Intro

Robert Berger

Reliable Embedded Systems e.U.

🅭🅯🄏⊜

October 24, 2018

Table of Contents - Section

Let me introduce myself

ЯELIABLEEMBEDDEDSYSTEMS Consulting Training Engineering

Robert Berger

Embedded Software Specialist

email:

robert.berger@ReliableEmbeddedSystems.com

phone:

+43 (0) 699 17 69 07 19

web:

http://ReliableEmbeddedSystems.com

Building world class world wide win/win cooperations by helping you to create better embedded software!

We trained (excerpt)

SIEMENS · PHILIPS · ABB · LAUTERBACH DEVELOPMENT TOOLS · SIGMA · bayern innovativ · SCS super computing systems

KEYMILE · SSI SCHÄFER · ifm · BOSCH · BRUKER · CYBERCOM GROUP · blueye

SysTec · BELDEN Sending All The Right Signals · CEVA · BENNING · AVL · SysMik GmbH Dresden · BFFT FAHRZEUGTECHNIK

ERICSSON · SEWERIN Technologies for leak detection · AIRBUS GROUP · bachmann. · PAUL SCHERRER INSTITUT PSI · Continental

BT · intel · WIN GD Simply a better different · minimax · EMERSON Network Power · ZOLLER · JUNG

ESG · LIEBHERR · SENSOPART · DELTA Smarter. Greener. Together.

THALES · GÖRLITZ · COGNEX · CEVA · ASP · B/S/H/ BOSCH SIEMENS

On-site training (excerpt)

1 trainee

- public
  - Vienna
  - Munich
  - Zurich

2 and more trainees

- private
  - at your site

More training offers

- The Yocto Project - An Overview (4 days) - http://rlbl.me/yocto
- Introduction to Embedded Linux and the Yocto Project (5 days) - http://rlbl.me/intely
- Embedded Linux: From Systems Architecture to Real-Time (5 days) - http://rlbl.me/elisa
- Embedded GNU/Linux Kernel Internals and Device Drivers (5 days) - http://rlbl.me/ldd
- Introduction to Embedded Linux in Theory and Practice - a Crash Course (3 days) - http://rlbl.me/elin
- (Embedded) Linux Debugging (min. 2 days) - http://rlbl.me/lindeb
- FreeRTOS in Theory and Practice (3 days) - http://rlbl.me/freertos

(Fun with) Libraries/SDK and OE/the YP

Robert Berger

Reliable Embedded Systems e.U.

🅭🅯🄏🅭

October 24, 2018

Table of Contents - Session

Table of Contents - Section

Software Developer

Figure: (normal) developer

- user-space
  - application(s)/tools
  - libraries
- kernel
  - kernel config
  - device drivers
  - fdt

Yocto Person

- takes stuff (and requests) from the "Software Developer"
  - user-space
    - application(s)/tools
    - libraries
  - kernel
    - kernel config
    - device drivers
    - fdt
- takes the "Yocto Project"
  - (it’s not an embedded Linux distribution)
  - creates a custom Linux distribution just for you
    - (cross)-toolchain
    - package management
    - reproduceable builds
    - keeps servers busy

Figure: (Yocto) developer

Cross-Toolchain + Autotooled lib

Listing 1: installed stuff - dir: 67_cross-toolchain_library-autotooled-lib

```
 1 pushd /opt/sdk-target-rootfs/poky/2.5/multi-v7-ml/imx6q-phytec-mira-rdk-nand/core-image-↩
   sato-sdk-multi-v7-ml
 2 
 3 ==== libs: ====
 4 ls -lah usr/lib/libhw.a usr/lib/libhw.la usr/lib/libhw.so usr/lib/libhw.so.1 usr/lib/↩
   libhw.so.1.0.2
 5 -rw-r--r-- 1 root root 9.7K Aug  2 19:10 usr/lib/libhw.a
 6 -rwxr-xr-x 1 root root  896 Aug  2 19:10 usr/lib/libhw.la
 7 lrwxrwxrwx 1 root root   14 Aug  2 19:10 usr/lib/libhw.so -> libhw.so.1.0.2
 8 lrwxrwxrwx 1 root root   14 Aug  2 19:10 usr/lib/libhw.so.1 -> libhw.so.1.0.2
 9 -rwxr-xr-x 1 root root  13K Aug  2 19:10 usr/lib/libhw.so.1.0.2
10 
11 ==== .h: ====
12 ls -lah --color usr/local/include/lib_hw.h
13 -rw-r--r-- 1 root root 1.9K Aug  2 19:10 usr/local/include/libhw.h
14 
15 ==== readelf find SONAME: ====
16 arm-poky-linux-gnueabi-readelf -a usr/lib/libhw.so | grep SONAME
17  0x0000000e (SONAME)                     Library soname: [libhw.so.1]
18 popd
```

Line 5: static library `libname.a`

Line 6: libtool archive `libname.la`

Line 7: linker name `libname.so`

Line 8: soname `libname.so.maj`

Line 9: real name `libname.so.maj.min`

Line 13: public header `.h`

Line 17: linker name contains soname and is symlink to real name

What should go where?

