dragon

Release 1.7.3

cynder

Jul 02, 2023

CONTENTS:

"dragon" is a build system primarily targeting jailbroken iOS devices, capable of building tweaks, preferences, frameworks, apps, and anything else related to them.

It's designed to be simple, both in installation and usage, and to be hackable and configurable at every step of the way.

CHAPTER

ONE

SETUP

1.1 Installing

Installing is incredibly simple:

```
pip3 install dragon
```

Type “dragon” in your terminal to complete the initial setup

1.2 Updating

Versions 1.6.0 and later:

```
dragon update
```

Updating from earlier versions:

```
rm -rf ~/.dragon
pip3 install --force-reinstall dragon
dragon
```

CHAPTER

TWO

COMMANDS

Running `dragon` without any arguments will list available commands, many of which have multiple aliases.

You can combine most commands to do multiple actions with one command.

2.1 Packaging Commands

2.1.1 Creating a new project/module

`dragon n`, `dragon new`, `dragon nic`, `dragon edit`, or `dragon create` will open the Project Editor

2.1.2 Building a package

`dragon b`, `dragon build`, or `dragon make` builds a package

Building a package for release

The `r` / `release` command can be added to the `build` command to define "NDEBUG" and undefine "DEBUG" within compiled code.

Passing this flag will also cause the contents of the DragonMake variable `dbgflags` to be ignored, and the contents of `releaseflags` to be used instead.

2.1.3 Clean Building a package

`dragon c` or `dragon clean` will clean the 'build cache'

Combine it with the build command to run a clean build (e.g. `dragon c b`)

2.2 Device Commands

2.2.1 Setting up a device

`dragon s` or `dragon device` will set up an installation target

2.2.2 Installing a package

`dragon i` or `dragon install` installs a package

Combine it with the build command, or use `dragon do` to build and install a package

2.2.3 Respringing a device

`dragon rs` or `dragon respring` will respring the current device (i.e. current installation target)

2.2.4 Running a command on the device

`dragon dr <commands>` or `dragon devicerun <commands>` will execute anything after the command on the current device (i.e. current installation target) [don't use quotes]

2.2.5 Installing any deb on the device

`dragon sn <file>` or `dragon send <file>` will install a `.deb` anywhere on your drive to the current device (i.e. current installation target)

2.2.6 Building and installing to the iOS Simulator

Adding the `sim` command to a set of commands targets the simulator. If added to an `install` command, it will install the specified deb to the iOS simulator

CHAPTER

THREE

QUICK-START GUIDE

After completing the setup, getting started with dragon is easy.

Creating your first project:

```
dragon n
```

This opens the dragon project editor

```
kritanta@nocturne ~/src/demo
> $ dragon n
[Project Editor] Project Name (demo)
>> DemoTweak
[Project Editor] Bundle ID (com.kritanta.demo)
>> me.krit.dragondemo
[Project Editor] Description (A cool MobileSubstrate Tweak)
>> Demo Tweak
[Project Editor] Author (kritanta)
>> krit
[Project Editor] Select Module Type
[0] > Tweak
[1] > CLI Tool
[2] > Library
[Project Editor] Select Item (0)
>> 0
[Project Editor] Name (TweakName)
>> DemoTweak
[Project Editor] Subdiretory Name (Leave empty for no subdir) ()
>>
[Project Editor] Comma seperated list of applications to inject (SpringBoard)
>>
```

Building your project:

```
dragon b
```

Installing your project:

```
dragon i
```

You can do both of these at the same time; most commands in dragon can be combined:

```
dragon b i
```

Or you can use the shorthand notation:

```
dragon do
```

Building and installing to the iOS Simulator:

```
dragon b i sim
```

CHAPTER

FOUR

THE DRAGONMAKE FORMAT

Intead of splitting up build instructions among a ton of 'Makefile's, dragon build variables are all declared in a single *DragonMake* file at the root of the project.

DragonMake files use YAML syntax.

```
name: DemoTweak
id: me.cynder.dragondemo
depends: mobilesubstrate
architecture: iphoneos-arm
description: Demo Tweak
author: cynder
section: Tweaks

DemoTweak:
  type: tweak
  filter:
    executables:
      - SpringBoard
  files:
    - DemoTweak.x
```

4.1 The Project

The full *DragonMake* represents the "Project", which contains one or more "Modules" (tweaks, prefs, etc).

```
name: DemoTweak
id: me.cynder.dragondemo
depends: mobilesubstrate
architecture: iphoneos-arm
description: Demo Tweak
author: cynder
section: Tweaks
```

4.1.1 Variables

Variable	Type	Description
name	String	Name of the project
icmd	String	(Optional) Command to run after installation on the target device

4.1.2 *control* Variables

If your project already has a *control* file you don't need to worry about these.

