Rockchip Linux SDK 编译 ROS2 说明

文件标识：RK-SM-YF-912

发布版本：V1.1.1

日期：2024-01-11

文件密级：□绝密 □秘密 □内部资料 ■公开

免责声明

本文档按"现状"提供，瑞芯微电子股份有限公司（"本公司"，下同）不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因，本文档将可能在未经任何通知的情况下，不定期进行更新或修改。

商标声明

"Rockchip"、"瑞芯微"、"瑞芯"均为本公司的注册商标，归本公司所有。

本文档可能提及的其他所有注册商标或商标，由其各自拥有者所有。

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

概述

部分RK Linux SDK 集成了ROS1的编译，但ROS1不在本文讨论范围内， 具体可以检查buildroot/package/rockchip/ros路径及docs/下的文档。

针对ROS2，RK Linux SDK仅提供ROS2所需要的依赖包，ROS2放到Docker中去编译。将ROS2的编译与RK Linux SDK最大程度解绑，方便ROS2版本更新维护。

ROS2文档[1] [2]显示，它提供了Docker镜像用以交叉编译arm/aarch64，比如Dockerfile_ubuntu_arm64_prebuilt[2]。但目标系统仅支持ubuntu 18.04(bionic)，与RK Linux SDK中的rootfs相差较多。借鉴该思路，我们编译ROS2总结为三个步骤：

- 首先完成RK Linux SDK 编译，它包含了一些ROS2所需要的应用包，比如python3、bullet、opencv、eigen等
- 进入Docker，利用Ubuntu的环境，以及RK Linux SDK的交叉编译工具链、Sysroot编译ROS2
- 退出Docker，重新打包RK Linux SDK Rootfs

当前已经验证可编译通过的有foxy, galactic, humble, iron四个ROS2发行版。

产品版本

芯片名称	内核版本
RK356x，RK3588等arm64位芯片	与内核版本无关

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V1.0.0	zhengsq	2021-09-09	初始版本
V1.1.0	zhengsq	2023-09-21	更新匹配当前新的Linux SDK， 增加ROS2 humble及iron支持， 修改编译Docker环境，调整文档结构
V1.1.1	zhengsq	2021-01-11	部分语句修正

目录

1. ROS2的具体版本

Rockchip Linux SDK基于Buildroot系统构建， 并持续在更新升级工具链、软件包到较新的版本。 因此最新的SDK在编译ROS2时，可能会遇到一些小的错误。 我们所提供的编译方法中，尽量将版本固定：

- 固定版本的ROS2源码， 从ROS2 githup下载，明确各个软件包的版本号

各ROS2发行版本号如下：

ROS2 发行版本	版本号	链接
foxy	ros2-release-foxy-20230620	https://github.com/ros2/ros2/releases/tag/release-foxy-20230620
galactic	ros2-release-galactic-20221209	https://github.com/ros2/ros2/releases/tag/release-galactic-20221209
humble	ros2-release-humble-20230724	https://github.com/ros2/ros2/releases/tag/release-humble-20230724
iron	ros2-release-iron-20230912	https://github.com/ros2/ros2/releases/tag/release-iron-20230912

已经编译通过的RK Linux SDK版本：

Linux SDK Buildroot Version	Python版本	匹配的Docker镜像	备注
linux-5.10-stan-rkr1	Python 3.10.5	jammy-ros2-build	需要加ros2_dep.config补丁
linux-4.19-gen-rkr3	Python 3.8.6	focal-ros2-build	需要更新并使能lttng相关补丁包：lttng-tools(2.12.3)、lttng-libust(2.12.3)、liburcu(0.13.0)需要更新

- 注： Docker host与Target rootfs中Python版本需要保持一致
- 32位系统未验证

2. 使用Docker编译所需的文件及说明

本节中所述的补丁、Docker 镜像、源码， 可从下述连接下载：

```
https://console.zbox.filez.com/l/iJBMWZ
```

```
tree
```

```
.
├── docker-focal-python38
│   └── rosdep.Dockerfile      # 制作Docker Image的Dockerfile
├── docker-jammy-python310
│   └── rosdep.Dockerfile      # 制作Docker Image的Dockerfile
├── focal-ros2-build.tar.gz    # 根据Dockerfile制作好的Docker Image
├── jammy-ros2-build.tar.gz    # 根据Dockerfile制作好的Docker Image
├── linux-sdk-patches
│   └── buildroot              # RK Linux SDK不同发布版本有可能会缺少的补丁
│       ├── 0001-package-add-libasio.patch
│       └── 0002-configs-rockchip-add-ros2-build-dependencies.patch
├── MD5SUM.txt                 # 各压缩包的MD5SUM检验码
├── ros2-build-scripts.tar.gz  # 编译脚本及补丁
└── ros2-sources.tar.gz        # ROS2及其部分依赖库的源码包
```

