
macholib Documentation

Release 1.16.2

Ronald Oussoren

Sep 25, 2022

Contents

1	General documentation	3
2	Reference Guide	11
3	Online Resources	19
4	Contributors	21
5	Indices and tables	23
	Python Module Index	25
	Index	27

macholib can be used to analyze and edit Mach-O headers, the executable format used by Mac OS X.

It's typically used as a dependency analysis tool, and also to rewrite dylib references in Mach-O headers to be `@executable_path` relative.

Though this tool targets a platform specific file format, it is pure python code that is platform and endian independent.

1.1 Release history

1.1.1 macholib 1.16.2

- Update classifiers for Python 3.11

1.1.2 macholib 1.16.1

- Added some new CPU subtype definitions

1.1.3 macholib 1.16

- Add `allow_unknown_load_commands` option to `MachO` and ``macholib.MachOHeader`.
PR by github user bhamiltoncx.

1.1.4 macholib 1.15.2

- Explicitly mention Python 3.10 in the project metadata

1.1.5 macholib 1.15.1

- Remove a debug print introduced in the previous release.

1.1.6 macholib 1.15

- Add support for new features in the macOS 11 SDK
- Fix link to repository in README.rst
- Fix `macholib.dyld.dyld_find` for system libraries on macOS 11 or later

1.1.7 macholib 1.14

- Repository moved to GitHub
- #32: Update the LC_NAMES table

1.1.8 macholib 1.13

31: Add two new load commands introduced in macOS 10.15

1.1.9 macholib 1.12

- #27: Missing describe method for `build_version_command`

1.1.10 macholib 1.11

- Add very hacky limited support for `@loader_path`. This is just enough to deal with extensions and dylibs found in Python binary wheels.

1.1.11 macholib 1.10

- #25: Add support for LC_NOTE and LC_BUILD_VERSION

1.1.12 macholib 1.9

Features:

- Add definition for `macholib.mach_o.reloc_type_generic`, which was used in code but never defined.
- #22: Add LICENSE file
- #23: Added “-help” option for “python -m macholib”
- Added function `macholib.MachO.lc_str_value` which should help in decoding value of `macholib.mach_o.lc_str`. Those values are offsets in the data of a load command, the function will return the actually value as a byte string.

See also issue #21.

Bug fixes:

- Pull request #15: Fix typo in `thread_command` class
- Patch by user “phdphuc” on bitbucket.

1.1.13 macholib 1.8

- Use the same dependency walk logic as otool
Patch by Taras Tsugrii <ttsugrii@fb.com>
- Added support for new load commands
Patch by David Dorsey <trogdorsey@gmail.com>, with enhancements by Ronald Oussoren.
- Fix procesing DSYM file from XCODE 6.x
Patch by HolmsBlazhey <andrey.blazhey@gmail.com>
- MachOGraph.locate(): When calling dyld_find(), use kwarg 'loader_path', not 'loader'.
Patch by Stuart Berg <bergs@janelia.hhmi.org>
- Add fields to thread_command
Patch by Asger Hautop Drewsen <asgerdrewsen@gmail.com>
- Add missing ARM_V7S subtype.
Patch by "NN"
- Fix for SymbolTable
Patch by Christian Klein <chris@5711.org>
- Use first Mach-O header as the default header
Patch by Christian Klein <chris@5711.org>
- Issue #17: add LC_LOAD_UPWARD_DYLIB to _RELOCATABLE set
- Issue #16: macholib "hangs" on invalid input
Due to the use of the range function on untrusted input the python process could hang when reading invalid input, due to trying to construct an enormous list.
- Issue #18: Bad version parsing in macho_version_helper
The order of subfields in mach_version_helper was reversed from reality.
- Issue #19: Fix aligment issue that prevented code signing
Patch by Brendan Simon
- Fix issue #14: Can't pass endian argument to p_uint64.from_str

1.1.14 macholib 1.7

- Added support for ARM64, LC_ENCRYPTION_INFO_64 and LC_LINKER_OPTION
Patch by Matthias Ringwald.
- Load commands now have a "describe" method that returns more information about the command.
Patch by David Dorsey.
- The MAGIC value in the header was always represented in the native byte order, instead of as the value read from the binary.
Patch by David Dorsey.

