Getting started

\texttt{lualatex} is a typesetter; \texttt{texlua} and \texttt{lualatex --luaonly} are lua interpreters. In lua interpreter mode, the lua tables \texttt{tex}, \texttt{token}, \texttt{node}, and \texttt{pdf} are unavailable.

\texttt{\directlua} expands immediately, \texttt{\latelua} is processed during \texttt{\shipout}.

\texttt{\luaexversion: 78}
\texttt{\luaexrevision: 3}
\texttt{\luaexdatenumber: 2014031011}

Attributes

There are 65536 attribute registers, each of which obeys grouping and can have any integer value other than the most negative number (which indicates the value is unset).

Catcode tables

There are 65536 category code tables, each of which can hold the full range of Unicode. Category table manipulation:

- \texttt{\initcatactable} \texttt{N} initialize an 'initex' table in \texttt{N}
- \texttt{\savcatactable} \texttt{N} save the current category codes to table \texttt{N} globally
- \texttt{\catactable} \texttt{N} switch to predefined table \texttt{N}

Filenames

\texttt{\input}, \texttt{\openin}, and \texttt{\font} accept braced file names to allow embedded spaces.

Images and forms

\texttt{\pdfximage} and \texttt{\pdfximage} accept optional dimension parameters in the same format as \texttt{\pdfximage}.

Preloaded lua modules

\texttt{slunicode} \texttt{http://luaforge.net/projects/sln}
\texttt{luazip} \texttt{http://www.keplerproject.org/luazip/}
\texttt{luafilesystem} \texttt{http://www.keplerproject.org/luafilesystem/}
\texttt{lpeg} \texttt{http://www.inf.puc-rio.br/~roberto/lpeg.html}
\texttt{lzlib} \texttt{http://luaforge.net/projects/lzlib/}
\texttt{md5} \texttt{http://www.inf.puc-rio.br/~roberto/md5/md5-5/md5.html}
\texttt{luazip} \texttt{http://www.keplerproject.org/luazip/}
\texttt{luasocket} \texttt{http://www.tecgraf.puc-rio.br/~diego/professional/luasocket/}

String extensions

\texttt{table = string.explode} \texttt{(string, [string])}
Break a string into pieces. The optional argument is a character possibly followed by a plus sign (default " +")
\texttt{number = string.utfvalues} \texttt{(string)}
Iterator that returns a value representing a single UTF-8 token.
\texttt{string = string.utfcharacters} \texttt{(string)}
Iterator that returns a string representing a single UTF-8 token.
\texttt{string = string.characters} \texttt{(string)}
Iterator that returns a string representing a single 8-byte token.
\texttt{string, string = string.characterpairs} \texttt{(string)}
Iterator that returns two strings representing two single UTF-8 tokens.
\texttt{number, number = string.bybytes} \texttt{(string)}
Iterator that returns a value representing a single 8-byte token.
\texttt{number, number = string.bybytes} \texttt{(string)}

Operating system extensions

\texttt{os.exec} \texttt{(table)}
Run an external command and exit. The table is an array of arguments, with an optional argp[0] in index 0.
\texttt{boolean = os.spawn} \texttt{(table)}
Run an external command and return its exit code. The table is an array of arguments, with an optional argp[0] in index 0.
\texttt{os.datenumber} \texttt{(string, string)}
Set an environment variable.
\texttt{number = os.getenv} \texttt{(string)}
Get the time as a floating point number (Unix only).
\texttt{table = os.times} \texttt{()}
Return process times.
\texttt{string = os.tmpdir} \texttt{()}
Create a temporary directory inside the current directory.
\texttt{table = os.uname} \texttt{()}
Return various information strings about the computer.
\texttt{string = os.selfdir} \texttt{()}
Return the directory path of argp[0].

Lua file system extensions

\texttt{boolean = os.isdir} \texttt{(string)}
Return true if the string is a directory.
\texttt{boolean = os.isfile} \texttt{(string)}
Return true if the string is a file.
\texttt{string = os.realpath} \texttt{(string)}
Return the FAT name of a file (Windows only).
\texttt{string = os.readlines} \texttt{(string)}
Return the contents of a symlink (Unix only).