在RK Linux SDK的Buildroot目录中，检查是否存在ros2_dep.config文件，

```
ls buildroot/configs/rockchip/ros2_dep.config
buildroot/configs/rockchip/ros2_dep.config

# 如该ros2_dep.config中缺少：LTTNG_TOOLS，手动加上（ROS2 iron有依赖）
tail -f buildroot/configs/rockchip/ros2_dep.config
# Required by ros2-iron tracetools; With LTTNG foxy/galactic/humble will build
tracetools too.
BR2_PACKAGE_LTTNG_TOOLS=y
BR2_PACKAGE_LTTNG_LIBUST=y
```

- 如不存在该文件，则需要在Buildroot目录中打上如下2个补丁

```
  0001-package-add-libasio.patch
  0002-configs-rockchip-add-ros2-build-dependencies.patch
```

- 检查lttng-tools(2.12.3)、lttng-libust(2.12.3)、liburcu(0.13.0)是否满足版本要求

3. 编译ROS2的依赖包

RK Linux SDK 中的Buildroot工程里，ros2_dep.config提供了编译、运行ROS2所需要的依赖包，需要添加并编译到rootfs。 例如，将ros2_dep.config添加到rockchip_rk356x_robot_defconfig中：

```
git diff
--- a/configs/rockchip_rk356x_robot_defconfig
+++ b/configs/rockchip_rk356x_robot_defconfig
@@ -10,6 +10,7 @@
 #include "wifi.config"
 #include "debug.config"
 #include "bt.config"
+#include "ros2_dep.config"
 BR2_TARGET_GENERIC_HOSTNAME="rk356x_robot"
 BR2_TARGET_GENERIC_ISSUE="Welcome to RK356X Buildroot For Robot"
 BR2_ROOTFS_OVERLAY:="board/rockchip/common/robot/base
board/rockchip/common/wifi"
```

完整编译rootfs后，进入下一步。

4. 准备Linux(Ubuntu) PC机的编译环境

Ubuntu PC机上安装docker程序：

```
sudo apt install docker.io
sudo usermod -aG docker $USER
newgrp docker  # 登录到docker用户组
```

4.1 导入Docker Image

首先检查RK Linux SDK 编译出来的Python版本，例如：

```
./buildroot/output/rockchip_rk3562_robot/host/bin/python --version
Python 3.10.5
```

按Python版本号，匹配对应的Docker Image 镜像：

Python版本	匹配的Docker镜像
Python 3.10.5	jammy-ros2-build
Python 3.8.6	focal-ros2-build

选择jammy-ros2-build，导入并进入到Docker Container：

```
gunzip jammy-ros2-build.tar.gz
docker image load -i jammy-ros2-build.tar
docker run -it --mount type=bind,source=/home/zsq/29/linux-
sdk/buildroot/output/rockchip_rk3562_robot/,target=/buildroot jammy-ros2-build
```

- 其中source=需要修改成相应的Linux SDK 编译的output目录的绝对路径
- 进入Container后， 默认用户是builder，密码默认是: rockchip

4.2 拷贝编译脚本及源码包

通过docker container cp命令，拷贝所需文件：

```
# 首先查找已登录的container ID
docker container ls
CONTAINER ID   IMAGE              COMMAND       CREATED          STATUS
PORTS     NAMES
c519d9d668f9   jammy-ros2-build   "/bin/bash"   15 minutes ago   Up 15 minutes
          pedantic_feynman

docker container cp ros2-sources.tar.gz c519d9d668f9:/tmp/
docker container cp ros2-sources.tar.gz c519d9d668f9:/tmp/
```

