(/)

An OS to build, deploy and securely manage billions of devices

Latest News:

Apache Mynewt 1.10.0, Apache NimBLE 1.5.0 (/download) released (May 6, 2022)

Installing Newt on Windows

You can develop and build Mynewt OS applications for your target boards on the Windows platform. This guide shows you how to install the latest release version of newt from binary or from source. The tool is written in Go (golang).

In Windows, we use MinGW as the development environment to build and run Mynewt OS applications for target boards. MinGW runs the bash shell and provides a Unix-like environment. This provides a uniform way to build Mynewt OS applications. The Mynewt documentation and tutorials use Unix commands and you can use the same Unix commands on MinGW to follow the tutorials. The documentation will note any commands or behaviors that are specific to Windows.

This guide shows you how to perform the following:

1. Install MSYS2/MinGW.
2. Install Git.
3. Install latest release (1.8.0) of newt from binary.
4. Install go for building newt from source.
5. Install latest release of newt from source.

See Installing Previous Releases of Newt (prev_releases.html) to install an earlier version of newt. You still need to set up your MinGW development environment.

Note: If you would like to contribute to the newt tool, see Contributing to Newt or Newtmgr Tools (../../misc/go_env.html).

Installing MSYS2/MinGW

MSYS2/MinGW provides a bash shell and tools to build applications that run on Windows. It includes three subsystems:

- MSYS2 toolchain to build POSIX applications that run on Windows.

- MinGW32 toolchains to build 32 bit native Windows applications.
- MinGW64 toolchains to build 64 bit native Windows applications.

The subsystems run the bash shell and provide a Unix-like environment. You can also run Windows applications from the shell. We will use the MinGW subsystem.

Note: You can skip this installation step if you already have MinGW installed (from an earlier MSYS2/MinGW or Git Bash installation), but you must list the **bin** path for your installation in your Windows Path. For example: if you installed MSYS2/MinGW in the **C:\msys64** directory, add **C:\msys64\usr\bin** to your Windows Path. If you are using Windows 10 WSL, ensure that you use the **C:\msys64\usr\bin\bash.exe** and not the Windows 10 WSL bash.

To install and setup MSYS2 and MinGW:

1. Download and run the MSYS2 installer (http://www.msys2.org). Select the 64 bit version if you are running on a 64 bit platform. Follow the prompts and check the `Run MSYS2 now` checkbox on the `Installation Complete` dialog.
2. In the MSYS2 terminal, run the `pacman -Syuu` command. If you get a message to run the update again, close the terminal and run the `pacman -Syuu` command in a new terminal.

   To start a new MSYS2 terminal, select the “MSYS2 MSYS” application from the Windows start menu.
3. Add a new user variable named **MSYS2_PATH_TYPE** and set the value to **inherit** in your Windows environment. This enables the MSYS2 and MinGW bash to inherit your Windows user **Path** values.

   To add the variable, select properties for your computer > Advanced system settings > Environment Variables > New
4. Add the MinGW **bin** path to your Windows Path. For example: if you install MSYS2/MinGW in the **C:\msys64** directory, add **C:\msys64\usr\bin** to your Windows Path.

   Note: If you are using Windows 10 WSL, ensure that you use the **C:\msys64\usr\bin\bash.exe** and not the Windows 10 WSL bash.
5. Run the `pacman -Su vim` command to install the vim editor.

   Note: You can also use a Windows editor. You can access your files from the **C:<msys-install-folder>\home\<username>** folder, where **msys-install-folder** is the folder you installed MSYS2 in. For example, if you installed MSYS2 in the **msys64** folder, your files are stored in **C:\msys64\home\<username>**
6. Run the `pacman -Su tar` command to install the tar tool.

You will need to start a MinGW terminal to run the commands specified in the Mynewt documentation and tutorials. To start a MinGW terminal, select the “MSYS2 Mingw” application from the start Menu (you can use either MinGW32 or MinGW64). In Windows, we use the MinGW subsystem to build Mynewt tools and applications.

Installing Git

Git can be installed as MinGW package from MinGW terminal:

```
$ pacman -S git
```

Alternatively download and install Git for Windows (https://git-for-windows.github.io).

Installing the Latest Release of the Newt Tool from Binary

You can install the latest release of newt from binary. It has been tested on Windows 10 64 bit platform.

1. Start a MinGW terminal.
2. Download the newt binary tar file apache-mynewt-newt-bin-windows-1.8.0.tgz (https://www.apache.org/dyn/closer.lua/mynewt/apache-mynewt-1.8.0/apache-mynewt-newt-bin-windows-1.8.0.tgz).
3. Extract the file:

    - If you previously built newt from the master branch, you can extract the file into your $GOPATH/bin directory. Note: This overwrites the current newt.exe in the directory and assumes that you are using $GOPATH/bin for your Go applications.

