATURES**
  - Remove package-management
  - Remove splash
  - SSH – Use dropbear instead of OpenSSH
    1. Add ssh-server-dropbear to IMAGE_FEATURES
For ARM 32-bit

- Ensure Thumb ISA is used

Select proper DEFAULTTUNE

1. DEFAULTTUNE ?= "cortexa7thf-neon-vfpv4"

Build Configuration:

```
BB_VERSION           = "1.47.0"
BUILD_SYS            = "x86_64-linux"
NATIVELSBSTRING      = "arch"
TARGET_SYS           = "arm-yoe-linux-gnueabi"
MACHINE              = "raspberrypi4"
DISTRO               = "yoe"
DISTRO_VERSION       = "3.1"
TUNE_FEATURES        = "arm vfp cortexa7 neon vfpv4 thumb callconvention-hard"
TARGET_FPU           = "hard"
```

- IMAGESIZE = 3320
+ IMAGESIZE = 2956
Default Tunes

- 16-bit ISA for MIPS
  - Enable MIPS16e ASE instructions
    1. Use mips16e in TUNE_FEATURES
    2. Set MIPS_INSTRUCTION_SET to mips16e
• Buildhistory is in TOPDIR/buildhistory

[kraj@apollo /mnt/b/yoe/master/buildhistory/images/raspberry4/glibc/yoe-simple-image]
% tree
.
├── build-id.txt
├── depends.dot
├── depends-nokernel.dot
├── depends-nokernel-nolibc.dot
├── depends-nokernel-nolibc-noupdate.dot
├── depends-nokernel-nolibc-noupdate-nomodules.dot
├── files-in-image.txt
├── image-files
│   ├── etc
│       ├── group
│       └── passwd
├── image-info.txt
├── installed-package-names.txt
├── installed-package-sizes.txt
└── installed-packages.txt
## Inspect installed-package-sizes.txt

- **Package sizes listed in size order**

<table>
<thead>
<tr>
<th>Size (KiB)</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>3023</td>
<td>bluez5</td>
</tr>
<tr>
<td>2568</td>
<td>shared-mime-info</td>
</tr>
<tr>
<td>2266</td>
<td>libglib-2.0-0</td>
</tr>
<tr>
<td>1652</td>
<td>libc6</td>
</tr>
<tr>
<td>1630</td>
<td>kernel-module-btrfs-5.4.47-v7l</td>
</tr>
<tr>
<td>1609</td>
<td>kernel-module-xfs-5.4.47-v7l</td>
</tr>
<tr>
<td>1397</td>
<td>libunistring2</td>
</tr>
<tr>
<td>1292</td>
<td>kernel-module-ocfs2-5.4.47-v7l</td>
</tr>
<tr>
<td>1202</td>
<td>ofono</td>
</tr>
<tr>
<td>1191</td>
<td>libgnutls30</td>
</tr>
<tr>
<td>1186</td>
<td>libstdc++6</td>
</tr>
<tr>
<td>1123</td>
<td>kernel-module-snd-soc-wm5102-5.4.47-v7l</td>
</tr>
<tr>
<td>1040</td>
<td>shadow</td>
</tr>
<tr>
<td>954</td>
<td>libx11-6</td>
</tr>
<tr>
<td>920</td>
<td>kernel-module-mac80211-5.4.47-v7l</td>
</tr>
<tr>
<td>886</td>
<td>kernel-module-cfg80211-5.4.47-v7l</td>
</tr>
<tr>
<td>884</td>
<td>kernel-module-cifs-5.4.47-v7l</td>
</tr>
</tbody>
</table>
| 883       | wpa-suppli
cant               |
Why a package is pulled into image?
- Buildhistory has the information
  - .dot files
- RHS expresses the dependence
- Search for
  - -> "<package-name>"

```plaintext
digraph depends {
  node [shape=plaintext]
  "96boards-tools" -> "e2fsprogs-resize2fs"
  "96boards-tools" -> "gptfdisk"
  "96boards-tools" -> "parted"
  "96boards-tools" -> "udev"
  "96boards-tools" -> "update-rc.d" [style=dotted]
  "96boards-tools" -> "util-linux"
  "alsa-utils-alsactl" -> "alsa-states" [style=dotted]
  "alsa-utils-alsactl" -> "libasound2"
  "alsa-utils-alsactl" -> "libc6"
  "alsa-utils-alsactl" -> "libasound2"
  "alsa-utils-alsamixer" -> "libc6"
  "alsa-utils-alsamixer" -> "libasound2"
  "alsa-utils-alsamixer" -> "libformw5"
  "alsa-utils-alsamixer" -> "libmenuw5"
  "alsa-utils-alsamixer" -> "libncursesw5"
  "alsa-utils-alsamixer" -> "libpanelw5"
  "alsa-utils-alsamixer" -> "libtinfo5"
  "apmd" -> "initd-functions"
  "apmd" -> "libapm1"
  "apmd" -> "libc6"
  "apm" -> "initd-functions"
  "apm" -> "libapm1"
  "apm" -> "libc6"
  "avahi-daemon" -> "base-files"
  "avahi-daemon" -> "base-passwd"
  "avahi-daemon" -> "libavahi-common3"
  "avahi-daemon" -> "libavahi-core7"
  "avahi-daemon" -> "libc6"
```
Kernel Tweaks

• BSPs may enforce installing all kmods
  • Inspect MACHINE_EXTRA_RRECOMMENDS
    1. Remove "kernel-modules"
    2. Add only needed kmods

• Avoid duplicate kernel image in rootfs
  • RDEPENDS_${KERNEL_PACKAGE_NAME}-base = ""

• Inspect kernel config for unneeded features

• Compile with size optimizations (kernel/configs/tiny.config)
Comparison of disk spaced used

- Musl + Busybox init/dev/login
  - space used in ext4 filesystem on running system using df: 1.9MB
  - adding sizes of files in image from buildhistory: 1.5MB
  - number of files in image: 595
- Busybox init/dev/login
  - space used in ext4 filesystem on running system using df: 3.5MB
  - adding sizes of files in image from buildhistory: 3.1MB
  - number of files in image: 621
- SysVinit
  - space used in ext4 filesystem on running system using df: 4.7MB
  - adding sizes of files in image from buildhistory: 4.3MB
  - number of files in image: 696
- Systemd
  - space used in ext4 filesystem on running system using df: 33.2MB
  - adding sizes of files in image from buildhistory: 22MB
  - number of files in image: 1,806

https://github.com/YoeDistro/yoe-distro/blob/dunfell/docs/libc-init.md
Future Work

- Experiment with toybox as init system
- Link Time Optimization Enabled system builds
- Additional compiler options for size (More than Os)
- More language runtime optimization information
  - Go, rust, python …
A| JãÜì  Zãì Ćşí Ćşí f á X
<Your Section Title>