NAME

Archive::Tar::File - a subclass for in-memory extracted file from Archive::Tar

SYNOPSIS

```
my @items = $tar->get_files;

print $_->name, ' ', $_->size, "\n" for @items;

print $object->get_content;
$object->replace_content('new content');

$object->rename( 'new/full/path/to/file.c' );
```

DESCRIPTION

Archive::Tar::Files provides a neat little object layer for in-memory extracted files. It's mostly used internally in Archive::Tar to tidy up the code, but there's no reason users shouldn't use this API as well.

Accessors

A lot of the methods in this package are accessors to the various fields in the tar header:

name

The file's name

mode

The file's mode

uid

The user id owning the file

gid

The group id owning the file

size

File size in bytes

mtime

Modification time. Adjusted to mac-time on MacOS if required

chksum

Checksum field for the tar header

type

File type -- numeric, but comparable to exported constants -- see Archive::Tar's documentation

linkname

If the file is a symlink, the file it's pointing to

magic

Tar magic string -- not useful for most users

version

Tar version string -- not useful for most users

uname

The user name that owns the file

gname

The group name that owns the file

devmajor

Device major number in case of a special file

devminor

Device minor number in case of a special file

prefix

Any directory to prefix to the extraction path, if any

raw

Raw tar header -- not useful for most users

Methods

new(file => $path)

Returns a new Archive::Tar::File object from an existing file.

Returns undef on failure.

new(data => $path, $data, $opt)

Returns a new Archive::Tar::File object from data.

`$path` defines the file name (which need not exist), `$data` the file contents, and `$opt` is a reference to a hash of attributes which may be used to override the default attributes (fields in the tar header), which are described above in the Accessors section.

Returns undef on failure.

new(chunk => $chunk)

Returns a new Archive::Tar::File object from a raw 512-byte tar archive chunk.

Returns undef on failure.

full_path

Returns the full path from the tar header; this is basically a concatenation of the `prefix` and `name` fields.

validate

Done by Archive::Tar internally when reading the tar file: validate the header against the checksum to ensure integer tar file.

Returns true on success, false on failure

has_content

Returns a boolean to indicate whether the current object has content. Some special files like directories and so on never will have any content. This method is mainly to make sure you don't get warnings for using uninitialized values when looking at an object's content.

get_content

Returns the current content for the in-memory file

get_content_by_ref

Returns the current content for the in-memory file as a scalar reference. Normal users won't need this, but it will save memory if you are dealing with very large data files in your tar archive, since it will pass the contents by reference, rather than make a copy of it first.

replace_content($content)

Replace the current content of the file with the new content. This only affects the in-memory archive, not the on-disk version until you write it.

Returns true on success, false on failure.

rename($new_name)

Rename the current file to $new_name.

Note that you must specify a Unix path for $new_name, since per tar standard, all files in the archive must be Unix paths.

Returns true on success and false on failure.

Convenience methods

To quickly check the type of a `Archive::Tar::File` object, you can use the following methods:

is_file
: Returns true if the file is of type `file`

is_dir
: Returns true if the file is of type `dir`

is_hardlink
: Returns true if the file is of type `hardlink`

is_symlink
: Returns true if the file is of type `symlink`

is_chardev
: Returns true if the file is of type `chardev`

is_blockdev
: Returns true if the file is of type `blockdev`

is_fifo
: Returns true if the file is of type `fifo`

is_socket
: Returns true if the file is of type `socket`

is_longlink
: Returns true if the file is of type `LongLink`. Should not happen after a successful `read`.

is_label
: Returns true if the file is of type `Label`. Should not happen after a successful `read`.

is_unknown
: Returns true if the file type is `unknown`