- Added various new constants to “macholib.mach_o”.

Patch by David Dorsey.

1.1.15 macholib 1.6.1

- ?

1.1.16 macholib 1.6

- Add support for ‘@loader_path’ link command in macholib.dyld:
 - Added function `macholib.dyld.dyld_loader_search`
 - This function is used by `macholib.dyld.dyld_find`, and that function now has an new (optional) argument with the path to the loader.
- Also add support for ‘@loader_path’ to macholib.MachoGraph, using the newly added ‘@loader_path’ support in the dyld module.

Due to this support the *macho_standalone* tool can now rewrite binaries that contain an ‘@loader_path’ load command.

1.1.17 macholib 1.5.2

- Issue #93: Show the name of the affected file in the exception message for Mach-O headers that are too large to relocate.

1.1.18 macholib 1.5.1

- There were no ‘classifiers’ in the package metadata due to a bug in setup.py.

1.1.19 macholib 1.5

macholib 1.5 is a minor feature release

- No longer use 2to3 to provide Python 3 support
As a side-effect of this macholib no longer supports Python 2.5 and earlier.
- Adds support for some new macho load commands
- Fix for py3k problem in `macho_standalone.py`
Patch by Guanqun Lu.
- Fix for some issues in `macho_dump.py`
Patch by Nam Nguyen
- Issue #10: Fix for LC_DATA_IN_CODE linker commands, without this fix py2app cannot build application bundles when the source binaries have been compiled with Xcode 4.5.
- Issue #6: Fix for LC_ENCRYPTION_INFO linker commands

- Use the mach header information to print the cpu type of a binary, instead of trying to deduce that from pointer width and endianness.

Changed the code because of issue #6, in which a user tries to dump a iOS binary which results in bogus output in the previous releases.

- The mapping `macholib.macho_dump.ARCH_MAP` is undocumented and no longer used by macholib itself. It will be removed in the next release.
- The command-line tools `macho_find`, `macho_dump` and `macho_standalone` are deprecated. Use “python -mmacholib” instead. That is:

```
$ python -mmacholib dump /usr/bin/grep
$ python -mmacholib find ~
$ python -mmacholib standalone myapp.app
```

This makes it clearer which version of the tools are used.

1.1.20 macholib 1.4.3

macholib 1.4.3 is a minor feature release

- Added strings for ‘x86_64’ and ‘ppc64’ to `macholib.mach_o.CPU_TYPE_NAMES`.
- `macho_find` and `macho_dump` were broken in the 1.4.2 release
- added ‘`macholib.util.NOT_SYSTEM_FILES`’, a list of files that aren’t system path’s even though they are located in system locations.

Needed to work around a bug in PySide (see issue #32 in the py2app tracker)

1.1.21 macholib 1.4.2

macholib 1.4.2 is a minor bugfix release

- The support for new load commands that was added in 1.4.1 contained a typo that caused problems on OSX 10.7 (Lion).

1.1.22 macholib 1.4.1

macholib 1.4.1 is a minor feature release

Features:

- Add support for a number of new MachO load commands that were added during the lifetime of OSX 10.6: `LC_LOAD_UPWARD_DYLIB`, `LC_VERSION_MIN_MACOSX`, `LC_VERSION_MIN_IPHONEOS` and `LC_FUNCTION_STARTS`.

1.1.23 macholib 1.4

macholib 1.4 is a feature release

Features:

- Documentation is now generated using [sphinx](http://packages.python.org/macholib) and can be viewed at <http://packages.python.org/macholib>.

- The repository has moved to bitbucket
- There now is a testsuite
- Private functionality inside modules was renamed to a name starting with an underscore.

Note: if this change affects your code you are relying on undefined implementation features, please stop using private functions.

- The basic packable types in `macholib.ptypes` were renamed to better represent the corresponding C type. The table below lists the old and new names (the old names are still available, but are deprecated and will be removed in a future release).