在Container 中，将其解压：

```
builder@c519d9d668f9:/opt/ros$ ls /tmp/
ros2-build-scripts.tar.gz  ros2-sources.tar.gz

builder@c519d9d668f9:/opt/ros$ tar zxf /tmp/ros2-build-scripts.tar.gz -C /
builder@c519d9d668f9:/opt/ros$ tar zxf /tmp/ros2-sources.tar.gz -C /

builder@c519d9d668f9:/opt/ros$ ls
cross-compile  foxy  galactic  humble  iron
```

4.3 修改脚本中的Python版本号

检查/opt/ros/cross-compile/cross-compile.mixin及build_ros2.sh， 将其中的Python版本号修改成RK Linux SDK对应的版本号，例如：

- 310修改成38，其中310表示Python3.10版本，以此类推
- 3.10修改成3.8

4.4 (可选)使用Docker编译ROS2

如果想要从头开始制作Docker Image，可使用RK提供的rosdep.Dockerfile：

```
docker build -t jammy-ros2-build -f rosdep.Dockerfile ./  # "./"不要少拷贝了，表示当
前目录
```

4.5 (可选)下载源码

如需要自己下载其它版本的源码，进入docker后，可使用vcs-import：

```
cd /opt/ros/foxy
mkdir src
vcs-import -w 10 --retry 10 --skip-existing --recursive src < ros2-release-foxy-20230620/ros2.repos
```

5. 编译ROS2

再次确认

- RK Linux SDK 已经有加上ros2_dep.config，并且rootfs完整编译通过
- 所选Docker Image与RK Linux SDK编译出来的Python是匹配的

选择所需ROS2版本， 并依次执行以下命令：

```
ls /opt/ros
cross-compile  foxy  galactic  humble  iron
cd /opt/ros/iron
./prepare-source.sh
./build-ros2.sh
# 编译成功后，应有类似提示：
...
Summary: 317 packages finished [15min 37s]
...
build ros quit & cleanup
```

说明：

- 编译生成的目标文件位于/buildroot/target/opt/ros目录
- 编译中间过程存放在/buildroot/build/ros目录

如build_ros2.sh未提示错误即成功编译。其中，还有部分包在Buildroot SDK环境中，无法编译、执行的，比如：

- rviz，依赖于X11/desktop。如果你需要这个功能，直接使用Ubuntu arm镜像，而不是Buildroot
- turtlesim，依赖于UI显示.
- 如果想要取消某个包的编译，在src对应的路径下，创建一个COLCON_IGNORE即可。比如`touch src/ros/ros_tutorials/turtlesim/COLCON_IGNORE`

5.1 TRY_RUN需要手动执行并记录结果

fastrtps TRY_RUN提示：

```
--- stderr: fastrtps
CMake Error: TRY_RUN() invoked in cross-compiling mode, please set the following
cache variables appropriately:
   SM_RUN_RESULT (advanced)
   SM_RUN_RESULT__TRYRUN_OUTPUT (advanced)
For details see /buildroot/build/ros/fastrtps/TryRunResults.cmake
```

- 需要按照说明，将应用程序放到板端执行，并按说明填写结果。例如：

```
  root@rk3562-buildroot:/# /tmp/cmTC_4f573-SM_RUN_RESULT
  PTHREAD_RWLOCK_PREFER_READER_NP

  # 根据上述执行结果，在docker中填入结果：
  cat /buildroot/build/ros/fastrtps/TryRunResults.cmake
  ....
  set( SM_RUN_RESULT
       "0"
       CACHE STRING "PTHREAD_RWLOCK_PREFER_READER_NP" FORCE)

  set( SM_RUN_RESULT__TRYRUN_OUTPUT
       "0"
       CACHE STRING "PTHREAD_RWLOCK_PREFER_READER_NP" FORCE)
```

rosbag2_cpp TRY_RUN提示：

```
--- stderr: rosbag2_cpp
CMake Error: TRY_RUN() invoked in cross-compiling mode, please set the following
cache variables appropriately:
   HAVE_SANITIZERS_EXITCODE (advanced)
   HAVE_SANITIZERS_EXITCODE__TRYRUN_OUTPUT (advanced)
For details see /buildroot/build/ros/rosbag2_cpp/TryRunResults.cmake
```

