luaotfload.conf

Luaotfload configuration file

Date:	2021-01-08
Copyright:	GPL v2.0
Version:	3.17
Manual section:	5
Manual group:	text processing

SYNOPSIS

- **./luaotfload{.conf,rc}**
- **XDG_CONFIG_HOME/luaotfload/luaotfload{.conf,rc}**
- **~/.luaotfloadrc**

DESCRIPTION

The file `luaotfload.conf` contains configuration options for *Luaotfload*, a font loading and font management component for LuaTeX.

EXAMPLE

A small Luaotfload configuration file with few customizations could look as follows:

```
[db]
    formats = afm,ttf
    compress = false

[misc]
    termwidth = 60

[run]
    log-level = 6
```

This will make Luaotfload ignore all font files except for PostScript binary fonts with a matching AFM file, and Truetype fonts. Also, an uncompressed index file will be dumped which is going to be much larger than the default gzip'ed index. The terminal width is truncated to 60 characters which influences the verbose output during

indexing. Finally, the verbosity is increased greatly: each font file being processed will be printed to the stdout on a separate line, along with lots of other information.

To observe the difference in behavior, save above snippet to `./luaotfload.conf` and update the font index:

```
luaotfload-tool --update --force
```

The current configuration can be written to disk using **luaotfload-tool**:

```
luaotfload-tool --dumpconf > luaotfload.conf
```

The result can itself be used as a configuration file.

SYNTAX

The configuration file syntax follows the common INI format. For a more detailed description please refer to the section "CONFIGURATION FILE" of **git-config**(1). A brief list of rules is given below:

- Blank lines and lines starting with a semicolon (`;`) are ignored.
- A configuration file is partitioned into sections that are declared by specifying the section title in brackets on a separate line:

  ```
  [some-section]
  ... section content ...
  ```

- Sections consist of one or more variable assignments of the form `variable-name = value` E. g.:

  ```
  [foo]
      bar = baz
      quux = xyzzy
      ...
  ```

- Section and variable names may contain only uppercase and lowercase letters as well as dashes (`-`).

VARIABLES

Variables in belong into a configuration section and their values must be of a certain type. Some of them have further constraints. For example, the "color callback" must be a string of one of the values `post_linebreak_filter`, `pre_linebreak_filter`, or `pre_output_filter`, defined in the section *run* of the configuration file.

Currently, the configuration is organized into four sections:

db Options relating to the font index.

misc Options without a clearly defined category.

paths Path and file name settings.

run Options controlling runtime behavior of Luaotfload.

The list of valid variables, the sections they are part of and their type is given below. Types represent Lua types that the values must be convertible to; they are abbreviated as follows: `s` for the *string* type, `n` for *number*, `b` for *boolean*. A value of `nil` means the variable is unset.

Section db

variable	type	default
compress	b	true
designsize-dimen	b	bp
formats	s	"otf,ttf,ttc"
max-fonts	n	2^51
scan-local	b	false
skip-read	b	false
strip	b	true
update-live	b	true

The flag compress determines whether the font index (usually luaotfload-names.lua[.gz] will be stored in compressed forms. If unset it is equivalent of passing --no-compress to **luaotfload-tool**. Since the file is only created for convenience and has no effect on the runtime behavior of Luaotfload, the flag should remain set. Most editors come with zlib support anyways.

The setting designsize-dimen applies when looking up fonts from families with design sizes. In Opentype, these are specified as "decipoints" where one decipoint equals ten DTP style "big points". When indexing fonts these values are converted to sp. In order to treat the values as though they were specified in TeX points or Didot points, set designsize-dimen to pt or dd.

The list of formats must be a comma separated sequence of strings containing one or more of these elements:

- otf (OpenType format),
- ttf and ttc (TrueType format),
- afm (Adobe Font Metrics),

It corresponds loosely to the --formats option to **luaotfload-tool**. Invalid or duplicate members are ignored; if the list does not contain any useful identifiers, the default list "otf,ttf,ttc" will be used.