        ```
        $ tar -xzf apache-mynewt-newt-bin-windows-1.8.0.tgz -C $GOPATH/bin --strip-components=1 apache-mynewt-newt-bin-windows-1.8.0/newt.exe
        ```

    - If you are installing newt for the first time and do not have a Go workspace setup, you can extract into /usr/bin directory:

        ```
        $ tar -xzf apache-mynewt-newt-bin-windows-1.8.0.tgz -C /usr/bin --strip-components=1 apache-mynewt-newt-bin-windows-1.8.0/newt.exe
        ```

4. Verify the installed version of newt. See Checking the Installed Version.

Installing Go

Newt requires **Go** version **1.13** or higher. If you do not have Go installed, it can be installed from MinGW package repository.

1. Open a MinWG terminal.
2. Install go package.

```
$ pacman -S mingw-w64-x86_64-go
```

Alternatively newest version of Go from **golang.org** can be used. To download and install a newer version of Go (https://golang.org/dl/).

Installing the Latest Release of Newt From Source

If you have an older version of Windows or a 32 bit platform, you can build and install the latest release version of newt from source.

1. Start a MinGw terminal.
2. Download and unpack the newt source:

```
$ wget -P /tmp https://github.com/apache/mynewt-newt/archive/mynewt_1_8_0_tag.tar.gz
$ tar -xzf /tmp/mynewt_1_8_0_tag.tar.gz
```

3. Run the build.sh to build the newt tool.

```
$ cd mynewt-newt-mynewt_1_8_0_tag
$ MSYS=winsymlinks:nativestrict ./build.sh
$ rm /tmp/mynewt_1_8_0_tag.tar.gz
```

4. You should see the `newt/newt.exe` executable. Move the executable to a bin directory in your PATH:
    - If you previously built newt from the master branch, you can move the executable to the $GOPATH/bin directory.

```
$ mv newt/newt.exe $GOPATH/bin
```

- If you are installing newt for the first time and do not have a Go workspace set up, you can move the executable to /usr/bin or a directory in your PATH:

```
$ mv newt/newt.exe /usr/bin
```

Checking the Installed Version

1. Check the version of newt:

```
$ newt version
Apache Newt 1.8.0 / ab96a8a-dirty / 2020-03-18_23:25
```

2. Get information about newt:

```
$ newt help
Newt allows you to create your own embedded application based on the Mynewt
operating system. Newt provides both build and package management in a single
tool, which allows you to compose an embedded application, and set of
projects, and then build the necessary artifacts from those projects. For more
information on the Mynewt operating system, please visit
https://mynewt.apache.org/.

Please use the newt help command, and specify the name of the command you want
help for, for help on how to use a specific command

Usage:
  newt [flags]
  newt [command]

Examples:
  newt
  newt help [<command-name>]
    For help on <command-name>.  If not specified, print this message.

Available Commands:
  apropos      Search manual page names and descriptions
  build        Build one or more targets
  clean        Delete build artifacts for one or more targets
  create-image Add image header to target binary
  debug        Open debugger session to target
  docs         Project documentation generation commands
  help         Help about any command
  info         Show project info
  load         Load built target to board
  man          Browse the man-page for given argument
  man-build    Build man pages
  mfg          Manufacturing flash image commands
  new          Create a new project
  pkg          Create and manage packages in the current workspace
  resign-image Obsolete
  run          build/create-image/download/debug <target>
  size         Size of target components
  target       Commands to create, delete, configure, and query targets
  test         Executes unit tests for one or more packages
  upgrade      Upgrade project dependencies
  vals         Display valid values for the specified element type(s)
  version      Display the Newt version number

Flags:
      --escape            Apply Windows escapes to shell commands (default true)
  -h, --help              Help for newt commands
  -j, --jobs int          Number of concurrent build jobs (default 8)
  -l, --loglevel string   Log level (default "WARN")
```

```
  -o, --outfile string    Filename to tee output to
  -q, --quiet             Be quiet; only display error output
  -s, --silent            Be silent; don't output anything
  -v, --verbose           Enable verbose output when executing commands

Use "newt [command] --help" for more information about a command.
```

Next: Installing Native Toolchain (../../get_started/native_install/native_tools.html)

Previous: Installing Newt on Linux (newt_linux.html)

Apache Mynewt is available under Apache License, version 2.0.

Apache Mynewt, Mynewt, Apache, the Apache feather logo, and the Apache Mynewt project logo are either registered trademarks or trademarks of the Apache Software Foundation in the United States and other countries.

Add to Slack

()