NAME

Font::TTF::Font – Memory representation of a font

SYNOPSIS

Here is the regression test (you provide your own font). Run it once and then again on the output of the first run. There should be no differences between the outputs of the two runs.

```
$f = Font::TTF::Font->open($ARGV[0]);

# force a read of all the tables
$f->tables_do(sub { $_[0]->read; });

# force read of all glyphs (use read_dat to use lots of memory!)
# $f->{'loca'}->glyphs_do(sub { $_[0]->read; });
$f->{'loca'}->glyphs_do(sub { $_[0]->read_dat; });
# NB. no need to $g->update since $f->{'glyf'}->out will do it for us

$f->out($ARGV[1]);
$f->release;            # clear up memory forcefully!
```

DESCRIPTION

A Truetype font consists of a header containing a directory of tables which constitute the rest of the file. This class holds that header and directory and also creates objects of the appropriate type for each table within the font. Note that it does not read each table into memory, but creates a short reference which can be read using the form:

```
$f->{$tablename}->read;
```

Classes are included that support many of the different TrueType tables. For those for which no special code exists, the table type `table` is used, which defaults to Font::TTF::Table. The current tables which are supported are:

```
table       Font::TTF::Table      - for unknown tables
EBDT        Font::TTF::EBDT
EBLC        Font::TTF::EBLC
Feat        Font::TTF::GrFeat
GDEF        Font::TTF::GDEF
GPOS        Font::TTF::GPOS
GSUB        Font::TTF::GSUB
Glat        Font::TTF::Glat
Gloc        Font::TTF::Gloc
LTSH        Font::TTF::LTSH
OS/2        Font::TTF::OS_2
PCLT        Font::TTF::PCLT
Sill        Font::TTF::Sill
Silf        Font::TTF::Silf
bsln        Font::TTF::Bsln
cmap        Font::TTF::Cmap       - see also Font::TTF::OldCmap
cvt         Font::TTF::Cvt_
fdsc        Font::TTF::Fdsc
feat        Font::TTF::Feat
fmtx        Font::TTF::Fmtx
fpgm        Font::TTF::Fpgm
glyf        Font::TTF::Glyf       - see also Font::TTF::Glyph
hdmx        Font::TTF::Hdmx
head        Font::TTF::Head
hhea        Font::TTF::Hhea
hmtx        Font::TTF::Hmtx
```

```
kern        Font::TTF::Kern       - see alternative Font::TTF::AATKern
loca        Font::TTF::Loca
maxp        Font::TTF::Maxp
mort        Font::TTF::Mort       - see also Font::TTF::OldMort
name        Font::TTF::Name
post        Font::TTF::Post
prep        Font::TTF::Prep
prop        Font::TTF::Prop
vhea        Font::TTF::Vhea
vmtx        Font::TTF::Vmtx
DSIG        FONT::TTF::DSIG
```

Links are:

Font::TTF::Table Font::TTF::EBDT Font::TTF::EBLC Font::TTF::GrFeat Font::TTF::GDEF Font::TTF::GPOS Font::TTF::GSUB Font::TTF::Glat Font::TTF::Gloc Font::TTF::LTSH Font::TTF::OS_2 Font::TTF::PCLT Font::TTF::Sill Font::TTF::Silf Font::TTF::Bsln Font::TTF::Cmap Font::TTF::Cvt_ Font::TTF::Fdsc Font::TTF::Feat Font::TTF::Fmtx Font::TTF::Fpgm Font::TTF::Glyf Font::TTF::Hdmx Font::TTF::Head Font::TTF::Hhea Font::TTF::Hmtx Font::TTF::Kern Font::TTF::Loca Font::TTF::Maxp Font::TTF::Mort Font::TTF::Name Font::TTF::Post Font::TTF::Prep Font::TTF::Prop Font::TTF::Vhea Font::TTF::Vmtx Font::TTF::OldCmap Font::TTF::Glyph Font::TTF::AATKern Font::TTF::OldMort Font::TTF::DSIG

INSTANCE VARIABLES

Instance variables begin with a space (and have lengths greater than the 4 characters which make up table names).

nocsum
: This is used during output to disable the creation of the file checksum in the head table. For example, during DSIG table creation, this flag will be set to ensure that the file checksum is left at zero.

noharmony
: If set, do not harmonize the script and lang trees of GPOS and GSUB tables. See Font::TTF::Ttopen for more info.

nocompress
: Is the default value controlling WOFF output table compression. If undef, all tables will be compressed if there is a size benefit in doing so. It may be set to an array of tagnames naming tables that should not be compressed, or to a scalar integer specifying a table size threshold below which tables will not be compressed. Note that individual Font::TTF::Table objects may override this default. See Font::TTF::Table for more info.

fname (R)
: Contains the filename of the font which this object was read from.

