How do I link librt and libpthread to a new layer in Yocto?

Asked 2 years ago Modified 1 year, 11 months ago Viewed 2k times

0

This question follows
[Where can I find documentation on aarch64-poky-linux-ld?](https://stackoverflow.com)

Please ignore sets of outer double-quotes. They seem necessary to get "StackOverflow" to accept my input.
My layer is "meta-oca-so", its package is "oca" and its recipe is oca_1.2.7.bb.
I revised my "oca makefile" to fix the previous and some linker errors to do with invalid link flags, and I'm now getting the full 29 "oca" .a libs to link together when I do "bitbake oca". I didn't have to do any clean while changing the link flags and redoing "bitbake oca".
Now when I do "bitbake oca", the error messages are:

```
~/Yocto/imx-yocto-bsp/build-wayland/tmp/work/aarch64-poky-linux/oca/1.2.7-r0/oca-1.2.7/Obj/linuxApp/Release/OcaProto
aarch64-poky-linux-ld: cannot find -lstdc++
aarch64-poky-linux-ld: cannot find -lrt
aarch64-poky-linux-ld: cannot find -lpthread
aarch64-poky-linux-ld: cannot find -ldns_sd
```

Focusing on "librt" and "libpthread" in this question, I added these lines to my "local.conf":

```
IMAGE_INSTALL_append=" librt"
TOOLCHAIN_TARGET_TASK_append=" librt"
IMAGE_INSTALL_append=" libpthread"
TOOLCHAIN_TARGET_TASK_append=" libpthread"
```

But it didn't clear its two link errors when I redid "bitbake oca".

When I tried the overall "bitbake imx-image-multimedia" I got these messages about "librt":

```
NOTE: Resolving any missing task queue dependencies
ERROR: Nothing RPROVIDES 'librt' (but /home/james/Yocto/imx-yocto-bsp/sources/meta-imx/meta-sdk/recipes-fsl/images/imx-image-multimedia.bb RDEPENDS on or otherwise requires it)
NOTE: Runtime target 'librt' is unbuildable, removing...
Missing or unbuildable dependency chain was: ['librt']
ERROR: Required build target 'imx-image-multimedia' has no buildable providers.
Missing or unbuildable dependency chain was: ['imx-image-multimedia', 'librt']
```

It's hard to find much info about "rt" because this two-letter combination is not statistically improbable.

Similarly, when I commented-out the two "librt" lines from "local.conf" and retried "bitbake imx-image-multimedia", I got these messages about "libpthread":

```
Missing or unbuildable dependency chain was: ['libpthread']
ERROR: Required build target 'imx-image-multimedia' has no buildable providers.
Missing or unbuildable dependency chain was: ['imx-image-multimedia', 'libpthread']
```

How do I obtain "librt" and "libpthread"?
Do I need to add a layer each to build them? Will there be a "dependency hell" tree of supporting packages needed for each of them?

Here's the relevant makefile component, makeOCA.inc. There are 63 "make" files of different descriptions and at different levels in the OCA project. It's pretty long, at ~1070 lines. I snipped the sections for non-Linux systems; I'm only concerned with linuxRelease here.

```
#  Project              : OCA
#  Module               : Multiple components
#  Description          : Include file for C / CPP makefiles.
#

################################################################################
# Environment variable checking
################################################################################
#
# If one of the variables below is not set,
# calling make will result in a "missing separator" error.
#
ifeq ($(CAP_HOME),)
  variable CAP_HOME is not set
else
  CAP_HOME := $(patsubst %/,%,$(subst \,/,$(CAP_HOME)))
endif

################################################################################
# Variables
################################################################################
ifeq ($(NAME_MKE),)
  NAME_MKE := makefileOCA
endif
ifeq ($(IGNORE_MAKE_ERRORS),Y)
  PREFIX := -@
else
  ifneq ($(DISPLAY_MAKE_CMDS),Y)
    PREFIX := @
  endif
endif
EMPTY :=
SPACE := $(EMPTY) $(EMPTY)
#
# Only set variables when the makefile is called with an actual target
# (not a phony one).
#
ifneq ($(PLATFORM),)
```

An important realization is where the documentation is about the path variables and other settings for the CFLAGS and LDFLAGS: with all variables in the Yocto mega-manual, in "Chapter 33. Variables Glossary", https://www.yoctoproject.org/docs/3.0/mega-manual/mega-manual.html#ref-variables-glos

linux yocto

Share Improve this question Follow

edited Apr 26, 2021 at 17:25

asked Mar 30, 2021 at 15:26

microajim
17 9

```
        $(SRC_HOME)/inc;\
```

IMAGE_INSTALL installs package in final image, this is not what you want, you want to add a build dependency (with DEPENDS += ""). That's being said, pthread and librt are probably already included by default in the Yocto toolchain. My feeling is it's more a problem of Makefile, can you show it (especially linker rule) ? – hilt0n Mar 31, 2021 at 7:48

@hilt0n: The project builds fine natively in Linux, but I had to hack up makeOCA.inc to get it compiling (not including linking) in Yocto. I included the key parts of it above. Can you give me the exact right syntax for 'DEPENDS += "" '? In the meantime I'll try looking it up. – microajim Mar 31, 2021 at 20:37

1 Answer

Sorted by: Highest score (default)

```
    # Work around to make sure only a single backslash is set
```

Yocto dependency concepts

2

First, you seem to mix various dependency concepts in Yocto:

- `IMAGE_INSTALL` is an image dependency telling to Yocto the recipe in parameter must be installed in the final image. It should be placed inside image recipe and not inside package recipe. I don't think adding that into a package recipe tells to Yocto to add this dependencies to the recipe's staging directory.
- `RDEPENDS` is used inside package recipe and tells to Yocto the recipe depends on the packages in parameter at runtime. Then it will install in final image the required runtime dependencies if you install the package requiring them.
  A basic example is a recipe containing a script shell depends at runtime on some shell interpreter.
- `DEPENDS` is used also inside package recipe and tells to Yocto the recipe depends on the packages in parameters at compile time. It will add the header files and the libraries in the staging directory of your recipe.