Name	Example	Description	Rootfs	SDK	pkg
real name	`libhw.so.1.0.2`	obj. file with library code	✓	✓	`libhw-so`
soname	`libhw.so.1`	symlink `libhw.so.1 -> libhw.so.1.0.2`	✓	✓	`libhw-so`
linker name	`libhw.so`	contains `libhw.so.1` symlink `libhw.so -> libhw.so.1.0.2`		✓	`libhw-so-dev`
public header	`libhw.h`	public header file		✓	`libhw-so-dev`

Table: shared library properties

`FILES_${PN}` - see `meta/conf/bitbake.conf`
`FILES_${PN}-dev` - see `meta/conf/bitbake.conf`

Name	Example	Description	Rootfs	SDK	pkg
static library	`libhw.a`	obj. file with (static. linkable) library code	statically linked	✓	`libhw-a-staticdev`
public header	`libhw.h`	public header file		✓	`libhw-a-dev`

Table: static library properties

`FILES_${PN}-staticdev` - see `meta/conf/bitbake.conf`
`FILES_${PN}-dev` - see `meta/conf/bitbake.conf`

Table of Contents - Section

Autotooled lib - recipe

Listing 2: `libhw-so_1.0.2.bb` - dir: 69.01_autotooled-lib-so

```
 1 DESCRIPTION = "Autotooled lib"
 2 SECTION = "examples"
 3 LICENSE = "GPLv3"
 4 LIC_FILES_CHKSUM = "file://COPYING;md5=d32239bcb673463ab874e80d47fae504"
 5
 6 DEPENDS = " virtual/gettext"
 7
 8 SRC_URI = "file://${PN}-${PV}.tar.gz"
 9
10 inherit autotools
```

Line 1: `DESCRIPTION` should give an extended (possibly multi-line) description of what recipe provides.

by default, `DESCRIPTION` is set to the value of `SUMMARY` - use for short descriptions

Line 2: e.g `SECTION` = `"base"`, `SECTION_${PN}-doc = "doc"`, `SECTION_${PN}-dev = "devel"`

Line 3: `LICENSE` The list of source licenses for the recipe

Line 4: `LIC_FILES_CHKSUM` Checksums of the license text in the recipe source code.

Line 6: `DEPENDS` build time dependency

Line 8: `SRC_URI` list of source files

Line 10: `autotools* classes` support Autotooled packages.

.so and .a?

Listing 3: build/inspect - dir: 69.01_autotooled-lib-so

```
 1 cd /tmp/yocto-autobuilder/yocto-autobuilder/yocto-worker/res-custom-sumo-multi-v7-core-↩
   image-minimal-libs
 2 source oe-init-build-env
 3 
 4 export RECIPE_NAME="libhw-so"
 5 
 6 # do we have the recipe somewhere?
 7 bitbake -s | grep ${RECIPE_NAME}
 8 
 9 # build it
10 bitbake -c cleansstate ${RECIPE_NAME}
11 bitbake ${RECIPE_NAME}
12 
13 # grep a bit for various variables
14 bitbake ${RECIPE_NAME} -e | grep ^PACKAGES=
15 
16 bitbake ${RECIPE_NAME} -e | grep ^FILES_${RECIPE_NAME}=
17 bitbake ${RECIPE_NAME} -e | grep ^DEPENDS_${RECIPE_NAME}=
18 bitbake ${RECIPE_NAME} -e | grep ^RDEPENDS_${RECIPE_NAME}=
```

Line 14: PACKAGES list of packages the recipe creates

Line 16: FILES list of files and directories that are placed in a package

Line 17: DEPENDS build time dependency

Line 18: RDEPENDS runtime dependency.

.so and .a?