Old name	New name
<code>p_byte</code>	<code>p_int8</code>
<code>p_ubyte</code>	<code>p_uint8</code>
<code>p_short</code>	<code>p_int16</code>
<code>p_ushort</code>	<code>p_uint16</code>
<code>p_int</code>	<code>p_int32</code>
<code>p_uint</code>	<code>p_uint32</code>
<code>p_long</code>	<code>p_int32</code>
<code>p_ulong</code>	<code>p_uint32</code>
<code>p_longlong</code>	<code>p_int64</code>
<code>p_ulonglong</code>	<code>p_uint64</code>

`Macholib.ptypes.p_ptr` is no longer present as it had an unclear definition and isn't actually used in the codebase.

Bug fixes:

- The semantics of `dyld.dyld_default_search` were changed a bit, it now first searches the framework path (if appropriate) and then the linker path, irrespective of the value of the `DYLD_FALLBACK*` environment variables.

Previous versions would change the search order when those variables was set, which is odd and doesn't correspond with the documented behaviour of the system dyld.

- It is once again possible to install using python2.5
- The source distribution includes all files, this was broken due to the switch to mercurial (which confused setup-tools)

1.1.24 macholib 1.3

macholib 1.3 is a feature release.

Features:

- Experimental Python 3.x support

This version contains lightly tested support for Python 3.

1.1.25 macholib 1.2.2

macholib 1.2.2 is a bugfix release.

Bug fixes:

- Macholib should work better with 64-bit code (patch by Marc-Antoine Parent)

1.2 License

Copyright (c) Bob Ippolito

Parts are copyright (c) 2010-2014 Ronald Oussoren

1.2.1 MIT License

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.3 Command-line tools

1.3.1 `python -m macholib find`

Usage:

```
$ python -mmacholib find dir...
```

Print the paths of all MachO binaries in the specified directories.

1.3.2 `python -m macholib standalone`

Usage:

```
$ python -m macholib standalone appbundle...
```

Convert one or more application bundles into standalone bundles. That is, copy all non-system shared libraries and frameworks used by the bundle into the bundle and rewrite load commands.

1.3.3 `python -mmacholib dump`

Usage:

```
$ python -mmacholib dump dir...
```

Prints information about all architectures in a Mach-O file as well as all libraries it links to.

2.1 macholib.MachO — Utilities for reading and writing Mach-O headers

This module defines a class `MachO`, which enables reading and writing the Mach-O header of an executable file or dynamic library on MacOS X.

`macholib.MachO.lc_str_value(offset, cmd_info)`

Returns the bytes for an `lc_str` value, given the *offset* of type `lc_str` and the `cmd_info` that contains the structure that contains the `lc_str` value. `cmd_info` is an item in the `commands` attribute of a `MachOHeader` instance.

class `macholib.MachO.MachO(filename, allow_unknown_load_commands=False)`

Creates a `MachO` object by reading the Mach-O headers from *filename*.

The *filename* should refer to an existing file in Mach-O format, and can refer to fat (universal) binaries.

When *allow_unknown_load_commands* is false the instance will raise an error when the specified file contains unknown load commands.

Note: more information will be added later

2.2 macholib.MachOGraph — Graph data structure of Mach-O dependencies

This module defines the class `MachOGraph` which represents the direct and indirect dependencies of one or more Mach-O files on other (library) files.

class `macholib.MachOGraph.MachOGraph(...)`

To be discussed.

2.3 macholib.MachOStandalone — Create standalone application bundles

This module defines class *MachOStandalone* which locates all Mach-O files in a directory (assumed to be the root of an application or plugin bundle) and then copies all non-system dependencies for the located files into the bundle.

```
class macholib.MachOStandalone.MachOStandalone (base[, dest[, graph[, env[, executable_path]]]])
```

2.4 macholib.SymbolTable — Class to read the symbol table from a Mach-O header

This module is deprecated because it is not by the author and likely contains bugs. It also does not work for 64-bit binaries.

```
class macholib.SymbolTable.SymbolTable (macho[, openfile])
```

Reads the SymbolTable for the given Mach-O object.

The option argument *openfile* specifies the function to use to open the file, defaulting to the builtin `open()` function.

Warning: As far as we know this class is not used by any user of the modulegraph package, and the code has not been updated after the initial implementation.

The end result of this is that the code does not support 64-bit code at all and likely doesn't work properly for 32-bit code as well.