That being said, `libstdc++`, `libpthread` and `librt` are part of the toolchain and you don't need to add explicit dependencies in your recipes for that. They will be installed in the staging directory or your recipe without any effort.

Regarding the `libdns_sd`, following my quick research, this library is provided by mDNS. You then need to add a dependency on it:

```
DEPENDS = "mdns"
```

As explained above, this will add the header files, *.a and *.so in the staging directory of your recipe.

What is missing ?

Once the dependencies are correctly configured, your compiler needs to know where are the include and library directories and this is clearly missing in your Makefile probably because you assume the dependencies are part of the host toolchain. When you build directly for the host, the compiler knows where to find includes (search in /usr/include for example) and libraries which makes this process quite simple.

However, as for a standard cross compilation project, in Yocto the headers and libraries are pass to the compiler/linker when called from your recipe.

Since you don't give the Yocto recipe, I will give an example on how it can be done and some suggestions. Also, I focus on compile aspect, the example is not complete and should probably be adapted for other build steps (like install).

```
DEPENDS += "my-extralib"

# Change eventually the default source directory
S = "${WORKDIR}/git"

EXTRA_OEMAKE = " 'CC=${CC}' \
    'CFLAGS=${CFLAGS} ${TARGET_CC_ARCH} -I${S}/inc
-I${STAGING_DIR_TARGET}${includedir}/my-extralib-include' \
    'LDFLAGS=-lrt -lpthread -lm -lmy-extralib -L${STAGING_DIR_TARGET}${libdir}/my-
extralib-libdir' \
    'BUILDDIR=${S}' \
    'OTHER_DEFINE=${OTHER_YOCTO_VAR}' \
"
```

Using EXTRA_OEMAKE lets Yocto do a lot of tasks automatically for you like the configure, compile and install. If you want to override them, this is also possible:

```
do_compile() {
    # modify the Makefile on the fly as an example
    cp ${S}/Makefile.orig ${S}/Makefile
    sed -i -e 's:= $(LDFLAGS):?= $(LDFLAGS):' ${S}/Makefile
    oe_runmake SOME_OVERRIDE='some-override'
}
```

You can also disable a step like the configure which is probably not used in your case:

```
do_configure[noexec] = "1"
```

Some tips

- If you start with Yocto, there is a lot to learn and it can be difficult to find how to manage all the possibilities and options at the beginning. I strongly suggest you start with a simple Makefile and minimum source files to keep focus on the Yocto side.
- You can find a lot of recipe examples on the web (also Yocto, Poky base recipes) which are probably close to you are trying to achieve. The documentation of Yocto is really

creating your recipe.

Share Improve this answer Follow

answered Apr 2, 2021 at 12:29

hilt0n

356 1 10

```
# That's for Linaro and the Snapdragon410 (Qualcomm APQ8016) processor; next is for
```

Thanks for your answer of Apr 2, 2021 at 12:29. My OCA recipe is given three links back, at "How do I get a complex non-Yocto makefile-based project to cross-compile in a Yocto layer?": stackoverflow.com/questions/66680355/… – microajim Apr 5, 2021 at 20:57

Is CFLAGS a Yocto variable that needs to be added so that my compiler can find the include and library directories? Does it stand for something like "Configure flags"? You put -lrt -lpthread just like I did (I think I also need -lstdc++) so the CFLAGS setting on the line before looks like the setting. I'll try looking up about it. I'll try the ' DEPENDS = "mdns" '. – microajim Apr 5, 2021 at 20:57

Can you tell me where the best place is to find documentation on IMAGE_INSTALL, RDEPENDS and DEPENDS? – microajim Apr 6, 2021 at 19:32

I tried adding ' DEPENDS = "mdns" ' in my meta-oca-so/conf/layer.conf file, as the last line, but it didn't have any effect, even after doing 'bitbake -c cleansstate oca'. When I do 'bitbake oca', the line 'aarch64-poky-linux-ld: cannot find -ldns_sd' remains in my error output along with the other three. Is "mdns" the right thing to put in the DEPENDS statement? What do I refer to in my mdns package to get the right thing to put there? I'll try adding using the CFLAGS next. – microajim Apr 7, 2021 at 16:21

During compilation, you can specify to compiler (in the Makefile who call the compiler/linker) which path contains the include files with -I (i uppercase) option. During linking, you can specify to linker which library to include with -l (l lowercase) and which path where to search the libraries with -L. Your makefile already specifies which lib to include but it doesn't specify where to look for these libraries. You need to update your Makefile(s) to accept these information from the Yocto's recipe. – hilt0n Apr 8, 2021 at 9:55

```
            winntDebug64 winntRelease64
```

```
        _OLDNAME_,$(@F).tmp,$(subst _NEWNAME_,$(@F),$(RN))))
        $(LINUXSYSTEMLIBS)
```