Callback table

\texttt{number, [string] = callback.register} \texttt{(string, function)}
Register a callback. Passing nil removes an existing callback. Returns nil, error on failure.
\texttt{table = callback.list} \texttt{()}
Produce a list of all known callback names.
\texttt{function = callback.find} \texttt{(string)}
Returns the function currently associated with a callback, or nil

File discovery callbacks

\texttt{string = find_read_file} \texttt{(number, string)}
Find a file for \texttt{\input} (0) or \texttt{\openin} (higher integers).
\texttt{string = find_write_file} \texttt{(number, string)}
Find a file for writing to the log file (0) or with \texttt{\write} (higher integers).
\texttt{string = find_font_file} \texttt{(string)}
Find a font metrics file.
\texttt{string = find_output_file} \texttt{(string)}
Find the output (PDF or DVI) file.
\texttt{string = find_format_file} \texttt{(string)}
Find the format file.
\texttt{string = find_vf_file} \texttt{(string)}
Find a VF file.
\texttt{string = find_map_file} \texttt{(string)}
Find a font map file.
\texttt{string = find_enc_file} \texttt{(string)}
Find a font encoding file.
\texttt{string = find_subfont_file} \texttt{(string)}
Find a subfont definition file.
\texttt{string = find_pk_file} \texttt{(string)}
Find a PK font bitmap file.
\texttt{string = find_data_file} \texttt{(string)}
Find an input data file for PDF attachment.
\texttt{string = find_opentype_file} \texttt{(string)}
Find an OpenType font file.
\texttt{string = find_truetype_file} \texttt{(string)}
Find an TrueType font file.
\texttt{string = find_type1_file} \texttt{(string)}
Find an Type1 (PostScript) font file.
\texttt{string = find_image_file} \texttt{(string)}
Find an image file for inclusion.

File reading callbacks

\texttt{table = open_read_file} \texttt{(string)}
Open a file for reading. The returned table should define key functions for "reader" and "close".

string = reader(table)
Read a line from a file opened with the open_read_file callback. The argument is the return value from open_read_file

close(table)
Close a file opened with the open_read_file callback. The argument is the return value from the open_read_file

boolean, string, number = read_font_file(string)
Read a TFM metrics file. Return true, the data, and the data length for success, false otherwise

boolean, string, number = read_vf_file(string)
Read a VF metrics file.

boolean, string, number = read_map_file(string)
Read a font map file.

boolean, string, number = read_enc_file(string)
Read a font encoding file.

boolean, string, number = read_sfd_file(string)
Read a subfont definition file.

boolean, string, number = read_type1_file(string)
Read a Type1 font.

boolean, string, number = read_opentype_file(string)
Read an OpenType font.

Tokenisation changes callbacks

token = process_input_buffer(string)
Modify the encoding of the input buffer.

token = process_output_buffer(string)
Modify the encoding of the output buffer.

table = token_filter()
Override the tokenization process. Return value is a token or an array of tokens

Node list callbacks

buildpage_filter(string)
Process objects as they are added to the main vertical list. The string argument gives some context.

buildpage_filter context information:

value explanation
alignment a (partial) alignment is being added
after_output an output routine has just finished
box a typeset box is being added
new_graf the beginning of a new paragraph
vmode_par \par was found in vertical mode
hmode_par \par was found in horizontal mode
insert an insert is added
penalty a penalty (in vertical mode)
before_display immediately before a display starts
after_display a display is finished
end \LaTeX{} is terminating (it’s all over)

node = pre_linebreak_filter(node, string)
Alter a node list before linebreaking takes place. The string argument gives some context.

pre_linebreak_filter context information:

value explanation
<empty> main vertical list
hbox \hbox in horizontal mode
adjusted_hbox \hbox in vertical mode
vbox \vbox

vtop \vtop
align \halign or \align
disc discretionaries
insert packaging an insert
vcenter \vcenter
local_box \localleftbox or \localrightbox
split_off top of a \vsplit
split_keep remainder of a \vsplit
align_set alignment cell
fin_row alignment row

node = linebreak_filter(node, boolean)
Override the linebreaking algorithm. The boolean is true if this is a pre-display break.

node = post_linebreak_filter(node, string)
Alter a node list after linebreaking has taken place. The string argument gives some context.

node = hpack_filter(node, string, number, string, string, string)
Alter a node list before horizontal packing takes place. The first string gives some context, the number is the desired size, the second string is either "exact" or "additional" (modifies the first string), the third string is the desired direction.

node = vpack_filter(node, string, number, string, number, string)
Alter a node list before vertical packing takes place. The second number is the desired max depth. See hpack_filter for the arguments.

node = pre_output_filter(node, string, number, string, number, string)
Alter a node list before boxing to \outputbox takes place. See vpack_filter for the arguments.

hypenate(node, node) Apply hyphenation to a node list.

ligaturing(node, node) Apply ligatureing to a node list.

kerning(node, node) Apply kerning to a node list.

node = misto_to_hlist(node, string, boolean)
Convert a math node list into a horizontal node list.

