NAME

Test::StagedFileProducer −− mtime−based file production engine

SYNOPSIS

```
use Test::StagedFileProducer;

my $wherever = '/your/test/directory';

my $producer = Test::StagedFileProducer->new(path => $wherever);
$producer->exclude("$wherever/log", "$wherever/build-stamp");

my $output = "$wherever/file.out";
$producer->add_stage(
      products => [$output],
      build =>sub {
          printf "Building $output.\n";
      },
      skip =>sub {
          printf "Skipping $output.\n";
      }
);

$producer->run(minimum_epoch => time, verbose => 1);
```

DESCRIPTION

Provides a way to define and stack file production stages that all depend on subsets of the same group of files.

After the stages are defined, the processing engine takes an inventory of all files in a target directory. It excludes some files, like logs, that should not be considered.

Each stage adds its own products to the list of files to be excluded before deciding whether to produce them. The decision is based on relative file modification times, in addition to a systemic rebuilding threshold. Before rebuilding, each stage asks a lower stage to make the same determination.

The result is an engine with file production stages that depend on successively larger sets of files.

FUNCTIONS

new(path => PATH)

Create a new instance focused on files in directory PATH.

exclude(LIST)

Excludes all absolute paths in LIST from all mtime comparisons. This is especially useful for logs. Calls to Path::Tiny−>realpath are made to ensure the elements are canonical and have a chance of matching something returned by File::Find::Rule.

add_stage(HASH)

Add a stage defined by HASH to the processing engine for processing after stages previously added. HASH can define the following keys:

$HASH{products} => LIST; a list of full-path filenames to be produced.

$HASH{minimum_epoch} => EPOCH; an integer threshold for maximum age

$HASH{build} => SUB; a sub executed when production is required.

$HASH{skip} => SUB; a sub executed when production is not required.

run(PARAMETERS)

Runs the defined engine using the given parameters, which are arranged in a matching list suitable for assignment to a hash. The following two parameters are currently available:

minimum_epoch => EPOCH; a systemic threshold, in epochs, below which rebuilding is mandatory for any product.

verbose => BOOLEAN; an option to enable more verbose reporting

_process_remaining_stages(LIST)

An internal subroutine that is used recursively to execute the stages. The list passed describes the list of files to be excluded from subsequent mtime calculations.

Please note that the bulk of the execution takes place after calling the next lower stage. That is to ensure that any lower build targets (or products, in our parlance) are met before the present stage attempts to do its job.