DPDK

DATA PLANE DEVELOPMENT KIT

Getting Started Guide for Windows

Release 20.08.0

Aug 08, 2020

CHAPTER

ONE

INTRODUCTION

This document contains instructions for installing and configuring the Data Plane Development Kit (DPDK) software. The document describes how to compile and run a DPDK application in a Windows* OS application environment, without going deeply into detail.

*Other names and brands may be claimed as the property of others.

CHAPTER

TWO

LIMITATIONS

DPDK for Windows is currently a work in progress. Not all DPDK source files compile. Support is being added in pieces so as to limit the overall scope of any individual patch series. The goal is to be able to run any DPDK application natively on Windows.

The ../contributing/abi_policy does not apply to the Windows build, as function versioning is not supported on Windows, therefore minor ABI versions may be incompatible.

CHAPTER

THREE

COMPILING THE DPDK TARGET FROM SOURCE

3.1 System Requirements

Building the DPDK and its applications requires one of the following environments:

- The Clang-LLVM C compiler and Microsoft MSVC linker.
- The MinGW-w64 toolchain (either native or cross).

The Meson Build system is used to prepare the sources for compilation with the Ninja backend. The installation of these tools is covered in this section.

3.2 Option 1. Clang-LLVM C Compiler and Microsoft MSVC Linker

3.2.1 Install the Compiler

Download and install the clang compiler from LLVM website. For example, Clang-LLVM direct download link:

```
http://releases.llvm.org/7.0.1/LLVM-7.0.1-win64.exe
```

3.2.2 Install the Linker

Download and install the Build Tools for Visual Studio to link and build the files on windows, from Microsoft website. When installing build tools, select the "Visual C++ build tools" option and ensure the Windows SDK is selected.

3.3 Option 2. MinGW-w64 Toolchain

On Linux, i.e. for cross-compilation, install MinGW-w64 via a package manager. Version 4.0.4 for Ubuntu 16.04 cannot be used due to a MinGW-w64 bug.

On Windows, obtain the latest version installer from MinGW-w64 repository. Any thread model (POSIX or Win32) can be chosen, DPDK does not rely on it. Install to a folder without spaces in its name, like `C:\MinGW`. This path is assumed for the rest of this guide.

3.4 Install the Build System

Download and install the build system from Meson website. A good option to choose is the MSI installer for both meson and ninja together:

```
http://mesonbuild.com/Getting-meson.html#installing-meson-and-ninja-with-the-msi-installer%22
```

Recommended version is either Meson 0.47.1 (baseline) or the latest release.

3.5 Install the Backend

If using Ninja, download and install the backend from Ninja website or install along with the meson build system.

3.6 Build the code

The build environment is setup to build the EAL and the helloworld example by default.

3.6.1 Option 1. Native Build on Windows

When using Clang-LLVM, specifying the compiler might be required to complete the meson command:

```
set CC=clang
```

When using MinGW-w64, it is sufficient to have toolchain executables in PATH:

```
set PATH=C:\MinGW\mingw64\bin;%PATH%
```

To compile the examples, the flag `-Dexamples` is required.

```
cd C:\Users\me\dpdk
meson -Dexamples=helloworld build
ninja -C build
```

3.6.2 Option 2. Cross-Compile with MinGW-w64

The cross-file option must be specified for Meson. Depending on the distribution, paths in this file may need adjustments.

```
meson --cross-file config/x86/cross-mingw -Dexamples=helloworld build
ninja -C build
```

CHAPTER

FOUR

RUNNING DPDK APPLICATIONS

4.1 Grant *Lock pages in memory* Privilege

Use of hugepages ("large pages" in Windows terminology) requires `SeLockMemoryPrivilege` for the user running an application.

1. Open *Local Security Policy* snap-in, either:
   - Control Panel / Computer Management / Local Security Policy;
   - or Win+R, type `secpol`, press Enter.
2. Open *Local Policies / User Rights Assignment / Lock pages in memory.*
3. Add desired users or groups to the list of grantees.
4. Privilege is applied upon next logon. In particular, if privilege has been granted to current user, a logoff is required before it is available.

See Large-Page Support in MSDN for details.

4.2 Load virt2phys Driver

Access to physical addresses is provided by a kernel-mode driver, virt2phys. It is mandatory at least for using hardware PMDs, but may also be required for mempools.

Refer to documentation in `dpdk-kmods` repository for details on system setup, driver build and installation. This driver is not signed, so signature checking must be disabled to load it.

> **Warning:** Disabling driver signature enforcement weakens OS security. It is discouraged in production environments.

Compiled package consists of `virt2phys.inf`, `virt2phys.cat`, and `virt2phys.sys`. It can be installed as follows from Elevated Command Prompt:

```
pnputil /add-driver Z:\path\to\virt2phys.inf /install
```

On Windows Server additional steps are required:

1. From Device Manager, Action menu, select "Add legacy hardware".
2. It will launch the "Add Hardware Wizard". Click "Next".
3. Select second option "Install the hardware that I manually select from a list (Advanced)".

4. On the next screen, "Kernel bypass" will be shown as a device class.
5. Select it, and click "Next".
6. The previously installed drivers will now be installed for the "Virtual to physical address translator" device.

When loaded successfully, the driver is shown in *Device Manager* as *Virtual to physical address translator* device under *Kernel bypass* category. Installed driver persists across reboots.

If DPDK is unable to communicate with the driver, a warning is printed on initialization (debug-level logs provide more details):

```
EAL: Cannot open virt2phys driver interface
```

4.3 Run the `helloworld` Example

Navigate to the examples in the build directory and run *dpdk-helloworld.exe*.

```
cd C:\Users\me\dpdk\build\examples
dpdk-helloworld.exe -l 0-3
hello from core 1
hello from core 3
hello from core 0
hello from core 2
```