NAME

Lintian::Collect::Package – Lintian base interface to binary and source package data collection

SYNOPSIS

```
use autodie;
use Lintian::Collect;

my ($name, $type, $dir) = ('foobar', 'source', '/path/to/lab-entry');
my $info = Lintian::Collect->new ($name, $type, $dir);
my $filename = "etc/conf.d/$name.conf";
my $file = $info->index_resolved_path($filename);
if ($file and $file->is_open_ok) {
    my $fd = $info->open;
    # Use $fd ...
    close($fd);
} elsif ($file) {
    print "$file is available, but is not a file or unsafe to open\n";
} else {
    print "$file is missing\n";
}
```

DESCRIPTION

Lintian::Collect::Package provides part of an interface to package data for source and binary packages. It implements data collection methods specific to all packages that can be unpacked (or can contain files)

This module is in its infancy. Most of Lintian still reads all data from files in the laboratory whenever that data is needed and generates that data via collect scripts. The goal is to eventually access all data about source packages via this module so that the module can cache data where appropriate and possibly retire collect scripts in favor of caching that data in memory.

INSTANCE METHODS

In addition to the instance methods listed below, all instance methods documented in the Lintian::Collect module are also available.

unpacked ([FILE])

Returns the path to the directory in which the package has been unpacked. FILE must be either a Lintian::Path object (>= 2.5.13~) or a string denoting the requested path. In the latter case, the path must be relative to the root of the package and should be normalized.

It is not permitted for FILE to be `undef`. If the "root" dir is desired either invoke this method without any arguments at all, pass it the correct Lintian::Path or the empty string.

If FILE is not in the package, it returns the path to a non-existent file entry.

The path returned is not guaranteed to be inside the Lintian Lab as the package may have been unpacked outside the Lab (e.g. as optimization).

Caveat with symlinks: Package is extracted as is and the path returned by this method points to the extracted file object. If this is a symlink, it may "escape the root" and point to a file outside the lab (and a path traversal).

The following code may be helpful in checking for path traversal:

```
use Lintian::Util qw(is_ancestor_of);

my $collect = ... ;
my $file = '../../../etc/passwd';
my $uroot = $collect->unpacked;
my $ufile = $collect->unpacked($file);
# $uroot will exist, but $ufile might not.
```

```
if ( -e $ufile && is_ancestor_of($uroot, $ufile)) {
   # has not escaped $uroot
   do_stuff($ufile);
} elsif ( -e $ufile) {
   # escaped $uroot
   die "Possibly path traversal ($file)";
} else {
   # Does not exists
}
```

Alternatively one can use normalize_pkg_path in Lintian::Util or link_normalized.

To get a list of entries in the package or the file meta data of the entries (as path objects), see ''sorted_index'' and ''index (FILE)''.

Needs-Info requirements for using *unpacked*: unpacked

file_info (FILE)
Returns the output of **file** (1) for FILE (if it exists) or `undef`.

NB: The value may have been calibrated by Lintian. A notorious example is gzip files, where **file** (1) can be unreliable at times (see #620289)

Needs-Info requirements for using *file_info*: file-info

md5sums
Returns a hashref mapping a FILE to its md5sum. The md5sum is computed by Lintian during extraction and is not guaranteed to match the md5sum in the ''md5sums'' control file.

Needs-Info requirements for using *md5sums*: md5sums

index (FILE)
Returns a path object to FILE in the package. FILE must be relative to the root of the unpacked package and must be without leading slash (or ''./''). If FILE is not in the package, it returns `undef`. If FILE is supposed to be a directory, it must be given with a trailing slash. Example:

```
my $file = $info->index ("usr/bin/lintian");
my $dir = $info->index ("usr/bin/");
```

To get a list of entries in the package, see ''sorted_index''. To actually access the underlying file (e.g. the contents), use ''unpacked ([FILE])''.

Note that the ''root directory'' (denoted by the empty string) will always be present, even if the underlying tarball omits it.

Needs-Info requirements for using *index*: unpacked

sorted_index
Returns a sorted array of file names listed in the package. The names will not have a leading slash (or ''./'') and can be passed to ''unpacked ([FILE])'' or ''index (FILE)'' as is.

The array will not contain the entry for the ''root'' of the package.

NB: For source packages, please see the ''index''-caveat.

Needs-Info requirements for using *sorted_index*: Same as index

index_resolved_path(PATH)
Resolve PATH (relative to the root of the package) and return the entry denoting the resolved path.

The resolution is done using resolve_path.

NB: For source packages, please see the ''index''-caveat.

Needs-Info requirements for using *index_resolved_path*: Same as index

AUTHOR

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

SEE ALSO

lintian (1), Lintian::Collect, Lintian::Collect::Binary, Lintian::Collect::Source