NAME

Lintian::File::Path – Lintian representation of a path entry in a package

SYNOPSIS

```
my ($name, $type, $dir) = ('lintian', 'source', '/path/to/entry');
```

INSTANCE METHODS

init_from_tar_output

get_quoted_filename

unescape_c_style

magic(COUNT)

Returns the specified COUNT of magic bytes for the file.

get_interpreter

Returns the interpreter for the file if it is a script.

is_script

Returns true if file is a script and false otherwise.

is_control

Returns true if file is a maintainer script and false otherwise.

identity

Returns the owner and group of the path, separated by a slash.

NB: If only numerical owner information is available in the package, this may return a numerical owner (except uid 0 is always mapped to "root")

operm

Returns the file permissions of this object in octal (e.g. 0644).

NB: This is only well defined for file entries that are subject to permissions (e.g. files). Particularly, the value is not well defined for symlinks.

children

Returns a list of children (as Lintian::File::Path objects) of this entry. The list and its contents should not be modified.

Only returns direct children of this directory. The entries are sorted by name.

NB: Returns the empty list for non-dir entries.

descendants

Returns a list of children (as Lintian::File::Path objects) of this entry. The list and its contents should not be modified.

Descends recursively into subdirectories and return the descendants in breadth-first order. Children of a given directory will be sorted by name.

NB: Returns the empty list for non-dir entries.

timestamp

Returns a Unix timestamp for the given path. This is a number of seconds since the start of Unix epoch in UTC.

child(BASENAME)

Returns the child named BASENAME if it is a child of this directory. Otherwise, this method returns `undef`.

Even for directories, BASENAME should not end with a slash.

When invoked on non-dirs, this method always returns `undef`.

Example:

```
$dir_entry->child('foo') => $entry OR undef
```

is_symlink
: Returns a truth value if this entry is a symlink.

is_hardlink
: Returns a truth value if this entry is a hardlink to a regular file.

  NB: The target of a hardlink is always a regular file (and not a dir etc.).

is_dir
: Returns a truth value if this entry is a dir.

  NB: Unlike the "-d `$dir`" operator this will never return true for symlinks, even if the symlink points to a dir.

is_file
: Returns a truth value if this entry is a regular file (or a hardlink to one).

  NB: Unlike the "-f `$file`" operator this will never return true for symlinks, even if the symlink points to a file (or hardlink).

is_regular_file
: Returns a truth value if this entry is a regular file.

  This is eqv. to `$path->is_file` and not `$path->is_hardlink`.

  NB: Unlike the "-f `$file`" operator this will never return true for symlinks, even if the symlink points to a file.

link_normalized
: Returns the target of the link normalized against it's directory name. If the link cannot be normalized or normalized path might escape the package root, this method returns `undef`.

  NB: This method will return the empty string for links pointing to the root dir of the package.

  Only available on "links" (i.e. symlinks or hardlinks). On non-links this will croak.

  Symlinks only: If you want the symlink target as a Lintian::File::Path object, use the resolve_path method with no arguments instead.

is_readable
: Returns a truth value if the permission bits of this entry have at least one bit denoting readability set (bitmask 0444).

is_writable
: Returns a truth value if the permission bits of this entry have at least one bit denoting writability set (bitmask 0222).

is_executable
: Returns a truth value if the permission bits of this entry have at least one bit denoting executability set (bitmask 0111).

unpacked_path
: Returns the path to this object on the file system, which must be a regular file, a hardlink or a directory.

  This method may fail if:

  - The object is neither a directory or a file-like object (e.g. a named pipe).
  - If the object is dangling symlink or the path traverses a symlink outside the package root.

  To test if this is safe to call, if the target is (supposed) to be a:

  - file or hardlink then test with "is_open_ok".

- dir then assert resolve_path returns a defined entry, for which ''is_dir'' returns a truth value.

is_open_ok
: Returns a truth value if it is safe to attempt open a read handle to the underlying file object.

  Returns a truth value if the path may be opened.

slurp
: Return the file contents as a scalar.