Variable	Type	Description
id	String	Bundle ID (e.g. me.cynder.demotweak) for the Project
author	String	Author of the project. Current account's username will be used if none is provided
description	String	Description of the package
version	String	Version of the project
section	String	Section to place this tweak in. (e.g. 'Tweaks')
depends	String	Comma separated list of bundle ids this package depends on
maintainer	String	(Optional) Maintainer of the project. Will use the value of 'author' if none is provided
provides	String	(Optional) Comma separated list of bundle ids this package provides

4.1.3 Debian Package Script Variables

Lists of commands can be specified with *preinst:*, *postinst:*, *prerm:* and/or *postrm:* to create packaging scripts included in the binary.

```
name: DemoTweak
id: me.cynder.dragondemo
depends: mobilesubstrate
architecture: iphoneos-arm
description: Demo Tweak
author: cynder
section: Tweaks
# This will run on the device after installation
postinst:
  - echo "Hello from dragon!"
```

4.2 Modules

Modules in the *DragonMake* represent individual components of your package.

These include things like a Tweak, Preferences, etc.

```
DemoTweak:
  type: tweak
  filter:
    executables:
```

(continues on next page)

(continued from previous page)

```
      - SpringBoard
  files:
    - DemoTweak.x
```

4.2.1 The "Important" Variables

Variable	Type	Description
type	String	Project type – see next section
dir	String	(Optional) Subdirectory the files are located in, if they're in one
files	List	List of files in the project to be compiled

Types

Type	Description
app	Build an application for jailbroken devices
tweak	Build a tweak for jailbroken devices
prefs	Build a preference bundle
bundle	Build some other type of bundle
resource-bundle	Build a bundle containing only resources
framework	Build a framework
library	Build a library
cli	Build a CLI tool/binary
static	Build a static library
stage	Module containing only a stage variable

Tweak bundle filters

Bundle filters tell MobileSubstrate (or whatever injection system your jailbreak uses) what processes to inject your tweak into.

dragon supports the standard Theos format, but allows specifying the values in the *DragonMake*, if you want.

```
DemoTweak:
  type: tweak
  # This bit
  filter:
    executables:
      - SpringBoard

  files:
    - DemoTweak.x
```

4.2.2 Common Module variables

None of these are required by default, but you may need some of them for various projects.

Variable	Type	Description
archs	List	List of archs to compile for
cflags	String/List	List (or a space seperated string) with cflags used at compilation time
releaseflags	String/List	List (or a space seperated string) with cflags used on release (dragon b r) builds
dbgflags	String/List	List (or a space seperated string) with cflags used on debug builds (without r/release command)
frameworks	List	List of frameworks to link against
libs	List	List of libraries to link against
entfile	String	File containing entitlements to codesign the module with
include	List	List of directories to search for headers in
additional_fw_dirs	List	List of additional directories to search for frameworks in
additional_lib_dirs	List	List of additional directories to search for libraries in
prefix	List	List of headers to be imported into ALL files at compilation time
for	String	Sets the target OS to build for [ios, watchos, host(macos)]
arc	Boolean	Enable ARC (Default: YES)
sysroot	String	Specify Directory the SDK is located in
targetvers	String	Version of the OS to target
macros	List	List of declaration flags (-D<value>) to add to the compilation flags

4.2.3 Setting Module Defaults

A special module can be specified with the name *all:*; its variables will be set as the "default" value for all Modules in the project.

If a Module specifies a different value than *all:*, it'll override the one declared in *all:*.

CHAPTER

FIVE

STRUCTURE

dragon is set up such that the resources you need are provided via submodules and additional resources can be added as desired.

frameworks/:
A place for frameworks (.framework) [uses .tbd format]

include/:
A place for headers (.h)

internal/:
A place for YAML configuration files (.yml) [not meant to be edited, but feel free to get your hands dirty]

lib/:
A place for libraries [uses .dylib or .tbd format]

sdks/:
A place for SDKs (.sdk) [should be patched to include private frameworks]

src/:
A place for out-sourced tools modified and built for use with dragon

toolchain/:
A place for a user-provided toolchain [unnecessary on Darwin platforms]

vendor/:
A place for tools and resources provided by dragon [not meant to be edited]

CHAPTER

SIX

THEOS SUPPORT

dragon aims to provide as much compatibility with theos projects and their structure as possible.

6.1 *control* files, Bundle filters, etc.

dragon ships with support for these in both Theos Makefile and DragonMake format projects.

6.2 Makefile interpreter

dragon includes a best-effort Makefile "interpreter" that attempts to translate as much from standard Theos project structure as possible.

It also includes several support files used with Theos projects.

Compiling a Theos project should be as simple as:

```
dragon b
```

If you encounter any issues with it, feel free to file an issue on https://github.com/DragonBuild/dragon.