Rockchip ROS2 Introduction

ID: RK-SM-YF-912

Release Version: V1.1.1

Release Date: 2024-01-11

Security Level: □Top-Secret □Secret □Internal ■Public

Rockchip Electronics Co., Ltd.

No.18 Building, A District, No.89, software Boulevard Fuzhou, Fujian,PRC

Website: www.rock-chips.com

Customer service Tel: +86-4007-700-590

Customer service Fax: +86-591-83951833

Customer service e-Mail: fae@rock-chips.com

Preface

Overview

RK Buildroot SDK integrates ROS1 compilation, but ROS1 is not within the scope of this document. For the detailed content of ROS1, you can check the documents under the buildroot/package/rockchip/ros path and docs/.

For ROS2, RK Linux SDK provides the dependency packages needed by ROS2, and ROS2 is compiled in Docker independently, aiming to separate ROS2 from RK Linux SDK completely and facilitate ROS2 version update and maintenance.

ROS2 documentation[1] [2]shows that it provides Docker images for cross-compiling arm/aarch64, such as Dockerfile_ubuntu_arm64_prebuilt[2]. However, the target system only supports ubuntu 18.04 (bionic), which is quite different from the rootfs in the RK Linux SDK. Drawing on this idea, we summarize the compilation of ROS2 into three steps:

- Firstly, you have to finish RK Linux SDK compilation, which contains some application packages required by ROS2, such as python3, bullet, opencv, eigen, etc.
- Secondly, enter Docker, build ROS2 by Ubuntu environment, cross-compilation tool chain of RK Linux SDK and Sysroot
- Thirdly, Exit Docker and repackage RK Linux SDK Rootfs

Currently, there are four ROS2 distributions that have been verified to be built successfully: foxy, galactic, humble, and iron.

Product Version

Chipset	Kernel Version
RK3566,RK3568,RK3588 and other arm64-bit chips	It has nothing to do with the kernel version

Intended Audience

This document (this guide) is mainly intended for:

Technical support engineers

Software development engineers

Revision History

Version	Author	Date	Change Description
V1.0.0	zhengsq	2021-09-09	Initial version
V1.1.0	zhengsq	2023-09-21	Update to match the current new Linux SDK, add support for ROS2 humble and iron, modify the Docker compilation environment, and change the document structure
V1.1.1	zhengsq	2024-01-11	Update some expression

Contents

1. Versions of ROS2

Rockchip Linux SDK is built based on the Buildroot system, and the tool chain and software packages are continuously updated to the newer versions. Therefore, the latest SDK may encounter some bugs when compiling ROS2. For compilation methods we provided, we try to fix the version as much as possible:

- Fixed version of ROS2 source code, download it from [ROS2 githup](), and make sure the version number of each software package

The version numbers of each ROS2 release are as follows:

ROS2 release version	Version number	Link
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

RK Linux SDK versions which have been successfully built:

Linux SDK Buildroot Version	Python Version	Matching Docker image	Remarks
linux-5.10-stan-rkr1	Python 3.10.5	jammy-ros2-build	Need to add ros2_dep.config patch
linux-4.19-gen-rkr3	Python 3.8.6	focal-ros2-build	Need to update and enable lttng related patch packages: lttng-tools(2.12.3), lttng-libust(2.12.3), liburcu(0.13.0) needs to be updated

- Note: The Python versions in Docker host and Target rootfs need to be consistent
- Not verified for 32-bit systems

2. Files and Instructions of Building in Docker

The patches, Docker images, and source code described in this section can be downloaded from the following address:

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

```
tree
.
├── docker-focal-python38
│ └── rosdep.Dockerfile       # Dockerfile for making Docker Image
├── docker-jammy-python310
│ └── rosdep.Dockerfile       # Dockerfile for making Docker Image
├── focal-ros2-build.tar.gz   # Docker Image made based on Dockerfile
├── jammy-ros2-build.tar.gz   # Docker Image made based on Dockerfile
├── linux-sdk-patches
│ └── buildroot               # Patches that may be missing from different
release versions of RK Linux SDK may
│ ├── 0001-package-add-libasio.patch
│ └── 0002-configs-rockchip-add-ros2-build-dependencies.patch
├── MD5SUM.txt                # MD5SUM check code of each compressed package
├── ros2-build-scripts.tar.gz # Compile scripts and patches
└── ros2-sources.tar.gz       # Source code package of ROS2 and some of its
dependent libraries
```

In the Buildroot directory of RK Linux SDK, check whether there is a ros2_dep.config file,

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

# If LTTNG_TOOLS is missing in the ros2_dep.config, add it manually (ROS2 iron
has dependencies)
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
```

- If there is no such file, you need to add the following 2 patches in the Buildroot directory

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

- Check whether lttng-tools(2.12.3), lttng-libust(2.12.3), liburcu(0.13.0) meet the version requirements

3. Dependencies of Building ROS2

In the Buildroot project in RK Linux SDK, ros2_dep.config provides the dependency packages needed to build and run ROS2, which need to be added and built to rootfs. For example, add ros2_dep.config to rockchip_rk356x_robot_defconfig:

