

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

Archive::Tar::File->new(file => \$path)

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

Returns undef on failure.

Archive::Tar::File->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.

Archive::Tar::File->new(chunk => \$chunk)

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

Returns undef on failure.

\$bool = \$file->extract([\$alternative_name])

Extract this object, optionally to an alternative name.

See Archive::Tar->extract_file for details.

Returns true on success and false on failure.

\$path = \$file->full_path

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

\$bool = \$file->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

\$bool = \$file->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.

\$content = \$file->get_content

Returns the current content for the in-memory file

\$cref = \$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.

\$bool = \$file->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.

\$bool = \$file->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.

\$bool = \$file->chmod \$mode)

Change mode of \$file to \$mode. The mode can be a string or a number which is interpreted as octal whether or not a leading 0 is given.

Returns true on success and false on failure.

\$bool = \$file->chown(\$user [, \$group])

Change owner of \$file to \$user. If a \$group is given that is changed as well. You can also pass a single parameter with a colon separating the use and group as in 'root:wheel'.

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:

\$file->is_file

Returns true if the file is of type file

\$file->is_dir

Returns true if the file is of type dir

\$file->is_hardlink

Returns true if the file is of type hardlink

\$file->is_symlink

Returns true if the file is of type $\operatorname{symlink}$

\$file->is chardev

Returns true if the file is of type chardev

\$file->is_blockdev

Returns true if the file is of type blockdev

\$file->is fifo



Returns true if the file is of type fifo

\$file->is_socket

Returns true if the file is of type socket

\$file->is_longlink

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

\$file->is_label

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

\$file->is_unknown

Returns true if the file type is unknown