NAME

Lintian::Unpacker −− Job handler to unpack collections

SYNOPSIS

```
use Lintian::DepMap::Properties;
use Lintian::Unpacker;

my $done = 1;
my $joblimit = 4;
my $collmap = Lintian::DepMap::Properties->new;
my %requested = ( 'debfiles' => 1 );
# Initialise $collmap with the collections and their relations
# - Each node in $collmap should an instance of L::CollScript
#   as property.
my $unpacker = Lintian::Unpacker->new ($collmap, \%requested,
                                       $joblimit);

while (1) {
    my $errhandler = sub {}; # Insert hook
    my @lpkgs; # List of Lintian::Lab::Entry instances
    $unpacker->reset_worklist;
    next unless $unpacker->prepare_tasks ($errhandler, @lpkgs);

    my %hooks = (
        'coll-hook' => sub {}, # Insert hook
        'finish-hook' => sub {}, # Insert hook
    );
    $unpacker->process_tasks ();
    last if $done;
}
```

DESCRIPTION

An unpacker class to extract data from lab entries and make it available via Lintian::Collect.

CLASS METHODS

new (COLLMAP, PROFILE[, OPTIONS])

Creates a new unpacker.

COLLMAP is a Lintian::DepMap::Properties describing the dependencies between the collections. Each node in COLLMAP must have a Lintian::CollScript as property.

OPTIONS is an optional hashref containing optional configurations. If a key is not present, its value is assumed to be `undef` unless otherwise stated. The following key/values are available:

"profile"

If this key is present and its value is defined, the value must be Lintian::Profile. The unpacker will use the enabled checks of the Profile to determine what collections to use.

If "profile" is not present or its value is undefined, then all collections in COLLMAP will be unpacked.

"extra-coll"

If this key is present and its value is defined, it must be a reference to a hash table. The keys are considered names of "extra" collections to unpack. The values in this table is ignored.

Extra collections will be unpacked on top of other collections.

NB: This value is ignored if "profile" is not given.

"jobs"
: This value is the max number of jobs to be run in parallel. Can be changed with the "jobs" method later. If omitted, it defaults to 0. Refer to "jobs" for more info.

INSTANCE METHODS

prepare_tasks (ERRHANDLER, LAB-ENTRY...)

Prepare a number of lab entries for unpacking.

The ERRHANDLER should be a code ref, which will be invoked in case that an entry is not in the laboratory and cannot be created (via the create method). It is invoked once per failed entry giving the entry as first (and only) argument.

If ERRHANDLER returns normally, the entry is skipped (and will not be unpacked later). If ERRHANDLER croaks/dies/etc., the method will attempt to update the status file for any entry it created before passing back the error to the caller (via die).

LAB-ENTRY is an array of lab entries to be processed. They must be instances of Lintian::Lab::Entry, but do not have to exists. They will be created as needed.

Returns a truth value if at least one entry needs to be processed and it did not cause an error. Otherwise, it returns `undef`.

NB: The status file is not updated for created entries on successful return. It should either be done by running the process_tasks method or manually.

process_tasks (HOOKS)

Process the current tasks. This method blocks until all tasks and jobs have terminated.

The return value is unspecified.

HOOKS (if given) is a hashref of hooks. The following hooks are available:

coll-hook (LPKG, EVENT, COLL, TASK_ID[, STATUS_OR_ERROR_MSG])

Called each time a new collection job is started or finished.

LPKG is the entry it is applied to. COLL is the collection being applied. EVENT is either "start" for a new job, "start-failed" for a job that failed to start (appears instead of a "start" event) or "finish" for a job terminating.

TASK_ID is the task id of the job (a string).

If the event is "finish", then STATUS_OR_ERROR_MSG is the exit code of the job (non-zero being an error). If the event is "start-failed", it is an error message explaining why the job failed to start. It is not defined for other events.

reset_worklist

Wait for all running jobs (see "wait_for_jobs") and discard the current worklist.

wait_for_jobs

Block and wait for all running jobs to terminate. Usually this is not needed unless process_tasks was interrupted somehow.

kill_jobs

Forcefully terminate all running jobs. Usually this is not needed unless process_tasks was interrupted somehow.

jobs

Returns or sets the max number of jobs to be processed in parallel.

If the limit is 0, then there is no limit for the number of parallel jobs.

AUTHOR

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

SEE ALSO

lintian (1), **Lintian::CollScript** (3), **Lintian::Lab::Entry** (3)