```
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"
```

After the rootfs is fully built, you can continue to the following steps..

4. Linux (Ubuntu) PC Environment Preparation

Install the Docker program in a Ubuntu PC:

```
sudo apt install docker.io
sudo usermod -aG docker $USER
newgrp docker  # Log in to docker user group
```

4.1 Import Docker Image

First check the Python version built by RK Linux SDK, for example:

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

According to the Python version number, match the corresponding Docker Image:

Python version	Matching Docker image
Python 3.10.5	jammy-ros2-build
Python 3.8.6	focal-ros2-build

Select jammy-ros2-build, import and enter 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
```

- `source=` needs to be modified to the absolute path of the output directory built by the corresponding Linux SDK
- After entering Container, the default user is builder and the default password is: rockchip

4.2 Copy the Compilation Script and Source Code Package

Use the docker container cp command to copy the required files:

```
# First find the logged in 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/
```

In the Container, decompress it:

```
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 Modify the Python Version Number in the Script

Check `/opt/ros/cross-compile/cross-compile.mixin` and `build_ros2.sh`, and change the Python version number to the version number corresponding to the RK Linux SDK, for example:

- 310 is changed to 38, where 310 represents the Python 3.10 version, and so on.
- 3.10 modified to 3.8

4.4 (Optional) Build ROS2 in Docker

If you want to make a Docker Image from the beginning, you can use the `rosdep.Dockerfile` provided by RK:

```
docker build -t jammy-ros2-build -f rosdep.Dockerfile ./  # don't forget to copy
"./" which means the current directory
```

4.5 (Optional) Download Source Code

If you need to download the source code of other versions yourself, after entering Docker, you can use `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. Build ROS2

Double check:

- RK Linux SDK has been added with `ros2_dep.config`, and rootfs has been fully built and passed.
- The selected Docker Image matches the Python compiled by RK Linux SDK

Select the required ROS2 version and execute the following commands in sequence:

```
  ls /opt/ros
cross-compile  foxy  galactic  humble  iron
  cd /opt/ros/iron
  ./prepare-source.sh
  ./build-ros2.sh
# After successful compilation, there should be a prompt similar to the
following:
...
Summary: 317 packages finished [15min 37s]
...
build ros quit & cleanup
```

Note：

- The compiled target files are located in the `/buildroot/target/opt/ros` directory.
- The intermediate compilation process is stored in the `/buildroot/build/ros` directory.

A successful compilation is indicated if build_ros2.sh does not prompt any errors. However, certain packages cannot be compiled or executed within the Buildroot SDK environment, such as:

- rviz, which relies on X11/desktop. If you require this functionality, use the Ubuntu arm image directly instead of Buildroot.
- turtlesim, which relies on UI to display.
- To exclude the compilation of a particular package, create a COLCON_IGNORE file in the corresponding src path. For example, `touch src/ros/ros_tutorials/turtlesim/COLCON_IGNORE`.

5.1 TRY_ RUN Requires Manual Execution and Recording of Results

fastrtps TRY_RUN prompt:

```
--- 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
```

- You need to follow the instructions to put the application on the board and execute it, and fill in the results according to the instructions. For example:

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

 # According to the above execution results, fill in the results in docker:
  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 prompt:

```
--- 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
```

- Same as above, you need to follow the instructions to put the application on the board and execute it, and fill in the results according to the instructions. For example:

```
  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 Build a ROS2 Package and Application Separately

Use the `colcon build` parameter `--packages-select <pacage_name>` to build a package separately. Please refer to the `colcon build --help` for details .

6. Package rootfs and Run ROS2

When finishing the above ROS2 compilation, enter the buildroot sdk and re-package the rootfs. ROS2 is installed in the /opt/ros directory.

```
cd /data/linux-sdk/rk3562
./build.sh rootfs   # Re-package rootfs.img
```

After flashing rootfs.img, enter the rk3562 EVB board and execute 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. Common Compilation Bugs That Have Been Fixed

7.1 Insufficient Memory of Compilation Host

Swap space can be enabled, such as 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 Reports `GLIBCXX_3.4.30' not found When running on the board

```
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)
```

Because there are many versions of RK Linux SDK and the tool chain version is constantly being updated, it is necessary to use the Linux SDK cross tool for compiling Rootfs to compile ROS2.

