NAME

Archive::Extract - A generic archive extracting mechanism

SYNOPSIS

```
use Archive::Extract;

### build an Archive::Extract object ###
my $ae = Archive::Extract->new( archive => 'foo.tgz' );

### extract to cwd() ###
my $ok = $ae->extract;

### extract to /tmp ###
my $ok = $ae->extract( to => '/tmp' );

### what if something went wrong?
my $ok = $ae->extract or die $ae->error;

### files from the archive ###
my $files   = $ae->files;

### dir that was extracted to ###
my $outdir  = $ae->extract_path;

### quick check methods ###
$ae->is_tar     # is it a .tar file?
$ae->is_tgz     # is it a .tar.gz or .tgz file?
$ae->is_gz;     # is it a .gz file?
$ae->is_zip;    # is it a .zip file?
$ae->is_bz2;    # is it a .bz2 file?
$ae->is_tbz;    # is it a .tar.bz2 or .tbz file?

### absolute path to the archive you provided ###
$ae->archive;

### commandline tools, if found ###
$ae->bin_tar     # path to /bin/tar, if found
$ae->bin_gzip    # path to /bin/gzip, if found
$ae->bin_unzip   # path to /bin/unzip, if found
$ae->bin_bunzip2 # path to /bin/bunzip2 if found
```

DESCRIPTION

Archive::Extract is a generic archive extraction mechanism.

It allows you to extract any archive file of the type .tar, .tar.gz, .gz, .Z, tar.bz2, .tbz, .bz2 or .zip without having to worry how it does so, or use different interfaces for each type by using either perl modules, or commandline tools on your system.

See the `HOW IT WORKS` section further down for details.

METHODS

$ae = Archive::Extract->new(archive => '/path/to/archive',[type => TYPE])

Creates a new `Archive::Extract` object based on the archive file you passed it. Automatically determines the type of archive based on the extension, but you can override that by explicitly providing the `type` argument.

Valid values for `type` are:

tar

Standard tar files, as produced by, for example, `/bin/tar`. Corresponds to a `.tar` suffix.

tgz

Gzip compressed tar files, as produced by, for example `/bin/tar -z`. Corresponds to a `.tgz` or `.tar.gz` suffix.

gz

Gzip compressed file, as produced by, for example `/bin/gzip`. Corresponds to a `.gz` suffix.

Z

Lempel-Ziv compressed file, as produced by, for example `/bin/compress`. Corresponds to a `.Z` suffix.

zip

Zip compressed file, as produced by, for example `/bin/zip`. Corresponds to a `.zip`, `.jar` or `.par` suffix.

bz2

Bzip2 compressed file, as produced by, for example, `/bin/bzip2`. Corresponds to a `.bz2` suffix.

tbz

Bzip2 compressed tar file, as produced by, for exmample `/bin/tar -j`. Corresponds to a `.tbz` or `.tar.bz2` suffix.

Returns a `Archive::Extract` object on success, or false on failure.

$ae->extract([to => '/output/path'])

Extracts the archive represented by the `Archive::Extract` object to the path of your choice as specified by the `to` argument. Defaults to `cwd()`.

Since `.gz` files never hold a directory, but only a single file; if the `to` argument is an existing directory, the file is extracted there, with it's `.gz` suffix stripped. If the `to` argument is not an existing directory, the `to` argument is understood to be a filename, if the archive type is `gz`. In the case that you did not specify a `to` argument, the output file will be the name of the archive file, stripped from it's `.gz` suffix, in the current working directory.

`extract` will try a pure perl solution first, and then fall back to commandline tools if they are available. See the `GLOBAL VARIABLES` section below on how to alter this behaviour.

It will return true on success, and false on failure.

On success, it will also set the follow attributes in the object:

$ae->extract_path

This is the directory that the files where extracted to.

$ae->files

This is an array ref with the paths of all the files in the archive, relative to the `to` argument you specified. To get the full path to an extracted file, you would use:

```
File::Spec->catfile( $to, $ae->files->[0] );
```

Note that all files from a tar archive will be in unix format, as per the tar specification.