Font definition callback

metrics = define_font(string, number)
Define a font from within lua code. The arguments are the user-supplied information, with negative numbers indicating scaled, positive numbers at

Event callbacks

pre_dump() Run actions just before format dumping takes place.
stop_run() Run actions just before the end of the typesetting run.
start_run() Run actions at the start of the typesetting run.
start_page_number() Run actions at the start of typeset page number message reporting.
stop_page_number() Run actions at the end of typeset page number message reporting.
show_error_hook() Run action at error reporting time.
finish_pdf_file() Run actions just before the PDF closing takes place.

Epdf table

--- All constructors:

PDFDoc = epdf.open(string)
Construct a PDFDoc object by opening a PDF document.

Annot = epdf.Annot(XRef, Dict, Catalog, Ref)
Construct an Annot object.

Annots = epdf.Annots(XRef, Catalog, Object)
Construct an Annots object.

Array = epdf.Array(XRef) Construct an Array object.

Dict = epdf.Dict(XRef) Construct a Dict object.
Object = epdf.Object()  # Construct an Object object.

PDFRectangle = epdf.PDFRectangle()  # Construct a PDFRectangle object.

--- Annots methods:

    boolean = Annot.isOK()  # Check if Annot object is ok.
    Object = Annot.getAppearance()  # Get Appearance object.
    AnnotBorder = Annot.getBorder()  # Get AnnotBorder object.
    boolean = AnnotBorder.match(Ref)  # Check if object number and generation matches Ref.

--- AnnosBorderStyle methods:

    number = AnnotBorderStyle.getWidth()  # Get border width.

--- Annos methods:

    integer = Annots.getNumAnnots()  # Number of Annots objects.
    Annot = Annots.getAnnot(integer)  # Get Annot object.

--- Array methods:

    Array:incRef(integer)  # Increment reference count to Array.
    Array:decRef()  # Decrement reference count to Array.

--- Catalog methods:

    integer = Catalog.initDict(integer)  # Initialize a Dict-type object with an empty dictionary.
    Catalog = Catalog.initCmd()  # Initialize a Command object.
    String = Catalog.initString(string)  # Initialize a String-type object.
    Name = Catalog.initName(string)  # Initialize a Name-type object.
    Null = Catalog.initNull()  # Initialize an empty object.
    String = Catalog.initReal(integer)  # Initialize a Real-type object.

--- LinkDest methods:

    LinkDest.getKind()  # Get the kind of LinkDest object.
    LinkDest.getKindName()  # Get the name of LinkDest kind.
    LinkDest = LinkDest.getPageRef(integer)  # Get the PageRef object.

--- Link methods:

    boolean = Link.isOK()  # Check if Link object is ok.
    Link = Link.inRect(number, number)  # Check if point is inside the link rectangle.

--- Links methods:

    integer = Links.getNumLinks()  # Number of links.
    Link = Links.getLink(integer)  # Get the link by number.

--- Object methods:

    Object.initDict()  # Initialize a Dict-type object.
    Object.initStream()  # Initialize a Stream-type object.
    Object.initRef()  # Initialize a Ref-type object.
    Object.initCmd()  # Initialize a Cmd-type object.
    Object.initError()  # Initialize an Error-type object.

--- LinkDest methods:

    LinkDest.isOK()  # Check if LinkDest object is ok.
    LinkDest.getKind()  # Get the kind of LinkDest object.
    LinkDest = LinkDest.getPageRef()  # Get the PageRef object.

--- Link methods:

    boolean = Link.isOK()  # Check if Link object is ok.
    Link = Link.inRect(number, number)  # Check if point is inside the link rectangle.

--- Links methods:

    integer = Links.getNumLinks()  # Number of links.
    Link = Links.getLink(integer)  # Get the link by number.

--- Object methods:

    Object.initDict()  # Initialize a Dict-type object.
    Object.initStream()  # Initialize a Stream-type object.
    Object.initRef()  # Initialize a Ref-type object.
    Object.initCmd()  # Initialize a Cmd-type object.
    Object.initError()  # Initialize an Error-type object.
    Object.initEOF()  # Initialize an EOF-type object.

--- LinkDest methods:

    LinkDest.isOK()  # Check if LinkDest object is ok.
    LinkDest.getKind()  # Get the kind of LinkDest object.
    LinkDest = LinkDest.getPageRef()  # Get the PageRef object.

--- Link methods:

    boolean = Link.isOK()  # Check if Link object is ok.
    Link = Link.inRect(number, number)  # Check if point is inside the link rectangle.

--- Links methods:

    integer = Links.getNumLinks()  # Number of links.
    Link = Links.getLink(integer)  # Get the link by number.