Listing 4: build/inspect - dir: 69.01_autotooled-lib-so

```
bitbake ${RECIPE_NAME} -e | grep ^FILES_${RECIPE_NAME}-dev=
bitbake ${RECIPE_NAME} -e | grep ^DEPENDS_${RECIPE_NAME}-dev=
bitbake ${RECIPE_NAME} -e | grep ^RDEPENDS_${RECIPE_NAME}-dev=

bitbake ${RECIPE_NAME} -e | grep ^FILES_${RECIPE_NAME}-dbg=
bitbake ${RECIPE_NAME} -e | grep ^DEPENDS_${RECIPE_NAME}-dbg=
bitbake ${RECIPE_NAME} -e | grep ^RDEPENDS_${RECIPE_NAME}-dbg=

bitbake ${RECIPE_NAME} -e | grep ^FILES_${RECIPE_NAME}-staticdev=
bitbake ${RECIPE_NAME} -e | grep ^DEPENDS_${RECIPE_NAME}-staticdev=
bitbake ${RECIPE_NAME} -e | grep ^RDEPENDS_${RECIPE_NAME}-staticdev=

bitbake ${RECIPE_NAME} -e | grep ^FILES_${RECIPE_NAME}-doc=
bitbake ${RECIPE_NAME} -e | grep ^DEPENDS_${RECIPE_NAME}-doc=
bitbake ${RECIPE_NAME} -e | grep ^RDEPENDS_${RECIPE_NAME}-doc=

bitbake ${RECIPE_NAME} -e | grep ^FILES_${RECIPE_NAME}-locale=
bitbake ${RECIPE_NAME} -e | grep ^DEPENDS_${RECIPE_NAME}-locale=
bitbake ${RECIPE_NAME} -e | grep ^RDEPENDS_${RECIPE_NAME}-locale=

# do we have a library somewhere in our packages
oe-pkgdata-util list-pkg-files -p ${RECIPE_NAME}
```

.so and .a?

Listing 5: output of interest - dir: 69.01_autotooled-lib-so

```
libhw-so                                              :1.0.2-r0
PACKAGES="libhw-so-dbg libhw-so-staticdev libhw-so-dev libhw-so-doc libhw-so-locale ↩
libhw-so"
FILES_libhw-so="/usr/bin/* /usr/sbin/* /usr/libexec/* /usr/lib/lib*.so.* /etc /com /var ↩
/bin/* /sbin/* /lib/*.so.* /lib/udev /usr/lib/udev /lib/udev /usr/lib/udev /usr/share/↩
libhw-so /usr/lib/libhw-so/* /usr/share/pixmaps /usr/share/applications /usr/share/idl /↩
usr/share/omf /usr/share/sounds /usr/lib/bonobo/servers"
FILES_libhw-so-dev="/usr/include /lib/lib*.so /usr/lib/lib*.so /usr/lib/*.la /usr/lib/*.↩
o /usr/lib/pkgconfig /usr/share/pkgconfig /usr/share/aclocal /lib/*.o /usr/lib/libhw-so↩
/*.la /lib/*.la"
RDEPENDS_libhw-so-dev="libhw-so (= 1.0.2-r0)"
FILES_libhw-so-staticdev="/usr/lib/*.a /lib/*.a /usr/lib/libhw-so/*.a"
RDEPENDS_libhw-so-staticdev="libhw-so-dev (= 1.0.2-r0)"

libhw-so:
        /usr/lib/libhw.so.1
        /usr/lib/libhw.so.1.0.2
libhw-so-dev:
        /usr/include/lib_hw.h
        /usr/lib/libhw.so
```

.so and .a?

Listing 6: build/inspect rootfs - dir: 69.01_autotooled-lib-so

```
cd /tmp/yocto-autobuilder/yocto-autobuilder/yocto-worker/res-custom-sumo-multi-v7-core-↩
image-minimal-libs
source oe-init-build-env

# add to conf/local.conf:
# IMAGE_INSTALL_append = " libhw-so"

bitbake core-image-minimal

# searching for rootfs
R=$(bitbake -e ${IMAGE} | grep ^IMAGE_ROOTFS=)
echo ${R}
ROOTFS=$(echo ${R} | sed "s/^IMAGE_ROOTFS=\"//" | sed "s/\"$//")
echo ${ROOTFS}

# searching for our lib in rootfs
pushd ${ROOTFS}
find | grep libhw
find | grep lib_hw
popd
```

Listing 7: output of interest - dir: 69.01_autotooled-lib-so

```
./usr/lib/libhw.so.1.0.2
./usr/lib/libhw.so.1
```

.so and .a?

Listing 8: build/inspect SDK - dir: 69.01_autotooled-lib-so

```
cd /tmp/yocto-autobuilder/yocto-autobuilder/yocto-worker/res-custom-sumo-multi-v7-core-↩
image-minimal-libs
source oe-init-build-env

bitbake core-image-minimal -c populate_sdk

# searching for SDK dir
SDKOUT=$(bitbake -e ${IMAGE} | grep ^SDK_OUTPUT=)
echo ${SDKOUT}
SDK_OUTPUT=$(echo ${SDKOUT} | sed "s/^SDK_OUTPUT=\"//" | sed "s/\"$//")
echo ${SDK_OUTPUT}

# searching for our lib in SDK
pushd ${SDK_OUTPUT}
find | grep libhw
find | grep lib_hw
popd
```

Listing 9: output of interest - dir: 69.01_autotooled-lib-so

```
./usr/lib/libhw.so
./usr/lib/libhw.so.1
./usr/lib/libhw.so.1.0.2

./usr/include/lib_hw.h
```

What goes where?