ACCESSORS

$ae->error([BOOL])

Returns the last encountered error as string. Pass it a true value to get the `Carp::longmess()` output instead.

$ae->extract_path

This is the directory the archive got extracted to. See `extract()` for details.

$ae->files

This is an array ref holding all the paths from the archive. See `extract()` for details.

$ae->archive

This is the full path to the archive file represented by this `Archive::Extract` object.

$ae->type

This is the type of archive represented by this `Archive::Extract` object. See accessors below for an easier way to use this. See the `new()` method for details.

$ae->types

Returns a list of all known `types` for `Archive::Extract`'s `new` method.

$ae->is_tgz

Returns true if the file is of type `.tar.gz`. See the `new()` method for details.

$ae->is_tar

Returns true if the file is of type `.tar`. See the `new()` method for details.

$ae->is_gz

Returns true if the file is of type `.gz`. See the `new()` method for details.

$ae->is_Z

Returns true if the file is of type `.Z`. See the `new()` method for details.

$ae->is_zip

Returns true if the file is of type `.zip`. See the `new()` method for details.

$ae->bin_tar

Returns the full path to your tar binary, if found.

$ae->bin_gzip

Returns the full path to your gzip binary, if found

$ae->bin_unzip

Returns the full path to your unzip binary, if found

HOW IT WORKS

`Archive::Extract` tries first to determine what type of archive you are passing it, by inspecting its suffix. It does not do this by using Mime magic, or something related. See `CAVEATS` below.

Once it has determined the file type, it knows which extraction methods it can use on the archive. It will try a perl solution first, then fall back to a commandline tool if that fails. If that also fails, it will return false, indicating it was unable to extract the archive. See the section on `GLOBAL VARIABLES`

to see how to alter this order.

CAVEATS

File Extensions

`Archive::Extract` trusts on the extension of the archive to determine what type it is, and what extractor methods therefore can be used. If your archives do not have any of the extensions as described in the `new()` method, you will have to specify the type explicitly, or `Archive::Extract` will not be able to extract the archive for you.

Supporting Very Large Files

`Archive::Extract` can use either pure perl modules or command line programs under the hood. Some of the pure perl modules (like `Archive::Tar` take the entire contents of the archive into memory, which may not be feasible on your system. Consider setting the global variable `$Archive::Extract::PREFER_BIN` to `1`, which will prefer the use of command line programs and won't consume so much memory.

See the `GLOBAL VARIABLES` section below for details.

Bunzip2 support of arbitrary extensions.

Older versions of `/bin/bunzip2` do not support arbitrary file extensions and insist on a `.bz2` suffix. Although we do our best to guard against this, if you experience a bunzip2 error, it may be related to this. For details, please see the `have_old_bunzip2` method.

GLOBAL VARIABLES

$Archive::Extract::DEBUG

Set this variable to `true` to have all calls to command line tools be printed out, including all their output. This also enables `Carp::longmess` errors, instead of the regular `carp` errors.

Good for tracking down why things don't work with your particular setup.

Defaults to `false`.

$Archive::Extract::WARN

This variable controls whether errors encountered internally by `Archive::Extract` should be `carp` 'd or not.

Set to false to silence warnings. Inspect the output of the `error()` method manually to see what went wrong.

Defaults to `true`.

$Archive::Extract::PREFER_BIN

This variables controls whether `Archive::Extract` should prefer the use of perl modules, or commandline tools to extract archives.

Set to `true` to have `Archive::Extract` prefer commandline tools.

Defaults to `false`.

TODO

Mime magic support

Maybe this module should use something like `File::Type` to determine the type, rather than blindly trust the suffix.

BUG REPORTS

Please report bugs or other issues to <bug-archive-extract@rt.cpan.org<gt>.

AUTHOR

This module by Jos Boumans <kane@cpan.org>.

COPYRIGHT