The variable max-fonts determines after processing how many font files the font scanner will terminate the search. This is useful for debugging issues with the font index and has the same effect as the option --max-fonts to **luaotfload-tools**.

The scan-local flag, if set, will incorporate the current working directory as a font search location. NB: This will potentially slow down document processing because a font index with local fonts will not be saved to disk, so these fonts will have to be re-indexed whenever the document is built.

The skip-read flag is only useful for debugging: It makes Luaotfload skip reading fonts. The font information for rebuilding the index is taken from the presently existing one.

Unsetting the strip flag prevents Luaotfload from removing data from the index that is only useful when processing font files. NB: this can increase the size of the index files significantly and has no effect on the runtime behavior.

If update-live is set, Luaotfload will reload the database if it cannot find a requested font. Those who prefer to update manually using **luaotfload-tool** should unset this flag. This option does not affect rebuilds due to version mismatch.

Section `default-features`

By default Luaotfload enables `node` mode and picks the default font features that are prescribed in the OpenType standard. This behavior may be overridden in the `default-features` section. Global defaults that will be applied for all scripts can be set via the `global` option, others by the script they are to be applied to. For example, a setting of

```
[default-features]
    global = mode=base,color=0000FF
    dflt   = smcp,onum
```

would force *base* mode, tint all fonts blue and activate small capitals and text figures globally. Features are specified as a comma separated list of variables or variable-value pairs. Variables without an explicit value are set to `true`.

Section `misc`

variable	type	default
statistics	b	`false`
termwidth	n	`nil`
version	s	<Luaotfload version>

With `statistics` enabled, extra statistics will be collected during index creation and appended to the index file. It may then be queried at the Lua end or inspected by reading the file itself.

The value of `termwidth`, if set, overrides the value retrieved by querying the properties of the terminal in which Luatex runs. This is useful if the engine runs with `shell_escape` disabled and the actual terminal dimensions cannot be retrieved.

The value of `version` is derived from the version string hard-coded in the Luaotfload source. Override at your own risk.

Section `paths`

variable	type	default
cache-dir	s	`"fonts"`
names-dir	s	`"names"`
index-file	s	`"luaotfload-names.lua"`
lookups-file	s	`"luaotfload-lookup-cache.lua"`

The paths `cache-dir` and `names-dir` determine the subdirectory inside the Luaotfload subtree of the `luatex-cache` directory where the font cache and the font index will be stored, respectively.

Inside the index directory, the names of the index file and the font lookup cache will be derived from the respective values of `index-file` and `lookups-file`. This is the filename base for the bytecode compiled version as well as – for the index – the gzipped version.

Section `run`

variable	type	default
anon-sequence	s	`"tex,path,name"`
color-callback	s	`"post_linebreak_filter"`
definer	s	`"patch"`
log-level	n	`0`
resolver	s	`"cached"`
fontloader	s	`"default"`

Unqualified font lookups are treated with the flexible "anonymous" mechanism. This involves a chain of lookups applied successively until the first one yields a match. By default, the lookup will first search for TFM fonts using the Kpathsea library. If this wasn't successful, an attempt is made at interpreting the request as an absolute path (like the `[/path/to/font/foo.ttf]` syntax) or a file name (`file:foo.ttf`). Finally, the request is interpreted as a font name and retrieved from the index (`name:Foo Regular`). This behavior can be configured by specifying a list as the value to `anon-sequence`. Available items are `tex`, `path`, `name` – representing the lookups described above, respectively –, and `file` for searching a filename but not an absolute path. Also, `my` lookups are valid values but they should only be used from within TeX documents, because there is no means of customizing a `my` lookups on the command line.