- 同上， 需要按照说明，将应用程序放到板端执行，并按说明填写结果。例如：

```
  set( HAVE_SANITIZERS_EXITCODE
       "127"
       CACHE STRING "error while loading shared libraries: liblsan.so.0: cannot
open shared object file: No such file or directory" FORCE)

  set( HAVE_SANITIZERS_EXITCODE__TRYRUN_OUTPUT
       "127"
       CACHE STRING "error while loading shared libraries: liblsan.so.0: cannot
open shared object file: No such file or directory" FORCE)
```

5.2 单独编译某个ROS2 package 及应用程序

使用colcon build的参数`--packages-select <pacage_name>`可单独编译包，可参考`colcon build --help`。

6. 打包rootfs并运行ROS2

在上述ROS2完整编译结束后，进入到buildroot sdk，重新打包rootfs即可。ROS2安装在/opt/ros目录下。

```
cd /data/linux-sdk/rk3562
./build.sh rootfs   # 重新打包rootfs.img
```

烧录rootfs.img后，进入rk3562板端，执行Hello World Demo：

```
# cd /opt/ros/
# export COLCON_CURRENT_PREFIX=/opt/ros
# export ROS_HOME=/userdata/
# source ./local_setup.sh
# ros2 pkg list
# ros2 pkg executables

# ros2 run demo_nodes_cpp listener &
# ros2 run demo_nodes_cpp talker
[INFO] [1501839280.834017748] [talker]: Publishing: 'Hello World: 1'
[INFO] [1501839280.839280957] [listener]: I heard: [Hello World: 1]
[INFO] [1501839281.831636015] [talker]: Publishing: 'Hello World: 2'
[INFO] [1501839281.835092640] [listener]: I heard: [Hello World: 2]
[INFO] [1501839282.831618532] [talker]: Publishing: 'Hello World: 3'
[INFO] [1501839282.835336782] [listener]: I heard: [Hello World: 3]

# ros2 run demo_nodes_py listener &
# ros2 run demo_nodes_py talker
```

7. 已经解决的常见编译报错

7.1 编译主机内存不足

可以开启交换空间，例如zram:

```
sudo -i su
# modprobe zram
# echo 12G > /sys/block/zram0/disksize
# echo 6G > /sys/block/zram0/mem_limit
# mkswap /dev/zram0
# swapon  /dev/zram0
# free -h
               total        used        free      shared  buff/cache   available
Mem:            14Gi       3.9Gi       5.5Gi        27Mi       5.4Gi        10Gi
Swap:           11Gi       2.7Gi       9.3Gi
```

7.2 在板端执行时报`GLIBCXX_3.4.30' not found

```
root@rk3562-buildroot:/opt/ros-foxy# ros2 run demo_nodes_cpp talker
/opt/ros-foxy/lib/demo_nodes_cpp/talker: /lib/libstdc++.so.6: version
`GLIBCXX_3.4.30' not found (required by /opt/ros-foxy/lib/librclcpp.so)
/opt/ros-foxy/lib/demo_nodes_cpp/talker: /lib/libstdc++.so.6: version
`GLIBCXX_3.4.30' not found (required by /opt/ros-foxy/lib/libspdlog.so.1)
```

因为RK Linux SDK版本较多， 工具链版本一直在更新，因此需要使用Linux SDK编译Rootfs的交叉工具来编译ROS2。

7.3 编译结果中出现x86_64动态库

```
ls /opt/ros/lib/python3.10/site-packages/rclpy/_rclpy_pybind11.cpython-310-
x86_64-linux-gnu.so
```

pybind11在交叉编译的环境中， 确实是会有一些已知的问题：

找到的python是HOST端的可执行文件，因此一系列参数也是根据HOST端生成，如：

```
PYTHON_MODULE_EXTENSION:INTERNAL=.cpython-310-x86_64-linux-gnu.so
```

在pybind11/tools/FindPythonLibsNew.cmake较新的代码中，建议若是Cross Compling，可在外部手动添加python的参数。基于此，

1. 修改src/ros2/pybind11_vendor中pybind11升级到v2.10.2
2. 并在pybind11_verdor/CMakeLists.txt中设置以下2个参数，指定具体的PYTHON_MODULE_EXTENSION：

```
    list(APPEND extra_cmake_args "-DPYBIND11_PYTHONLIBS_OVERWRITE=OFF")
    list(APPEND extra_cmake_args "-DPYTHON_MODULE_EXTENSION=.cpython-310-
aarch64-linux-gnu.so")
```