  This method may fail for the same reasons as ''open([LAYER])''.

follow
: Return dereferenced link if applicable

resolve_path([PATH])
: Resolve PATH relative to this path entry.

  If PATH starts with a slash and the file hierarchy has a well-defined root directory, then PATH will instead be resolved relatively to the root dir. If the file hierarchy does not have a well-defined root dir (e.g. for source packages), this method will return `undef`.

  If PATH is omitted, then the entry is resolved and the target is returned if it is valid. Except for symlinks, all entries always resolve to themselves. NB: hardlinks also resolve as themselves.

  It is an error to attempt to resolve a PATH against a non-directory and non-symlink entry – as such resolution would always fail (i.e. foo/../bar is an invalid path unless foo is a directory or a symlink to a dir).

  The resolution takes symlinks into account and following them provided that the target path is valid (and can be followed safely). If the path is invalid or circular (symlinks), escapes the root directory or follows an unsafe symlink, the method returns `undef`. Otherwise, it returns the path entry that denotes the target path.

  If PATH contains at least one path segment and ends with a slash, then the resolved path will end in a directory (or fail). Otherwise, the resolved PATH can end in any entry *except* a symlink.

  Examples:

```
$symlink_entry->resolve_path => $nonsymlink_entry OR undef

$x->resolve_path => $x

For directory or symlink entries (dol), you can also resolve a path:

$dol_entry->resolve_path('some/../where') => $nonsymlink_entry OR undef

# Note the trailing slash
$dol_entry->resolve_path('some/../where/') => $dir_entry OR undef
```

name
: Returns the name of the file (relative to the package root).

  NB: It will never have any leading ''./'' (or ''/'') in it.

basename
: Returns the ''filename'' part of the name, similar **basename** (1) or File::Basename::basename (without passing a suffix to strip in either case).

  NB: Returns the empty string for the ''root'' dir.

dirname
: Returns the ''directory'' part of the name, similar to **dirname** (1) or File::Basename::dirname. The dirname will end with a trailing slash (except the ''root'' dir – see below).

NB: Returns the empty string for the ‘‘root’’ dir.

link
: If this is a link (i.e. is_symlink or is_hardlink returns a truth value), this method returns the target of the link.

  If this is not a link, then this returns undef.

  If the path is a symlink this method can be used to determine if the symlink is relative or absolute. This is *not* true for hardlinks, where the link target is always relative to the root.

  NB: Even for symlinks, a leading ‘‘./’’ will be stripped.

normalized
faux
: Returns a truth value if this entry absent in the package. This can happen if a package does not include all intermediate directories.

size
: Returns the size of the path in bytes.

  NB: Only regular files can have a non-zero file size.

date
: Return the modification date as YYYY-MM-DD.

time
perm
path_info
owner
: Returns the owner of the path entry as a username.

  NB: If only numerical owner information is available in the package, this may return a numerical owner (except uid 0 is always mapped to ‘‘root’’)

group
: Returns the group of the path entry as a username.

  NB: If only numerical owner information is available in the package, this may return a numerical group (except gid 0 is always mapped to ‘‘root’’)

uid
: Returns the uid of the owner of the path entry.

  NB: If the uid is not available, 0 will be returned. This usually happens if the numerical data is not collected (e.g. in source packages)

gid
: Returns the gid of the owner of the path entry.

  NB: If the gid is not available, 0 will be returned. This usually happens if the numerical data is not collected (e.g. in source packages)

file_info
: Return the data from **file** (1) if it has been collected.

  Note this is only defined for files as Lintian only runs **file** (1) on files.

java_info
script
strings
objdump
control
`basedir`
index

parent_dir
child_table
sorted_children
: Returns the parent directory entry of this entry as a Lintian::File::Path.

  NB: Returns `undef` for the "root" dir.

`childnames`
parent_dir
: Return the parent dir entry of this the path entry.

dereferenced

AUTHOR

Originally written by Niels Thykier <niels@thykier.net> for Lintian.

SEE ALSO

lintian (1)