(/)

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)

Docs (/documentation/) / Concepts
Edit on GitHub (https://github.com/apache/mynewt-documentation/edit/master/docs/concepts.rst)

Search documentation

Version: latest

Introduction (index.html)
Setup & Get Started (get_started/index.html)
Concepts
Tutorials (tutorials/tutorials.html)
Third-party Resources (external_links.html)
OS User Guide (os/os_user_guide.html)
BLE User Guide (network/index.html)
Newt Tool Guide (newt/index.html)
Newt Manager Guide (newtmgr/index.html)
Mynewt FAQ (mynewt_faq/index.html)
Appendix (misc/index.html)

Concepts

This page is meant to introduce you to some of the concepts inherent to the Apache Mynewt Operating System, and *Newt* the tool that stitches a project built on Apache Mynewt together.

- Project
- Package
- Target
- Configuration

Project

The project is the base directory of your embedded software tree. It is a workspace that contains a logical collection of source code, for one or more of your applications. A project consists of the following items:

- Project Definition: defines project level dependencies, and parameters (located in `project.yml`)
- Packages

Package are described in detail in the section below.

Here is an example project definition file from the default Apache Mynewt project:

```
# project.yml
# <snip>
project.name: "my_project"

project.repositories:
    - apache-mynewt-core

# Use github's distribution mechanism for core ASF libraries.
# This provides mirroring automatically for us.
#
repository.apache-mynewt-core:
    type: github
    vers: 1-latest
    user: apache
    repo: mynewt-core
```

A couple of things to note in the project definition:

- `project.repositories`: Defines the remote repositories that this project relies upon.
- `repository.apache-mynewt-core`: Defines the repository information for the `apache-mynewt-core` repository.
- `vers=1-latest`: Defines the repository version. This string will use the latest stable version in the 'Master' github branch. To use the latest version in the master branch, just change it to `vers=0-dev`. Note that this branch might not be stable.

Repositories are versioned collections of packages.

Projects can rely on remote repositories for functionality, and the newt tool will resolve those remote repositories, and download the correct version into your local source tree. Newly fetched repositories are put in the `repos` directory of your project, and can be referenced throughout the system by using the `@` specifier.

By default, the `@apache-mynewt-core` repository is included in every project. Apache Mynewt Core contains all the base functionality of the Apache Mynewt Operating System, including the Real Time Kernel, Bluetooth Networking Stack, Flash File System, Console, Shell and Bootloader.

NOTE: Any project can be converted into a repository by providing it with a `repository.yml` file and putting it up onto Github. More information about repositories can be found in the Newt documentation.

Package

A package is a collection items that form a fundamental unit in the Mynewt Operating System. Packages can be:

- Applications
- Libraries
- Compiler definitions
- Targets

A package is identified by having a `pkg.yml` file in it's base directory. Here is a sample `pkg.yml` file for the blinky applicaton:

```
# pkg.yml
# <snip>
pkg.name: apps/blinky
pkg.type: app
pkg.description: Basic example application which blinks an LED.
pkg.author: "Apache Mynewt <dev@mynewt.apache.org>"
pkg.homepage: "http://mynewt.apache.org/"
pkg.keywords:

pkg.deps:
    - "@apache-mynewt-core/libs/os"
    - "@apache-mynewt-core/hw/hal"
    - "@apache-mynewt-core/libs/console/full"
```

Packages have a few features worth noting:

- Dependencies: Packages can rely upon other packages, and when they do they will inherit their functionality (header files, library definitions, etc.)
- APIs: Packages can export named APIs, and they can require that certain APIs be present, in order to compile.

Everything that newt knows about within a project’s directory is a package. This makes it very clean and easy to write re-usable components, which can describe their Dependencies and APIs to the rest of the system.

Target

A target in Apache Mynewt is very similar to a target in *make*. It is the collection of parameters that must be passed to Newt in order to generate a reproducible build. A target represents the top of the build tree, and any packages or parameters specified at the target level, cascade down to all dependencies.

Targets are also packages, and are stored in the `targets/` directory at the base of your project. Most targets consist of:

- `app`: The application to build.
- `bsp`: The board support package to combine with that application
- `build_profile`: Either `debug` or `optimized`.

Targets can also have additional items specified, including:

- `aflags`: Any additional assembler flags you might want to specify to the build.
- `cflags`: Any additional compiler flags you might want to specify to the build.
- `lflags`: Any additional linker flags you might want to specify to the build.

In order to create and manipulate targets, the *newt* tool offers a set of helper commands, you can find more information about these by issuing:

```
$ newt target
Usage:
  newt target [flags]
  newt target [command]

Available Commands:
  amend       Add, change, or delete values for multi-value target variables
  cmake
  config      View or populate a target's system configuration
  copy        Copy target
  create      Create a target
  delete      Delete target
  dep         View target's dependency graph
  revdep      View target's reverse-dependency graph
  set         Set target configuration variable
  show        View target configuration variables

Global Flags:
  -h, --help              Help for newt commands
  -j, --jobs int          Number of concurrent build jobs (default 2)
  -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 target [command] --help" for more information about a command.
```

Configuration

Additional help topics:

```
$ newt target config show <target-name>
...
* PACKAGE: sys/stats
  * Setting: STATS_CLI
    * Description: Expose the "stat" shell command.
    * Value: 0
  * Setting: STATS_NAMES
    * Description: Include and report the textual name of each statistic.
    * Value: 0
  * Setting: STATS_NEWTMGR
    * Description: Expose the "stat" newtmgr command.
    * Value: 0
...
$
```

Previous: Debugging Mynewt (get_started/debug.html)

Next: Tutorials (tutorials/tutorials.html)