--- Object methods:

    Object.initDict()  # Initialize a Dict-type object.
    Object.initStream()  # Initialize a Stream-type object.
    Object.initRef()  # Initialize a Ref-type object.
    Object.initCmd()  # Initialize a Cmd-type object.
    Object.initError()  # Initialize an Error-type object.
    Object.initEOF()  # Initialize an EOF-type object.

--- LinkDest methods:

    LinkDest.isOK()  # Check if LinkDest object is ok.
    LinkDest.getKind()  # Get the kind of LinkDest object.
    LinkDest = LinkDest.getPageRef()  # Get the PageRef object.

--- Link methods:

    boolean = Link.isOK()  # Check if Link object is ok.
    Link = Link.inRect(number, number)  # Check if point is inside the link rectangle.

--- Links methods:

    integer = Links.getNumLinks()  # Number of links.
    Link = Links.getLink(integer)  # Get the link by number.

--- Object methods:

    Object.initDict()  # Initialize a Dict-type object.
    Object.initStream()  # Initialize a Stream-type object.
    Object.initRef()  # Initialize a Ref-type object.
    Object.initCmd()  # Initialize a Cmd-type object.
    Object.initError()  # Initialize an Error-type object.
    Object.initEOF()  # Initialize an EOF-type object.

--- LinkDest methods:

    LinkDest.isOK()  # Check if LinkDest object is ok.
    LinkDest.getKind()  # Get the kind of LinkDest object.
    LinkDest = LinkDest.getPageRef()  # Get the PageRef object.

--- Link methods:

    boolean = Link.isOK()  # Check if Link object is ok.
    Link = Link.inRect(number, number)  # Check if point is inside the link rectangle.

--- Links methods:

    integer = Links.getNumLinks()  # Number of links.
    Link = Links.getLink(integer)  # Get the link by number.
Get name from Name-type object as a string.

Object:arrayGetLength()  Get array length from Array-type object.

Object:arrayAdd(Object)  Add Object to Array-type object.

Object:arrayGet(integer)  Get Object from Array-type object.

Object:arrayGetNF(integer)  Get Object from Array-type object, not resolving indirection.

Object:dictGetLength()  Get dictionary length from Dict-type object.

Object:dictAdd(string, Object)  Add Object to Dict-type object.

Object:dictGet(string, Object)  Get Object from Dict-type object.

Object:dictGetKey(integer)  Get Dict key of Dict-type object by number.

Object:dictGetVal(integer)  Get Dict value of Dict-type object by number.

Object:dictGetValNF(integer)  Get Dict value of Dict-type object by number, not resolving indirection.

PDFDoc:isOK()  Get if PDFDoc object is ok.

PDFDoc:getInt()  Get integer from Int-type object.

PDFDoc:getReal()  Get number from Real-type object.

PDFDoc:getNum()  Get number from Num-type object.

PDFDoc:getString()  Get string from String-type object.

PDFDoc:getName()  Get object name from Ref-type object.

Object:getCmd()  Get cmd from Object-type object.

Object:getRefGen()  Get ref gen from Object-type object.

Object:getStream()  Get stream from Object-type object.

Object:getDict()  Get dict from Object-type object.

Object:getArray()  Get array from Object-type object.

Object:getReal()  Get real from Object-type object.

Object:getInt()  Get int from Object-type object.

Object:getString()  Get string from Object-type object.

Object:getNum()  Get num from Object-type object.

Object:getDict()  Get dict from Object-type object.

Object:getArray()  Get array from Object-type object.

Object:getString()  Get string from Object-type object.
Luatex 0.66 short reference

Font table

metrics = font.read_tfm(string, number)
Parse a font metrics file, at the size indicated by the number.

metrics = font.read_vf(string, number)
Parse a virtual font metrics file, at the size indicated by the number.

metrics = font.getfont(number)
Fetch an internal font id as a lua table.

font.setfont(number, metrics)
Set an internal font id from a lua table.

boolean = font.frozen(number)
True if the font is frozen and can no longer be altered.

number = font.define(metrics)
Process a font metrics table and stores it in the internal font table, returning its internal id.

number = font.nextid() Return the next free font id number.

number = font.id(string)
Return the font id of the font accessed by the classname given.

[number] = font.current([number]) Get or set the currently active font number = font.max() Return the highest used font id at this moment.

number, metrics = font.each()
Iterate over all the defined fonts.

Font loader table

table = fontloader.info(string)
Get various information fields from an font file.

fontloader.info returned information:

key type explanation

fontname string the POSTSCRIPT name of the font
furname string the formal name of the font
familyname string the family name this font belongs to
weight string a string indicating the color value of the font
version string the internal font version
italicangle float the slant angle

luafont, table = fontloader.open(string, [string])
Parse a font file and return a table representing its contents. The optional argument is the name of the desired font in case of font collection files.