7.3 There is x86_64 Dynamic Library in the Compilation Result

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

There are indeed some known issues with pybind11 in a cross-compilation environment:

The python is the executable file on the HOST side, so a series of parameters are also generated based on the HOST side, such as:

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

In the newer code of `pybind11/tools/FindPythonLibsNew.cmake`, it is recommended that if it is Cross Compling, you can manually add python parameters externally. Based on this,

1. Modify `pybind11 in src/ros2/pybind11_vendor` to upgrade to v2.10.2
2. And set the following 2 parameters in `pybind11_verdor/CMakeLists.txt` to specify the detailed 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. In cross-compile.mixin, also need to declare:

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

After the above modifications, it was still found that when rclpy was compiled, the PYTHON_MODULE_EXTENSION obtained in the CMakeCache.txt still pointed to "x86_64", but when it was compiled again for the second time, it would be modified to the expected aarch64. The reason is:

1. In `pybind11/tools/pybind11NewTools.cmake`, if PYBIND11_PYTHON_EXECUTABLE_LAST has not been set, or its value has been modified, PYTHON_MODULE_EXTENSION will be cleared directly

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

2. There is a similar case in pybind11 issue #236
3. Solution: Modify pybind11: If PYBIND11_PYTHONLIBS_OVERWRITE = "OFF", the above parameters will not be reset:

```
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 The google_benchmark Project Is Missing the Limits Header File

When compiling foxy, the following error will be reported:

```
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();
      |                              ^~~~~~~~~~~~~~
```

The reason is missing header files:

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

#include <limits>
```

- This modification has been fixed in the newer version of ROS2; the patch package is also included

7.5 _FORTIFY_SOURCE is Defined in the Linux SDK Tool Chain

```
--- 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
```

- The error is reported only when `-DCMAKE_TOOLCHAIN_FILE="/opt/aarch64-buildroot-linux-gnu_sdk-buildroot/share/buildroot/toolchainfile.cmake"` cross tool chain is specified, and the cmake defines `_FORTIFY_SOURCE`
- CMAKE_TOOLCHAIN_FILE can be left unspecified, or delete _FORTIFY_SOURCE

7.6 Cannot Find exlibConfig.cmake in CMake

Compiling `ament_cmake_vendor_package` reports that exlib cannot be found, but in fact the exlib library is specified correctly.

```
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, then you will find:
[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 "")

It has gone to find sysroot/$CMAKE_PREFIX_PATH in the Toolchain directory, so it
couldn't find it.
```

- The CMAKE_PREFIX_PATH setting is correct and it is a path containing the exlib library
- Through `strace make`, you can see that the path actually found by the tool chain is incorrect, and `/opt/aarch64-buildroot-linux-gnu_sdk-buildroot/aarch64-buildroot-linux-gnu/sysroot/` is added.
- Reason: The parameter `--cmake-args -DCMAKE_TOOLCHAIN_FILE="/opt/aarch64-buildroot-linux-gnu_sdk-buildroot/share/buildroot/toolchainfile.cmake"` is specified in the colcon command. This setting is different from the export environment variable and the compilation tool chain set in mimix, which leads to this problem.

7.7 pkg-config is Not Found

```
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)
```

- First, `pkgconf` (not pkg-config) should be installed in docker, which will be used by `pkg_check_modules()` in cmake.
- If `pkgconf` is also compiled in the Linux SDK, `host-pkgconf` will also be compiled. Because it is different from the `pkgconf` version of docker, when searching for the `pkgconf.so` dynamic library, the host `pkgconf.so` compiled by buildroot is found, so it fails.

7.8 PKG_CONFIG_PATH Needs to Be Set

When compiling `src/ros2/ros2_tracing/tracetools/`, specifies in the CMakeLists.txt:

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

Compilation error:

```
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)
```

- Using `strace -f` to capture logs, it was found that the search did not occur within the sysroot of the Linux SDK, resulting in an error.
- The environment variable needs to be set: `export PKG_CONFIG_PATH=/buildroot/host/aarch64-buildroot-linux-gnu/sysroot/usr/lib/pkgconfig`

In another case, pkg-config found lttng within Docker instead of the target, resulting in the following error:

```
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]
```

- As the detected lttng within Docker has a different version from the one in Buildroot,The declaration of the former ttng-ust.pc needs to link to lttng-ust-common. However, the lttng-ust-common library is missing in Buildroot.
- Similarly, the environment variable needs to be set: `export PKG_CONFIG_PATH=/buildroot/host/aarch64-buildroot-linux-gnu/sysroot/usr/lib/pkgconfig`.
- This parameter has already been specified in the compilation script.
- It is not necessary to install the lttng package in Docker.

7.9 Setting CMAKE_INCLUDE_PATH is required

```
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)
```

Because the `include` file paths generated during the Linux SDK compilation process need to be specified separately, otherwise cmake cannot locate them, as shown below:

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

- This parameter has been specified in the compilation script

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'
```

During cross-compilation, Python uses the `/usr/bin/python` of the host. When `numpy/numpyconfig.h` cannot be found, the following method of obtaining the include directory cannot accurately get the path for the target board:

```
# 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())"
  )
```

After clarifying that the search path is defined by `PythonExtra_INCLUDE_DIRS`, The definition of finding the parameter in pybind11 is from PYTHON_INCLUDE_DIR. As it is cross-compiling, this value can be pre-set in the cross_compile.mimix file.

- This parameter has already been specified in the compilation script and multiple directory can be specified

8. Areas for Improvement

8.1 Removal of Unnecessary Installation Files

For instance, cmake, header files, static libraries, etc., which can be installed into the sysroot directory.

8.2 The arm32-bit rootfs in the Linux SDK Does Not Yet Support ROS2 Compilation

9. Reference Index

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