3. 在cross-compile.mixin中，也声明:

```
        - "-DPYBIND11_PYTHONLIBS_OVERWRITE=OFF"
        - "-DPYTHON_MODULE_EXTENSION=.cpython-310-aarch64-linux-gnu.so"
```

上述修改后， 仍然发现rclpy在编译时，其CMakeCache.txt中得到的PYTHON_MODULE_EXTENSION仍指向"x86_64"，但第二次再编译时，会被修改成预期的aarch64。这是因为：

1. 在pybind11/tools/pybind11NewTools.cmake中， 若未设置过PYBIND11_PYTHON_EXECUTABLE_LAST、或它值被修改了，会直接清空PYTHON_MODULE_EXTENSION

```
    76 if(NOT ${_Python}_EXECUTABLE STREQUAL PYBIND11_PYTHON_EXECUTABLE_LAST)
    77   # Detect changes to the Python version/binary in subsequent CMake runs,
and refresh config if needed
    78   unset(PYTHON_IS_DEBUG CACHE)
    79   unset(PYTHON_MODULE_EXTENSION CACHE)
    80   set(PYBIND11_PYTHON_EXECUTABLE_LAST
    81       "${${_Python}_EXECUTABLE}"
    82       CACHE INTERNAL "Python executable during the last CMake run")
    83 endif()
```

2. 在[pybind11 issue #236]也有类似的现象

3. 解决： 修改pybind11： 若是PYBIND11_PYTHONLIBS_OVERWRITE = "OFF"，则不重设上述参数：

```
   commit f7f1f2a927dd785d109833e411325de4c248719f (HEAD -> v2.10.2-fix)
   Author: cross-build for rk-linux-sdk <cross-build@rk-linux-sdk.com>
   Date:   Fri Sep 22 08:24:58 2023 +0000

       Do not override the PYTHON_MODULE_EXTENSION if cross building

       As suggested in tools/FindPythonLibsNew.cmake,
PYBIND11_PYTHONLIBS_OVERWRITE is a flag to indicate that we set python variables
manually when cross building.

       In this case, do not override variables if PYBIND11_PYTHON_EXECUTABLE_LAST
changed or is empty.

   diff --git a/tools/pybind11NewTools.cmake b/tools/pybind11NewTools.cmake
   index 7d7424a7..91980dad 100644
   --- a/tools/pybind11NewTools.cmake
   +++ b/tools/pybind11NewTools.cmake
   @@ -73,7 +73,7 @@ if(NOT DEFINED ${_Python}_EXECUTABLE)

    endif()

   -if(NOT ${_Python}_EXECUTABLE STREQUAL PYBIND11_PYTHON_EXECUTABLE_LAST)
   +if(NOT ${_Python}_EXECUTABLE STREQUAL PYBIND11_PYTHON_EXECUTABLE_LAST AND NOT
PYBIND11_PYTHONLIBS_OVERWRITE STREQUAL "OFF")
      # Detect changes to the Python version/binary in subsequent CMake runs, and
refresh config if needed
      unset(PYTHON_IS_DEBUG CACHE)
      unset(PYTHON_MODULE_EXTENSION CACHE)
```

7.4 google_benchmark工程缺少limits头文件

编译foxy时会报如下错误，

```
In file included from /buildroot/build/ros/google_benchmark_vendor/benchmark-
1.5.2-prefix/src/benchmark-1.5.2/src/benchmark_register.cc:15:
/buildroot/build/ros/google_benchmark_vendor/benchmark-1.5.2-
prefix/src/benchmark-1.5.2/src/benchmark_register.h: In function ‘typename
std::vector<T>::iterator benchmark::internal::AddPowers(std::vector<T>*, T, T,
int)’:
/buildroot/build/ros/google_benchmark_vendor/benchmark-1.5.2-
prefix/src/benchmark-1.5.2/src/benchmark_register.h:22:30: error:
‘numeric_limits’ is not a member of ‘std’
   22 |   static const T kmax = std::numeric_limits<T>::max();
      |                              ^~~~~~~~~~~~~~
```

原因是缺少头文件：

```
/buildroot/build/ros/google_benchmark_vendor/benchmark-1.5.2-
prefix/src/benchmark-1.5.2/src/benchmark_register.h

#include <limits>
```

