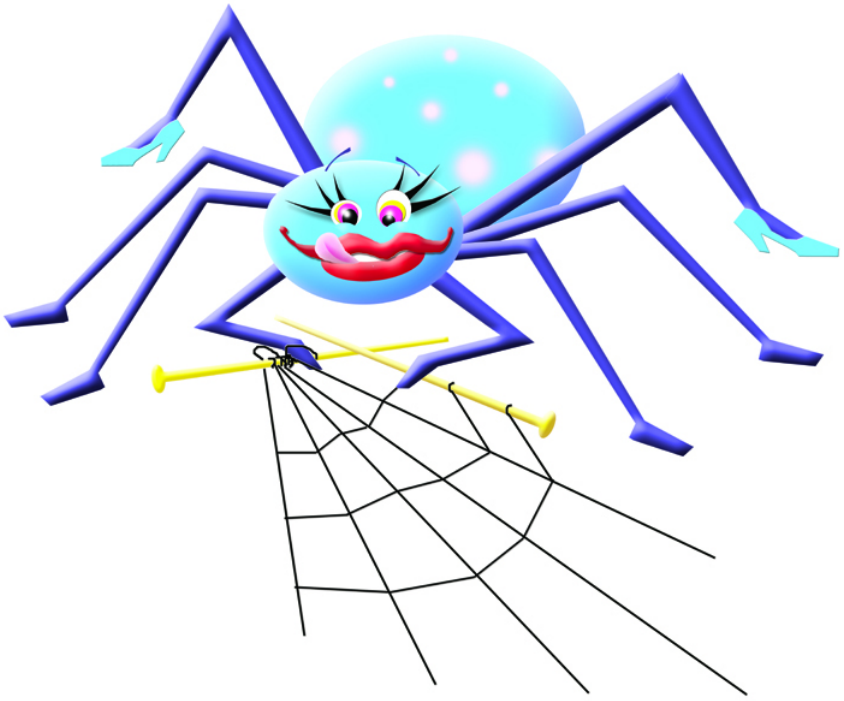
Nancy, the lazy weaver



User's guide

Reuben Thomas

2nd April 2016

1 Introduction

Nancy is a simple macro processor that weaves files together from other files and the output of programs.

2 Installation

Nancy is written in Perl, and requires version 5.10 or later, the `File::Slurp` module, and the `RRT` modules (available from `https://github.com/rrthomas/perl`.

3 Invocation

Nancy takes two arguments:

```
nancy [OPTION...] TEMPLATE PATH
```

where `TEMPLATE` is name of the template file, and `PATH` is the path of the file or directory to weave.

The following command-line `OPTION`s may be given:

`--root DIRECTORY` Set the source root directory (the default is the current directory).

`--list-files` List on standard error the files used to make each page.

`--version` Show the version number of Nancy.

`--help` Show help on how to run Nancy.

The options may be abbreviated to any unambiguous prefix.

4 Operation

Nancy weaves a path given a template as follows:

1. Set the initial text to `$include{TEMPLATE}`, unless `TEMPLATE` is `-`, in which case set the initial text to the contents of standard input.
2. Scan the text for commands. Weave any arguments to the command, run each command, and replace the command by the result.
3. Output the resultant text, eliding any final newline. (This last part may look tricky, but it almost always does what you want, and makes `$include` behave better in various contexts.)

A command takes the form `$COMMAND` or `$COMMAND{ARGUMENT, ...}`.

Nancy recognises these commands:

`$include{FILE, ARGUMENT, ...}` Look up the given file. If it is executable, run it as a command with the given arguments and collect the output. Otherwise, read the contents of the given file. Weave and return the result.

`$paste{FILE, ARGUMENT, ...}` Like `$include`, but does not weave its result before returning it.

`$path` Return the `PATH` argument.

`$root` Return the root directory.

`$template` Return the `TEMPLATE` argument.

The last three commands are mostly useful as arguments to `$include`.

Only one guarantee is made about the order in which commands are processed: if one command is nested inside another, the inner command will be processed first. (The order only matters for `$include` commands that run executables; if you nest them, you have to deal with this potential pitfall.)

To find the file `FILE` specified by a `$include` command, Nancy proceeds thus:

1. See whether `ROOT/PATH/FILE` is a file (or a symbolic link to a file). If so, return the file path.
2. If not, remove the last directory from `PATH` and try again, until `PATH` is empty.
3. Try looking for `ROOT/FILE`.
4. Try looking for file on the user's `PATH` (the list of directories specified by the `PATH` environment variable).
5. If no file is found, Nancy stops with an error message.

For example, if the root directory is `/dir`, `PATH` is `foo/bar/baz`, and Nancy is trying to find `file.html`, it will try the following , in order:

1. `/dir/foo/bar/baz/file.html`
2. `/dir/foo/bar/file.html`
3. `/dir/foo/file.html`
4. `/dir/file.html`

5 Example: generating a web site

Suppose a web site has the following page design, from top to bottom: logo, navigation menu, breadcrumb trail, page body.

Most of the elements are the same on each page, but the breadcrumb trail has to show the canonical path to each page, and the logo is bigger on the home page, which is the default `index.html`.

Suppose further that the web site has the following structure, where each line corresponds to a page:

- Home page

- People
  - Jo Bloggs
  - Hilary Pilary
  - ...
- Places
  - Vladivostok
  - Timbuktu
  - ...

The basic page template looks something like this:

```
<html>
  <link href="style.css" rel="stylesheet" type="text/css">
  <title>$include{title}</title>
  <body>
    <div class="logo">$include{logo.html}</div>
    <div class="menu">$include{menu.html}</div>
    <div class="breadcrumb">$include{breadcrumb.html}</div>
    <div class="main">$include{main.html}</div>
  </body>
</html>
```

Making the menu an included file is not strictly necessary, but makes the template easier to read. The pages will be laid out as follows:

- `/`
  - `index.html`
  - `people/`
    - `index.html`
    - `jo_bloggs.html`
    - `hilary_pilary.html`
  - `places/`
    - `index.html`
    - `vladivostok.html`
    - `timbuktu.html`

The corresponding source files will be laid out as follows. This may look a little confusing at first, but note the similarity to the HTML pages, and hold on for the explanation!

- `source/`
  - `template.html` (the template shown above)

- menu.html
- logo.html
- breadcrumb.html
- index.html/
    * main.html
    * logo.html
- people/
    * breadcrumb.html
    * index.html/
        · main.html
    * jo_bloggs.html/
        · main.html
    * hilary_pilary.html/
        · main.html
- places/
    * breadcrumb.html
    * index.html/
        · main.html
    * vladivostok.html/
        · main.html
    * timbuktu.html/
        · main.html

Note that there is only one menu fragment (the main menu is the same for every page), while each section has its own breadcrumb trail (`breadcrumb.html`), and each page has its own content (`main.html`).

Now consider how Nancy builds the page whose URL is `vladivostok.html`. According to the rules given in section 4, Nancy will look first for files in `source/places/vladivostok.html`, then in `source/places`, and finally in `source`. Hence, the actual list of files used to assemble the page is:

- `source/template.html`
- `source/logo.html`
- `source/menu.html`
- `source/places/breadcrumb.html`
- `source/places/vladivostok.html/main.html`

For the site's index page, the file `index.html/logo.html` will be used for the logo fragment, which can refer to the larger graphic desired.

This scheme, though simple, is surprisingly flexible; this simple example has covered all the standard techniques for Nancy's use.