NAME

File::MimeInfo − Determine file type from the file name

SYNOPSIS

```
  use File::MimeInfo;
  my $mime_type = mimetype($file);
  my $mime_type2 = mimetype('test.png');
```

DESCRIPTION

This module can be used to determine the mime type of a file. It tries to implement the freedesktop specification for a shared MIME database.

For this module shared-mime-info-spec 0.13 was used.

This package only uses the globs file. No real magic checking is used. The File::MimeInfo::Magic package is provided for magic typing.

If you want to determine the mimetype of data in a memory buffer you should use File::MimeInfo::Magic in combination with IO::Scalar.

This module loads the various data files when needed. If you want to hash data earlier see the `rehash` methods below.

EXPORT

The method `mimetype` is exported by default. The methods `inodetype`, `globs`, `extensions`, `describe`, `mimetype_canon` and `mimetype_isa` can be exported on demand.

METHODS

`new()`

Simple constructor to allow Object Oriented use of this module. If you want to use this, include the package as `use File::MimeInfo ();` to avoid importing sub `mimetype()`.

`mimetype($file)`

Returns a mimetype string for `$file`, returns undef on failure.

This method bundles `inodetype` and `globs`.

If these methods are unsuccessful the file is read and the mimetype defaults to 'text/plain' or to 'application/octet−stream' when the first ten chars of the file match ascii control chars (white spaces excluded). If the file doesn't exist or isn't readable `undef` is returned.

`inodetype($file)`

Returns a mimetype in the 'inode' namespace or undef when the file is actually a normal file.

`globs($file)`

Returns a mimetype string for `$file` based on the filename and filename extensions. Returns undef on failure. The file doesn't need to exist.

Behaviour in list context (wantarray) is unspecified and will change in future releases.

`default($file)`

This method decides whether a file is binary or plain text by looking at the first few bytes in the file. Used to decide between "text/plain" and "application/octet−stream" if all other methods have failed.

The spec states that we should check for the ascii control chars and let higher bit chars pass to allow utf8. We try to be more intelligent using perl utf8 support.

`extensions($mimetype)`

In list context, returns the list of filename extensions that map to the given mimetype. In scalar context, returns the first extension that is found in the database for this mimetype.

`describe($mimetype, $lang)`

Returns a description of this mimetype as supplied by the mime info database. You can specify a language with the optional parameter `$lang`, this should be the two letter language code used in the xml files. Also you can set the global variable `$File::MimeInfo::LANG` to specify a language.

This method returns undef when no xml file was found (i.e. the mimetype doesn't exist in the database). It returns an empty string when the xml file doesn't contain a description in the language you specified.

Currently no real xml parsing is done, it trusts the xml files are nicely formatted.

`mimetype_canon($mimetype)`
: Returns the canonical mimetype for a given mimetype. Deprecated mimetypes are typically aliased to their canonical variants. This method only checks aliases, doesn't check whether the mimetype exists.

Use this method as a filter when you take a mimetype as input.

`mimetype_isa($mimetype)`
`mimetype_isa($mimetype, $mimetype)`
: When give only one argument this method returns a list with mimetypes that are parent classes for this mimetype.

When given two arguments returns true if the second mimetype is a parent class of the first one.

This method checks the subclasses table and applies a few rules for implicit subclasses.

`rehash()`
: Rehash the data files. Glob information is preparsed when this method is called.

If you want to by-pass the XDG basedir system you can specify your database directories by setting `@File::MimeInfo::DIRS`. But normally it is better to change the XDG basedir environment variables.

`rehash_aliases()`
: Rehashes the *mime/aliases* files.

`rehash_subclasses()`
: Rehashes the *mime/subclasses* files.

DIAGNOSTICS

This module throws an exception when it can't find any data files, when it can't open a data file it found for reading or when a subroutine doesn't get enough arguments. In the first case you either don't have the freedesktop mime info database installed, or your environment variables point to the wrong places, in the second case you have the database installed, but it is broken (the mime info database should logically be world readable).

TODO

Make an option for using some caching mechanism to reduce init time.

Make `describe()` use real xml parsing ?

LIMITATIONS

Perl versions prior to 5.8.0 do not have the ':utf8' IO Layer, thus for the default method and for reading the xml files utf8 is not supported for these versions.

Since it is not possible to distinguish between encoding types (utf8, latin1, latin2 etc.) in a straightforward manner only utf8 is supported (because the spec recommends this).

This module does not yet check extended attributes for a mimetype. Patches for this are very welcome.

AUTHOR

Jaap Karssenberg <pardus@cpan.org> Maintained by Michiel Beijen <michiel.beijen@gmail.com>

COPYRIGHT

Copyright (c) 2003, 2012 Jaap G Karssenberg. All rights reserved. This program is free software; you can redistribute it and/or modify it under the same terms as Perl itself.

SEE ALSO

File::BaseDir, File::MimeInfo::Magic, File::MimeInfo::Applications, File::MimeInfo::Rox

related CPAN modules
: File::MMagic

freedesktop specifications used
: <http://www.freedesktop.org/wiki/Specifications/shared−mime−info−spec>, <http://www.freedesktop.org/wiki/Specifications/basedir−spec>, <http://www.freedesktop.org/wiki/Specifications/desktop−entry−spec>

freedesktop mime database
: <http://www.freedesktop.org/wiki/Software/shared−mime−info>