NAME

Font::TTF::Silf – The main Graphite table

DESCRIPTION

The Silf table holds the core of the Graphite rules for a font. A Silf table has potentially multiple silf subtables, although there is usually only one. Within a silf subtable, there are a number of passes which contain the actual finite state machines to match rules and the constraint and action code to be executed when a rule matches.

INSTANCE VARIABLES

Version
: Silf table format version

Compiler
: Lowest compiler version necessary to fully support the semantics expressed in this Graphite description

SILF
: An array of Silf subtables

  maxGlyphID
  : The maximum glyph id referenced including pseudo and non glyphs

  Ascent
  : Extra ascent to be added to the font ascent.

  Descent
  : Extra descent to be added to the font descent. Both values are assumed to be positive for a descender below the base line.

  substPass
  : Pass index into PASS of the first substitution pass.

  posPass
  : Pass index into PASS of the first positioning pass.

  justPass
  : Pass index into PASS of the first justification pass.

  bidiPass
  : Pass index of the pass before which the bidirectional processing pass will be executed. 0xFF indicates that there is no bidi pass to be executed.

  Flags
  : A bitfield of flags:

```
0 - Indicates there are line end contextual rules in one of the passes
```

  maxPreContext
  : Maximum length of a context preceding a cross line boundary contextualisation.

  maxPostContext
  : Maximum length of a context following a cross line boundary contextualsation.

  attrPseudo
  : Glyph attribute for the actual glyph id associated with a pseudo glyph.

  attrBreakWeight
  : Glyph attribute number of the attribute holding the default breakweight associated with a glyph.

  attrDirectionality
  : Glyph attribute number of the attribute holding the default directionality value associated with a glyph.

JUST
: The may be a number of justification levels each with their own property values. This points to an array of hashes, one for each justification level.

  attrStretch
  : Glyph attribute number for the amount of stretch allowed before this glyph.

  attrShrink
  : Glyph attribute number for the amount of shrink allowed before this glyph.

  attrStep
  : Glyph attribute number specifying the minimum granularity of actual spacing associated with this glyph at this level.

  attrWeight
  : Glyph attribute number giving the weight associated with spreading space across a run of glyphs.

  runto
  : Which level starts the next stage.

numLigComp
: Number of initial glyph attributes that represent ligature components

numUserAttr
: Number of user defined slot attributes referenced. Tells the engine how much space to allocate to a slot for user attributes.

maxCompPerLig
: Maximum number of components per ligature.

direction
: Supported directions for this writing system

CRIT_FEATURE
: Array of critical features.

scripts
: Array of script tags that indicate which set of GDL rules to execute if there is more than one in a font.

lbGID
: Glyph ID of the linebreak pseudo glyph.

pseudos
: Hash of Unicode values to pseduo glyph ids.

classes
: This is an array of classes, each of which is an array of glyph ids in class order.

PASS
: The details of rules and actions are stored in passes. This value is an array of pass subobjects one for each pass.

  flags
  : This is a bitfield:

```
0 - If true, this pass makes no change to the slot stream considered a
    Only slot attributes are expected to change (for example during po
```

  maxRuleLoop
  : How many times the engine will allow rules to be tested and run without the engine advancing through the input slot stream.

maxRuleContext
: Number of slots of input needed to run this pass.

maxBackup
: Number of slots by which the following pass needs to trail this pass (i.e. the maximum this pass is allowed to back up).

numRules
: Number of action code blocks, and so uncompressed rules, in this pass.

numRows
: Number of rows in the finite state machine.

numTransitional
: Number of rows in the finite state machine that are not final states. This specifies the number of rows in the fsm element.

numSuccess
: Number of success states. A success state may also be a transitional state.

numColumns
: Number of columns in the finite state machine.

colmap
: A hash, indexed by glyphid, that gives the fsm column number associated with that glyphid. If not present, then the glyphid is not part of the fsm and will finish fsm processing if it occurs.

rulemap
: An array of arrays, one for each success state. Each array holds a list of rule numbers associated with that state.

minRulePreContext
: Minimum number of items in a rule's precontext.

maxRulePreContext
: The maximum number of items in any rule's precontext.

startStates
: Array of starting state numbers dependeing on the length of actual precontext. There are maxRulePreContext − minRulePreContext + 1 of these.

ruleSortKeys
: An array of sort keys one for each rule giving the length of the rule including its precontext.

rulePreContexts
: An array of precontext lengths for each rule.

fsm
: A two dimensional array such that `$p->{'fsm'}[$row][$col]` gives the row of the next node to try in the fsm.

passConstraintLen
: Length in bytes of the passConstraint code.

passConstraintCode
: A byte string holding the pass constraint code.

constraintCode
: An array of byte strings holding the constraint code for each rule.

actionCode
: An array of byte strings holding the action code for each rule.

`@opcodes`
: Each array holds the name of the opcode, the number of operand bytes and a string describing the operands. The characters in the string have the following meaning:

```
c - lsb of class id
C - msb of class id
f - feature index
g - lsb of glyph attribute id
G - msb of glyph attribute id
l - lsb of a 32-bit extension to a 16-bit number
L - msb of a 32-bit number
m - glyph metric id
n - lsb of a number
N - msb of a 16-bit number
o - offset (jump)
s - slot reference
S - slot attribute id
v - variable number of following arguments
```

read

Reads the Silf table into the internal data structure

out

Outputs a Silf data structure to a font file in binary format

`$t->`*minsize()*

Returns the minimum size this table can be. If it is smaller than this, then the table must be bad and should be deleted or whatever.

AUTHOR

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

LICENSING