2.5 macholib.dyld — Dyld emulation

This module defines a number of functions that can be used to emulate the functionality of the dynamic linker (dyld) w.r.t. looking for library files and frameworks.

```
macholib.dyld.dyld_image_suffix([env])
```

Looks up the suffix to append to shared library and framework names and returns this value when found. Returns `None` when no suffix should be appended.

The *env* argument is a dictionary, which defaults to `os.environ`.

See the description of `DYLD_IMAGE_SUFFIX` in the manual page for `dyld(1)` for more information.

```
macholib.dyld.dyld_framework_path([env])
```

Returns a user-specified framework search path, or an empty list when only the default search path should be used.

The *env* argument is a dictionary, which defaults to `os.environ`.

See the description of `DYLD_FRAMEWORK_PATH` in the manual page for `dyld(1)` for more information.

```
macholib.dyld.dyld_library_path([env])
```

Returns a user-specified library search path, or an empty list when only the default search path should be used.

The *env* argument is a dictionary, which defaults to `os.environ`.

See the description of `DYLD_LIBRARY_PATH` in the manual page for `dyld(1)` for more information.

`macholib.dyld.dyld_fallback_framework_path([env])`

Return a user specified list of directories where to look for frameworks that aren't in their install path, or an empty list when the default fallback path should be used.

The *env* argument is a dictionary, which defaults to `os.environ`.

See the description of DYLD_FALLBACK_FRAMEWORK_PATH in the manual page for dyld(1) for more information.

`macholib.dyld.dyld_fallback_library_path([env])`

Return a user specified list of directories where to look for libraries that aren't in their install path, or an empty list when the default fallback path should be used.

The *env* argument is a dictionary, which defaults to `os.environ`.

See the description of DYLD_FALLBACK_LIBRARY_PATH in the manual page for dyld(1) for more information.

`macholib.dyld.dyld_image_suffix_search(iterator[, env])`

Yields all items in *iterator*, and prepends names with the image suffix to those items when the suffix is specified.

The *env* argument is a dictionary, which defaults to `os.environ`.

`macholib.dyld.dyld_override_search(name[, env])`

If *name* is a framework name yield filesystem paths relative to the entries in the framework search path.

Always yield the filesystem paths relative to the entries in the library search path.

The *env* argument is a dictionary, which defaults to `os.environ`.

`macholib.dyld.dyld_executable_path_search(name, executable_path)`

If *name* is a path starting with `@executable_path/` yield the path relative to the specified *executable_path*.

If *executable_path* is None nothing is yielded.

`macholib.dyld.dyld_loader_search(name, loader_path)`

If *name* is a path starting with `@loader_path/` yield the path relative to the specified *loader_path*.

If *loader_path* is None nothing is yielded.

`macholib.dyld.dyld_default_search(name[, env])`

Yield the filesystem locations to look for a dynamic library or framework using the default locations used by the system dynamic linker.

This function will look in `~/Library/Frameworks` for frameworks, even though the system dynamic linker doesn't.

The *env* argument is a dictionary, which defaults to `os.environ`.

`macholib.dyld.dyld_find(name[, executable_path[, env[, loader_path]]])`

Returns the path of the requested dynamic library, raises `ValueError` when the library cannot be found.

This function searches for the library in the same locations and the system dynamic linker.

The *executable_path* should be the filesystem path of the executable to which the library is linked (either directly or indirectly).

The *env* argument is a dictionary, which defaults to `os.environ`.

The *loader_path* argument is an optional filesystem path for the object file (binary of shared library) that references *name*.

Changed in version 1.6: Added the *loader_path* argument.

`macholib.dyld.framework_find(fn[, executable_path[, env]])`

Find a framework using the same semantics as the system dynamic linker, but will accept looser names than the system linker.

This function will return a correct result for input values like:

- Python
- Python.framework
- Python.framework/Versions/Current

2.6 macholib.dylib — Generic dylib path manipulation

This module defines a function `dylib_info()` that can extract useful information from the name of a dynamic library.