Name	Example	Description	Rootfs	SDK	pkg
real name	`libhw.so.1.0.2`	obj. file with library code	✓	✓	`libhw-so`
soname	`libhw.so.1`	symlink `libhw.so.1 -> libhw.so.1.0.2`	✓	✓	`libhw-so`
linker name	`libhw.so`	contains `libhw.so.1` symlink `libhw.so -> libhw.so.1.0.2`		✓	`libhw-so-dev`
public header	`libhw.h`	public header file		✓	`libhw-so-dev`

Table: shared library properties

.a?

Listing 10: Peter Paul & Mary: Where Have All The Flowers Gone slightly modified

```
Where have all the static libs gone,
long time passing?
Where have all the static libs gone, long time ago?
Where have all the static libs gone?
Young girls have picked them everyone.
Oh, when will they ever learn?
Oh, when will they ever learn?
```

- Including Static Library Files (MM)

Table of Contents - Section

.a not built

- meta/conf/distro/include/no-static-libs.inc

```
DISABLE_STATIC ?= " --disable-static"
EXTRA_OECONF_append[1] = "${DISABLE_STATIC}"
```

[1] Appending and Prepending (Override Style Syntax) (BB)

Autotooled lib - recipe

Listing 11: libhw-a_1.0.2.bb - dir: 69.02.01_autotooled-lib-a

```
DESCRIPTION = "Autotooled lib"
SECTION = "examples"
LICENSE = "GPLv3"
LIC_FILES_CHKSUM = "file://COPYING;md5=d32239bcb673463ab874e80d47fae504"

DEPENDS = " virtual/gettext"

SRC_URI = "file://${PN}-${PV}.tar.gz"

# By default EXTRA_OECONF is set to --disable-static ...
EXTRA_OECONF_remove = "--disable-static"

# enable static and disable shared (for fun)
EXTRA_OECONF += "--enable-static --disable-shared"

# allow empty (main) package
ALLOW_EMPTY_${PN} = "1"

inherit autotools
```

Autotooled lib - recipe

Listing 12: output of interest - dir: 69.02.01_autotooled-lib-a

```
oe-pkgdata-util list-pkg-files -p libhw-a

libhw-a-dev:
        /usr/include/lib_hw.h
libhw-a-staticdev:
        /usr/lib/libhw.a
```

nothing in SDK - solution 1

- not surprisingly, since we did not add `libhw-a` to SDK
- add to top level config file e.g. `local.conf`

```
IMAGE_INSTALL_append[1] = " libhw-a"[2]
```

- add to recipe `ALLOW_EMPTY_${PN} = "1"`[2]
- add to top level config file e.g. `local.conf`

```
TOOLCHAIN_TARGET_TASK_append = " libhw-a-staticdev"[3]
```

Listing 13: output of interest - dir: 69.02.01_autotooled-lib-a

```
./opt/poky/2.5/sysroots/armv7a-neon-poky-linux-gnueabi/usr/lib/libhw.a
./opt/poky/2.5/sysroots/armv7a-neon-poky-linux-gnueabi/usr/include/lib_hw.h
```

[1] Appending and Prepending (Override Style Syntax) (BB)

[2] for `libhw-a-dev` i.e. `lib_hw.h` to end up in SDK

[3] for `libhw-a-staticdev` i.e. `libhw.a` to end up in SDK

nothing in SDK - solution 2

- add to top level config file e.g. `local.conf`

  `IMAGE_INSTALL_append`[1] `= " libhw-a"`[2]

- add to recipe `ALLOW_EMPTY_${PN} = "1"`[2]
- add to top level config file e.g. `local.conf`

  `SDKIMAGE_FEATURES_append`[1] `= " staticdev-pkgs"`[3]

Listing 14: output of interest - dir: 69.02.02_autotooled-lib-a

```
./opt/poky/2.5/sysroots/armv7a-neon-poky-linux-gnueabi/usr/lib/libhw.a
./opt/poky/2.5/sysroots/armv7a-neon-poky-linux-gnueabi/usr/include/lib_hw.h
```

[1] Appending and Prepending (Override Style Syntax) (BB)

[2] for `libhw-a-dev` i.e. `lib_hw.h` to end up in SDK

[3] for `libhw-a-staticdev` i.e. `libhw.a` to end up in SDK

Table of Contents - Section

Thank you!

ЯELIABLEEMBEDDEDSYSTEMS Consulting Training Engineering

Robert Berger

Embedded Software Specialist

email:

robert.berger@ReliableEmbeddedSystems.com

phone:

+43 (0) 699 17 69 07 19

web:

http://ReliableEmbeddedSystems.com

Building world class world wide win/win cooperations by helping you to create better embedded software!