- 该修改在ROS2较新版本中已经修复； 补丁包也有包含

7.5 Linux SDK工具链中定义_FORTIFY_SOURCE

```
--- stderr: mimick_vendor
Cloning into 'mimick-f171450b5ebaa3d2538c762a059dfc6ab7a01039'...
fatal: unable to access 'https://github.com/ros2/Mimick.git/': gnutls_handshake()
failed: Error in the pull function.
Cloning into 'mimick-f171450b5ebaa3d2538c762a059dfc6ab7a01039'...
HEAD is now at f171450 Add armv7l as a 32-bit ARM architecture. (#16)
In file included from /opt/aarch64-buildroot-linux-gnu_sdk-buildroot/aarch64-
buildroot-linux-gnu/sysroot/usr/include/errno.h:25,
                 from /buildroot/build/ros/mimick_vendor/mimick-
f171450b5ebaa3d2538c762a059dfc6ab7a01039-prefix/src/mimick-
f171450b5ebaa3d2538c762a059dfc6ab7a01039/include/mimick/mock.h:27,
                 from /buildroot/build/ros/mimick_vendor/mimick-
f171450b5ebaa3d2538c762a059dfc6ab7a01039-prefix/src/mimick-
f171450b5ebaa3d2538c762a059dfc6ab7a01039/include/mimick/mimick.h:401,
                 from /buildroot/build/ros/mimick_vendor/mimick-
f171450b5ebaa3d2538c762a059dfc6ab7a01039-prefix/src/mimick-
f171450b5ebaa3d2538c762a059dfc6ab7a01039/sample/strdup/test.c:1:
/opt/aarch64-buildroot-linux-gnu_sdk-buildroot/aarch64-buildroot-linux-
gnu/sysroot/usr/include/features.h:412:4: error: #warning _FORTIFY_SOURCE
requires compiling with optimization (-O) [-Werror=cpp]
  412 | #  warning _FORTIFY_SOURCE requires compiling with optimization (-O)
      |    ^~~~~~~
cc1: all warnings being treated as errors
make[5]: *** [sample/strdup/CMakeFiles/strdup_test.dir/build.make:63:
sample/strdup/CMakeFiles/strdup_test.dir/test.c.o] Error 1
make[4]: *** [CMakeFiles/Makefile2:302:
sample/strdup/CMakeFiles/strdup_test.dir/all] Error 2
make[4]: *** Waiting for unfinished jobs....
```

```
In file included from /opt/aarch64-buildroot-linux-gnu_sdk-buildroot/aarch64-
buildroot-linux-gnu/sysroot/usr/include/errno.h:25,
                 from /buildroot/build/ros/mimick_vendor/mimick-
f171450b5ebaa3d2538c762a059dfc6ab7a01039-prefix/src/mimick-
f171450b5ebaa3d2538c762a059dfc6ab7a01039/include/mimick/mock.h:27,
                 from /buildroot/build/ros/mimick_vendor/mimick-
f171450b5ebaa3d2538c762a059dfc6ab7a01039-prefix/src/mimick-
f171450b5ebaa3d2538c762a059dfc6ab7a01039/include/mimick/mimick.h:401,
                 from /buildroot/build/ros/mimick_vendor/mimick-
f171450b5ebaa3d2538c762a059dfc6ab7a01039-prefix/src/mimick-
f171450b5ebaa3d2538c762a059dfc6ab7a01039/test/test.c:1:
/opt/aarch64-buildroot-linux-gnu_sdk-buildroot/aarch64-buildroot-linux-
gnu/sysroot/usr/include/features.h:412:4: error: #warning _FORTIFY_SOURCE
requires compiling with optimization (-O) [-Werror=cpp]
  412 | #  warning _FORTIFY_SOURCE requires compiling with optimization (-O)
      |    ^~~~~~~
cc1: all warnings being treated as errors
```

- 报错仅在指定了-DCMAKE_TOOLCHAIN_FILE="/opt/aarch64-buildroot-linux-gnu_sdk-buildroot/share/buildroot/toolchainfile.cmake"交叉工具链，且该cmake定义了_FORTIFY_SOURCE
- 可不指定CMAKE_TOOLCHAIN_FILE，或删除_FORTIFY_SOURCE