`macholib.dylib.dylib_info(filename)`

A dylib name can take one of the following four forms:

- Location/Name.SomeVersion_Suffix.dylib
- Location/Name.SomeVersion.dylib
- Location/Name_Suffix.dylib
- Location/Name.dylib

Returns None if not found or a mapping equivalent to:

```
dict(  
    location='Location',  
    name='Name.SomeVersion_Suffix.dylib',  
    shortname='Name',  
    version='SomeVersion',  
    suffix='Suffix',  
)
```

Note: *SomeVersion* and *Suffix* are optional and may be None if not present.

2.7 macholib.framework — Generic framework path manipulation

This module defines a function `framework_info()` that can extract useful information from the name of a dynamic library in a framework.

`macholib.framework.framework_info(filename)`

A framework name can take one of the following four forms:

- Location/Name.framework/Versions/SomeVersion/Name_Suffix
- Location/Name.framework/Versions/SomeVersion/Name
- Location/Name.framework/Name_Suffix
- Location/Name.framework/Name

Returns None if not found, or a mapping equivalent to:

```
dict(
    location='Location',
    name='Name.framework/Versions/SomeVersion/Name_Suffix',
    shortname='Name',
    version='SomeVersion',
    suffix='Suffix',
)
```

Note: *SomeVersion* and *Suffix* are optional and may be None if not present.

2.8 macholib.mach_o — Low-level definitions

This module defines constants and packable structure types that correspond to elements of a Mach-O file.

The names of classes and constants is the same as those in the Mach-O header files and [Apple's documentation](#). This document therefore doesn't explicitly document the names in this module.

2.9 macholib.ptypes — Packable types

The module `macholib.ptypes` defines types that can be serialized into byte arrays, both for basic types and structured types (C struct values).

2.9.1 Utility functions

`macholib.ptypes.sizeof(value)`

Returns the size in bytes of an object when packed, raises `ValueError` for inappropriate values.

`macholib.ptypes.pyunpackable(name, pytype, format)`

Returns a packable type that is a subclass of the Python type `pytype`. The value is converted to and from the packed format using the struct `format`.

2.9.2 Packable types

class `macholib.ptypes.BasePackable`

All packable types are a subclass of `BasePackable`, which defines the basic interface but is itself an abstract base class.

`_endian_`

The byteorder of a packed value. This will be `"<"` for little endian values and `">"` for big-endian ones.

Note: the `endianness` option is a public value to be able to support both big- and little-endian file formats.

The name suggests that this attribute is private, this is partially for historical reasons and partially to avoid conflicts with field names in C structs.

from_mmap (*mmap*, *ptr*, ***kw*)

This class method constructs the value from a subview of a `mmap.mmap` object. It uses bytes starting at offset *ptr* and reads just enough bytes to read the entire object.

from_fileobj (*fp*, ***kw*)

This class method constructs the value by reading just enough bytes from a file-like object.

Note: The file must be opened in binary mode, that is read calls should return byte-strings and not unicode-strings.

from_str (*value*, ***kw*)

This class method construct the value by using the struct module to parse the given bytes.

Note: contrary to what the name suggests the argument to this method is a byte-string, not a unicode-string.

from_tuple (*fp*, ***kw*)

This class method constructs the object from a tuple with all fields.

to_str ()

Returns a byte representation of the value.

Note: there is no default implementation for this method

to_fileobj (*fp*)

Write a byte representation of the value to the given file-like object. The file should be opened in binary mode.

to_mmap (*mmap*, *ptr*)

Write the byte representation of the value to a `mmap.mmap` object, starting at offset *ptr*.

class macholib.ptypes.**Structure** (...)

fields

This class attribute is a list that contains the fields of the structure in the right order. Every item of this list is a tuple with 2 arguments: the first element is the name of the field, and the second the packable type for the field.

Every subclass of `Structure` must define `_fields_` to be usefull, and the value of `_fields_` should not be changed after class construction.

2.9.3 Basic packables

Other than the core functionality this module defines a number of `pypackable()` types that correspond to useful basic C types.

class macholib.ptypes.**p_char** ([*value*])

A byte string of length 1

class macholib.ptypes.**p_int8**

An 8-bit signed integer

class macholib.ptypes.**p_uint8**

An 8-bit unsigned integer

```
class macholib.ptypes.p_int16  
    An 16-bit signed integer  
class macholib.ptypes.p_uint16  
    An 16-bit unsigned integer  
class macholib.ptypes.p_int32  
    An 32-bit signed integer  
class macholib.ptypes.p_uint32  
    An 32-bit unsigned integer  
class macholib.ptypes.p_int64  
    An 64-bit signed integer  
class macholib.ptypes.p_uint64  
    An 64-bit unsigned integer  
class macholib.ptypes.p_float  
    An floating point value of type float  
class macholib.ptypes.p_double  
    An floating point value of type double
```

Note: the module exports a number of other types with names starting with `p_`, such as `p_int`. Those types are deprecated and should not be used.