INFILE (P)
: The file handle which reflects the source file for this font.

OFFSET (P)
: Contains the offset from the beginning of the read file of this particular font directory, thus providing support for TrueType Collections.

WOFF
: Contains a reference to a `Font::TTF::Woff` object.

METHODS

Font::TTF::Font->AddTable($tablename, `$class`**)**
: Adds the given class to be used when representing the given table name. It also 'requires' the class for you.

Font::TTF::Font->Init
: For those people who like making fonts without reading them. This subroutine will require all the table code for the various table types for you. Not needed if using Font::TTF::Font::read before using a table.

Font::TTF::Font->new(%props)
Creates a new font object and initialises with the given properties. This is primarily for use when a TTF is embedded somewhere. Notice that the properties are automatically preceded by a space when inserted into the object. This is in order that fields do not clash with tables.

Font::TTF::Font->open($fname)
Reads the header and directory for the given font file and creates appropriate objects for each table in the font.

`$f`**->read**
Reads a Truetype font directory starting from location `$self->{'OFFSET'}` in the file. This has been separated from the `open` function to allow support for embedded TTFs for example in TTCs. Also reads the `head` and `maxp` tables immediately.

`$f`**->out($fname [,** `@tablelist`**])**
Writes a TTF file consisting of the tables in tablelist. The list is checked to ensure that only tables that exist are output. (This means that you cannot have non table information stored in the font object with key length of exactly 4)

In many cases the user simply wants to output all the tables in alphabetical order. This can be done by not including a `@tablelist`, in which case the subroutine will output all the defined tables in the font in alphabetical order.

Returns `$f` on success and undef on failure, including warnings.

All output files must include the `head` table.

`$f`**->out_xml($filename [,** `@tables`**])**
Outputs the font in XML format

`$f`**->XML_start($context,** `$tag`**,** `%attrs`**)**
Handles start messages from the XML parser. Of particular interest to us are <font> and <table>.

`$f`**->update**
Sends update to all the tables in the font and then resets all the isDirty flags on each table. The data structure in now consistent as a font (we hope).

`$f`**->dirty**
Dirties all the tables in the font

`$f`**->tables_do(&func [, tables])**
Calls &func for each table in the font. Calls the table in alphabetical sort order as per the order in the directory:

```
&func($table, $name);
```

May optionally take a list of table names in which case func is called for each of them in the given order.

`$f`**->release**
Releases ALL of the memory used by the TTF font and all of its component objects. After calling this method, do **NOT** expect to have anything left in the `Font::TTF::Font` object.

NOTE, that it is important that you call this method on any `Font::TTF::Font` object when you wish to destruct it and free up its memory. Internally, we track things in a structure that can result in circular references, and without calling '`release()`' these will not properly get cleaned up by Perl. Once you've called this method, though, don't expect to be able to do anything else with the `Font::TTF::Font` object; it'll have **no** internal state whatsoever.

Developer note: As part of the brute-force cleanup done here, this method will throw a warning message whenever unexpected key values are found within the `Font::TTF::Font` object. This is done to help ensure that any unexpected and unfreed values are brought to your attention so that you can bug us to keep the module updated properly; otherwise the potential for memory leaks due to dangling circular references will exist.

BUGS

Bugs abound aplenty I am sure. There is a lot of code here and plenty of scope. The parts of the code which haven't been implemented yet are:

Post
: Version 4 format types are not supported yet.

Cmap
: Format type 2 (MBCS) has not been implemented yet and therefore may cause somewhat spurious results for this table type.

Kern
: Only type 0 & type 2 tables are supported (type 1 & type 3 yet to come).

TTC and WOFF
: The current Font::TTF::Font::out method does not support the writing of TrueType Collections or WOFF files.

DSIG
: Haven't figured out how to correctly calculate and output digital signature (DSIG) table

In addition there are weaknesses or features of this module library

- There is very little (or no) error reporting. This means that if you have garbled data or garbled data structures, then you are liable to generate duff fonts.
- The exposing of the internal data structures everywhere means that doing radical re-structuring is almost impossible. But it stop the code from becoming ridiculously large.

Apart from these, I try to keep the code in a state of ''no known bugs'', which given the amount of testing this code has had, is not a guarantee of high quality, yet.

For more details see the appropriate class files.

AUTHOR

Martin Hosken <http://scripts.sil.org/FontUtils>.

LICENSING