7.6 CMake找不到exlibConfig.cmake

编译ament_cmake_vendor_package报找不到exlib，但实际该exlib库都被正确指定了。

```
root@db4be0cd3eca:/buildroot/build/ros/ament_cmake_vendor_package/test# make
[ 33%] Built target exlib_bad
[ 66%] Built target exlib_good
[ 71%] Performing configure step for 'depender'
loading initial cache file
/buildroot/build/ros/ament_cmake_vendor_package/test/depender-config.cmake
CMake Error at CMakeLists.txt:4 (find_package):
  By not providing "Findexlib.cmake" in CMAKE_MODULE_PATH this project has
  asked CMake to find a package configuration file provided by "exlib", but
  CMake did not find one.

  Could not find a package configuration file provided by "exlib" with any of
  the following names:

    exlibConfig.cmake
    exlib-config.cmake

  Add the installation prefix of "exlib" to CMAKE_PREFIX_PATH or set
  "exlib_DIR" to a directory containing one of the above files.  If "exlib"
  provides a separate development package or SDK, be sure it has been
  installed.

# strace make  可以看到:
```

```
[pid 458018] newfstatat(AT_FDCWD, "/opt/aarch64-buildroot-linux-gnu_sdk-
buildroot/aarch64-buildroot-linux-
gnu/sysroot/buildroot/build/ros/ament_cmake_vendor_package/test/exlib_bad-
prefix/install", 0x7ffca495cf50, 0) = -1 ENOENT (No such file or directory)

# :
# grep CMAKE_PREFIX_PATH depender-config.cmake
set(CMAKE_PREFIX_PATH [=
[/buildroot/build/ros/ament_cmake_vendor_package/test/exlib_bad-
prefix/install;/buildroot/build/ros/ament_cmake_vendor_package/test/exlib_good-
prefix/install;/buildroot/build/ros/ament_cmake_vendor_package/test/depender-
prefix/install]=] CACHE INTERNAL "")

它去找了Toolchain目录下的sysroot/$CMAKE_PREFIX_PATH，所以找不到。
```

- CMAKE_PREFIX_PATH设置是正确的，有包含exlib库的路径
- 通过strace make 可以看到工具链实际去找的路径不正确，多加了/opt/aarch64-buildroot-linux-gnu_sdk-buildroot/aarch64-buildroot-linux-gnu/sysroot/
- 原因： colcon命令中指定了参数 --cmake-args -DCMAKE_TOOLCHAIN_FILE="/opt/aarch64-buildroot-linux-gnu_sdk-buildroot/share/buildroot/toolchainfile.cmake" ，该设置与export环境变量、mimix中设置的编译工具链不同导致

7.7 pkg-config找不到

```
Starting >>> tracetools
--- stderr: tracetools
CMake Error at /usr/share/cmake-
3.22/Modules/FindPackageHandleStandardArgs.cmake:230(message):
  Could NOT find PkgConfig (missing: PKG_CONFIG_EXECUTABLE)
      Reason given by package: The command
      "/usr/bin/pkg-config" --version
      failed with output:
      stderr:
        /usr/bin/pkg-config: symbol lookup error: /usr/bin/pkg-config: undefined
symbol: pkgconf_cross_personality_deinit
      result:
  127
Call Stack (most recent call first):
  /usr/share/cmake-3.22/Modules/FindPackageHandleStandardArgs.cmake:594
(_FPHSA_FAILURE_MESSAGE)
  /usr/share/cmake-3.22/Modules/FindPkgConfig.cmake:99
(find_package_handle_standard_args)
  CMakeLists.txt:35 (find_package)
```

- 首先docker中应该有安装pkgconf（不是pkg-config），cmake中pkg_check_modules()会使用到
- Linux SDK中如果也编译了pkgconf，也会编译host-pkgconf，因为与docker的pkgconf版本不同， 在pkgconf.so动态库搜索时， 找到的是buildroot 编译的host pkgconf.so，所以失败

7.8 需要设置PKG_CONFIG_PATH

在编译src/ros2/ros2_tracing/tracetools/时， 其CMakeLists.txt中指定：

```
pkg_check_modules(LTTNG REQUIRED lttng-ust)
```

编译报错：

```
Starting >>> tracetools
--- stderr: tracetools
CMake Error at /usr/share/cmake-3.22/Modules/FindPkgConfig.cmake:611 (message):
  A required package was not found
Call Stack (most recent call first):
  /usr/share/cmake-3.22/Modules/FindPkgConfig.cmake:833
(_pkg_check_modules_internal)
  CMakeLists.txt:36 (pkg_check_modules)
```