CHAPTER 3

Online Resources

- [Sourcecode repository on bitbucket](#)
- [The issue tracker](#)
- [Mac OS X ABI Mach-O File Format Reference at Apple](#)

CHAPTER 4

Contributors

Macholib was written by Bob Ippolito and is currently maintained by Ronald Oussoren <ronaldoussoren@mac.com>.

CHAPTER 5

Indices and tables

- `genindex`
- `modindex`
- `search`

m

- `macholib.dyld`, [12](#)
- `macholib.dylib`, [14](#)
- `macholib.framework`, [14](#)
- `macholib.mach_o`, [15](#)
- `macholib.MachO`, [11](#)
- `macholib.MachOGraph`, [11](#)
- `macholib.MachOStandalone`, [12](#)
- `macholib.ptypes`, [15](#)
- `macholib.SymbolTable`, [12](#)

B

BasePackable (class in macholib.ptypes), 15

BasePackable._endian_ (in module macholib.ptypes), 15

D

dyld_framework_path() (in module macholib.dyld), 12

dyld_default_search() (in module macholib.dyld), 13

dyld_executable_path_search() (in module macholib.dyld), 13

dyld_fallback_framework_path() (in module macholib.dyld), 12

dyld_fallback_library_path() (in module macholib.dyld), 13

dyld_find() (in module macholib.dyld), 13

dyld_image_suffix() (in module macholib.dyld), 12

dyld_image_suffix_search() (in module macholib.dyld), 13

dyld_library_path() (in module macholib.dyld), 12

dyld_loader_search() (in module macholib.dyld), 13

dyld_override_search() (in module macholib.dyld), 13

dylib_info() (in module macholib.dylib), 14

F

framework_find() (in module macholib.dyld), 13

framework_info() (in module macholib.framework), 14

from_fileobj() (macholib.ptypes.BasePackable method), 16

from_mmap() (macholib.ptypes.BasePackable method), 15

from_str() (macholib.ptypes.BasePackable method), 16

from_tuple() (macholib.ptypes.BasePackable method), 16

L

lc_str_value() (in module macholib.MachO), 11

M

MachO (class in macholib.MachO), 11

MachOGraph (class in macholib.MachOGraph), 11

macholib.dyld (module), 12

macholib.dylib (module), 14

macholib.framework (module), 14

macholib.mach_o (module), 15

macholib.MachO (module), 11

macholib.MachOGraph (module), 11

macholib.MachOStandalone (module), 12

macholib.ptypes (module), 15

macholib.SymbolTable (module), 12

MachOStandalone (class in macholib.MachOStandalone), 12

P

p_char (class in macholib.ptypes), 16

p_double (class in macholib.ptypes), 17

p_float (class in macholib.ptypes), 17

p_int16 (class in macholib.ptypes), 17

p_int32 (class in macholib.ptypes), 17

p_int64 (class in macholib.ptypes), 17

p_int8 (class in macholib.ptypes), 16

p_uint16 (class in macholib.ptypes), 17

p_uint32 (class in macholib.ptypes), 17

p_uint64 (class in macholib.ptypes), 17

p_uint8 (class in macholib.ptypes), 16

pypackable() (in module macholib.ptypes), 15

S

sizeof() (in module macholib.ptypes), 15

Structure (class in macholib.ptypes), 16

`Structure.__fields__` (*in module macholib.ptypes*),
[16](#)
`SymbolTable` (*class in macholib.SymbolTable*), [12](#)

T

`to_fileobj()` (*macholib.ptypes.BasePackable*
method), [16](#)
`to_mmap()` (*macholib.ptypes.BasePackable method*),
[16](#)
`to_str()` (*macholib.ptypes.BasePackable method*), [16](#)