The `color-callback` option determines the stage at which fonts that defined with a `color=xxyyzz` feature will be colorized. By default this happens in a `post_linebreak_filter` but alternatively the `pre_linebreak_filter` or `pre_output_filter` may be chosen, which is faster but might produce inconsistent output. The `pre_output_filter` used to be the default in the 1.x series of Luaotfload, whilst later versions up to and including 2.5 hooked into the `pre_linebreak_filter` which naturally didn't affect any glyphs inserting during hyphenation. Both are kept around as options to restore the previous behavior if necessary.

The `definer` allows for switching the `define_font` callback. Apart from the default `patch` one may also choose the `generic` one that comes with the vanilla fontloader. Beware that this might break tools like Fontspec that rely on the `patch_font` callback provided by Luaotfload to perform important corrections on font data.

The fontloader backend can be selected by setting the value of `fontloader`. The most important choices are `default`, which will load the dedicated Luaotfload fontloader, and `reference`, the upstream package as shipped with Luaotfload. Other than those, a file name accessible via kpathsea can be specified.

Alternatively, the individual files that constitute the fontloader can be loaded directly. While less efficient, this greatly aids debugging since error messages will reference the actual line numbers of the source files and explanatory comments are not stripped. Currently, three distinct loading strategies are available: `unpackaged` will load the batch that is part of Luaotfload. These contain the identical source code that the reference fontloader has been compiled from. Another option, `context` will attempt to load the same files by their names in the Context format from the search path. Consequently this option allows to use the version of Context that comes with the TeX distribution. Distros tend to prefer the stable version ("current" in Context jargon) of those files so certain bugs encountered in the more bleeding edge Luaotfload can be avoided this way. A third option is to use `context` with a colon to specify

a directory prefix where the *TEXMF* is located that the files should be loaded from, e. g. `context:~/context/tex/texmf-context`. This can be used when referencing another distribution like the Context minimals that is installed under a different path not indexed by kpathsea.

The value of `log-level` sets the default verbosity of messages printed by Luaotfload. Only messages defined with a verbosity of less than or equal to the supplied value will be output on the terminal. At a log level of five Luaotfload can be very noisy. Also, printing too many messages will slow down the interpreter due to line buffering being disabled (see **setbuf**(3)).

The `resolver` setting allows choosing the font name resolution function: With the default value `cached` Luaotfload saves the result of a successful font name request to a cache file to speed up subsequent lookups. The alternative, `normal` circumvents the cache and resolves every request individually. (Since to the restructuring of the font name index in Luaotfload 2.4 the performance difference between the cached and uncached lookups should be marginal.)

FILES

Luaotfload only processes the first configuration file it encounters at one of the search locations. The file name may be either `luaotfload.conf` or `luaotfloadrc`, except for the dotfile in the user's home directory which is expected at `~/.luaotfloadrc`.

Configuration files are located following a series of steps. The search terminates as soon as a suitable file is encountered. The sequence of locations that Luaotfload looks at is

i. The current working directory of the LuaTeX process.

ii. The subdirectory `luaotfload/` inside the XDG configuration tree, e. g. `/home/oenothea/config/luaotfload/`.

iii. The dotfile.

iv. The *TEXMF* (using kpathsea).

SEE ALSO

luaotfload-tool(1), **luatex**(1), **lua**(1)

- `texdoc luaotfload` to display the PDF manual for the *Luaotfload* package
- Luaotfload development https://github.com/latex3/luaotfload
- LuaLaTeX mailing list http://tug.org/pipermail/lualatex-dev/
- LuaTeX http://luatex.org/
- Luaotfload on CTAN http://ctan.org/pkg/luaotfload

REFERENCES

- The XDG base specification http://standards.freedesktop.org/basedir-spec/basedir-spec-latest.html.

AUTHORS

Luaotfload was developed by the LuaLaTeX dev team (https://github.com/lualatex/). It is currently maintained by the LaTeX Project Team at https://github.com/latex3/luaotfload

This manual page was written by Philipp Gesang <phg@phi-gamma.net>.