- 通过strace -f 去抓取log， 发现并未在Linux SDK的sysroot中去查找，因此报错
- 需要设置环境变量：export PKG_CONFIG_PATH=/buildroot/host/aarch64-buildroot-linux-gnu/sysroot/usr/lib/pkgconfig

另一种情况是pkg-config找到了docker中的lttng，而不是target目标的，并报错如下：

```
Starting >>> tracetools
--- stderr: tracetools
/usr/lib/gcc-cross/aarch64-linux-gnu/11/../../../../aarch64-linux-gnu/bin/ld:
cannot find -llttng-ust-common: No such file or directory
collect2: error: ld returned 1 exit status
gmake[2]: *** [CMakeFiles/tracetools.dir/build.make:129: libtracetools.so] Error
1
gmake[1]: *** [CMakeFiles/Makefile2:161: CMakeFiles/tracetools.dir/all] Error 2
gmake[1]: *** Waiting for unfinished jobs....
gmake: *** [Makefile:146: all] Error 2
---
Failed   <<< tracetools [4.72s, exited with code 2]
```

- 因为找到的是docker的lttng，它的版本与buildroot不同， 前者lttng-ust.pc声明需要链接lttng-ust-common，但buildroot中缺少lttng-ust-common这个库
- 同样需要设置环境变量：export PKG_CONFIG_PATH=/buildroot/host/aarch64-buildroot-linux-gnu/sysroot/usr/lib/pkgconfig
- 该参数已经在编译脚本中指定
- Docker中可以不安装lttng包

7.9 需要设置CMAKE_INCLUDE_PATH

```
Starting >>> orocos_kdl_vendor
--- stderr: orocos_kdl_vendor
Cloning into 'orocos_kdl-507de66'...
done.
HEAD is now at 507de66 Fix CMake warning on Windows (#392)
Submodule 'python_orocos_kdl/pybind11' (https://github.com/pybind/pybind11.git)
registered for path 'python_orocos_kdl/pybind11'
Cloning into '/buildroot/build/ros/orocos_kdl_vendor/orocos_kdl-507de66-
prefix/src/orocos_kdl-507de66/python_orocos_kdl/pybind11'...
CMake Error: The following variables are used in this project, but they are set
to NOTFOUND.
Please set them or make sure they are set and tested correctly in the CMake
files:
EIGEN3_INCLUDE_DIR (ADVANCED)
```

因为Linux SDK编译过程中生成的include文件路径需要单独指定， 否则cmake无法搜索得到，如下：

```
export CMAKE_INCLUDE_PATH='/buildroot/host/aarch64-buildroot-linux-
gnu/sysroot/usr/include/'
```

- 该参数已经在编译脚本中指定

7.10 unsafe header/library used in cross-compilation

```
--- stderr: action_msgs
aarch64-buildroot-linux-gnu-gcc: WARNING: unsafe header/library path used in
cross-compilation: '-isystem' '/usr/local/lib/python3.10/dist-
packages/numpy/core/include'
```

在交叉编译过程中， python使用的是host端的/usr/bin/python，当numpy/numpyconfig.h查找不到时， 下列的获取include dir无法正确得到目标板子的路径：

```
# Check if numpy is in the include path
find_file(_numpy_h numpy/numpyconfig.h
  PATHS ${PythonExtra_INCLUDE_DIRS}
)

if(APPLE OR WIN32 OR NOT _numpy_h)
  # add include directory for numpy headers
  set(_python_code
    "import numpy"
    "print(numpy.get_include())"
  )
```

明确是由PythonExtra_INCLUDE_DIRS定义查找路径后， 在pybind11中查找该参数的定义是来自PYTHON_INCLUDE_DIR，因为我们是交叉编译，可在cross_compile.mimix中预设好该值。

- 该参数已经在编译脚本中指定，可以指定多个目录

8. 待完善项

8.1 删除不必要的安装文件

例如： cmake、头文件、静态库等，这些是可以安装到sysroot目录下的。

8.2 Linux SDK 中arm32位的rootfs尚未支持ROS2编译

9. 参考索引

1. https://docs.ros.org/en/foxy/Guides/Cross-compilation.html
2. https://github.com/ros-tooling/cross_compile.git
3. https://docs.ros.org/en/foxy/Releases.html