

**NAME**

File::MimeInfo::Cookbook – various code snippets

**DESCRIPTION**

Some code snippets for non-basic uses of the File::MimeInfo module:

**Matching an extension**

A file does not have to actually exist in order to get a mimetype for it. This means that the following will work:

```
my $extension = '*.txt';
my $mimetype = mimetype( $extension );
```

**Mimotyping an scalar**

If you want to find the mimetype of a scalar value you need magic mimotyping; after all a scalar doesn't have a filename or inode. What you need to do is to use IO::Scalar :

```
use File::MimeInfo::Magic;
use IO::Scalar;

my $io_scalar = new IO::Scalar \$data;
my $mimetype = mimetype( $io_scalar );
```

In fact most other IO:: will work as long as they support the seek() and read() methods. Of course if you want really obscure things to happen you can always write your own IO object and feed it in there.

Be aware that when using a filehandle like this you need to set the :utf8 binmode yourself if appropriate.

**Mimotyping a filehandle**

Regrettably for non-seekable filehandles like STDIN simply using an IO:: object will not work. You will need to buffer enough of the data for a proper mimotyping. For example you could mimetype data from STDIN like this:

```
use File::MimeInfo::Magic;
use IO::Scalar;

my $data;
read(STDIN, $data, $File::MimeInfo::Magic::max_buffer);
my $io_scalar = new IO::Scalar \$data;
my $mimetype = mimetype( $io_scalar );
```

Be aware that when using a filehandle like this you need to set the :utf8 binmode yourself if appropriate.

**Creating a new filename**

Say you have a temporary file that you want to save with a more proper filename.

```
use File::MimeInfo::Magic qw#mimetype extensions#;
use File::Copy;

my $tmpfile = '/tmp/foo';
my $mimetype = mimetype($tmpfile);
my $extension = extensions($mimetype);
my $newfile = 'untitled1';
$newfile .= ".$extension" if length $extension;
move($tmpfile, $newfile);
```

**Force the use of a certain database directory**

Normally you just need to add the dir where your mime database lives to either the XDG\_DATA\_HOME or XDG\_DATA\_DIRS environment variables for it to be found. But in some rare cases you may want to

by-pass this system all together. Try one of the following:

```
@File::MimeInfo::DIRS = ('/home/me/share/mime');  
eval 'use File::MimeInfo';  
die if $@;
```

or:

```
use File::MimeInfo;  
@File::MimeInfo::DIRS = ('/home/me/share/mime');  
File::MimeInfo->rehash();
```

This can also be used for switching between databases at run time while leaving other XDG configuration stuff alone.

## AUTHOR

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

## COPYRIGHT

Copyright (c) 2005, 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::MimeInfo