The optional return value contains any parser error strings.

Listing all of the substructure returned from fontloader.open would take too much room, see the big reference manual.

fontloader.apply_featurefile(luafont, string)
Apply a feature file to a fontloader table.

fontloader.apply_afmfile(luafont, string)
Apply an AFM file to a fontloader table.

Image table

Full list of <image> object fields:

field name type description

depth number the image depth for \LUA\TeX{} (in scaled points)
height number the image height for \LUA\TeX{} (in scaled points)
width number the image width for \LUA\TeX{} (in scaled points)
transform number the image transform, integer number 0..7
attr string the image attributes for \LUA\TeX{}
filename string the image file name
stream string the raw stream data for an /XObject /Form object
page ?? the identifier for the requested image page (type is number or string, default is the number 1)
pagebox string the requested bounding box, one of none, media, crop, bleed, trim, art
bbox table table with 4 bounding box dimensions llx, lly, urx, andurry overruling the pagebox entry
filepath string the full (expanded) file name of the image
colordepth number the number of bits used by the color space
colorspace number the color space object number
imagetype string one of pdf, png, jpg, jbig2, or nil
objnum number the PDF image object number
index number the PDF image name suffix
pages number the total number of available pages
xsize number the natural image width
ysize number the natural image height
xres number the horizontal natural image resolution (in DPI)
yres number the vertical natural image resolution (in DPI)

image = img.new([table])

table = img.keys()
Returns a table with possible image table keys, including retrieved information.

image = img.scan(image)
Processes an image file and stores the retrieved information in the image object.

image = img.copy(image) Copy an image.

image = img.write(image) Write the image to the PDF file.

image = img.immediatewrite(image)
Write the image to the PDF file immediately.

date = img.date(image) Returns the node associated with an image.


Kpathsea table

kpse.set_program_name(string, [string])
Initialize the kpathsea library by setting the program name. The optional string allows explicit programme setting.

kpathsea = kpse.new(string, [string])
Create a new kpathsea library instance. The optional string allows explicit programme setting.

string = kpse.find_file(string, [string], [boolean], [number])
Find a file. The optional string is the file type as supported by the standalone kpse which program (default is "test", no autodiscovery takes place).

The optional boolean indicates whether the file must exist. The optional number is the dpi value for PK files.

string = kpse.lookup(string, table)
Find a file (extended interface). The kpse.lookup options match commandline arguments from kpse which:

key type description

diag number set debugging flags for this lookup
format string use specific file type (see list above)
dpi number use this resolution for this lookup; default 600
path string search in the given path
all boolean output all matches, not just the first
must-exist boolean search the disk as well as ls-R if necessary
mktxpk boolean disable/enable mktxpk generation for this lookup
mktxps boolean disable/enable mktxps generation for this lookup
mktxpm boolean disable/enable mktxpm generation for this lookup
mktxpm boolean disable/enable mktxpm generation for this lookup
subdir string or table only output matches whose directory part ends with the given string(s)

kpse.init_prog(string, number, string, [string])
Initialize a PK generation program. The optional string is the metafont mode fallback name.

string = kpse.readable_file(string)
Returns true if a file exists and is readable.

string = kpse.expand_path(string)  Expand a path.
string = kpse.expand_var(string)   Expand a variable.
string = kpse.expand_braces(string) Expand the braces in a variable.
string = kpse.show_path(string)   List the search path for a specific file type.
string = kpse.var_value(string)   Return the value of a variable.
string = kpse.version()     Return the kpathsea version.

### Language table

**language = lang.new[number]**

Create a new language object, with an optional fixed id number.
**number = lang.id[language]**

Returns the current internal language id number.
**[string] = lang.hyphenation[language, [string]]**

Get or set hyphenation exceptions.
**lang.clear_hyphenation[language]**

Clear the set of hyphenation exceptions.
**string = lang.clean[string]**

Creates a hyphenation key from the supplied hyphenation exception.
**[string] = langpatterns[language, [string]]**

Get or set hyphenation patterns.
**lang.clear_patterns[language]**

Clear the set of hyphenation patterns.
**[number] = lang.prehyphenchar[language, [number]]**

Set the pre-hyphenchar for implicit hyphenation.
**[number] = lang.posthyphenchar[language, [number]]**

Set the post-hyphenchar for implicit hyphenation.
**[number] = lang.preexhyphenchar[language, [number]]**

Set the pre-ex-phynchar for explicit hyphenation.
**[number] = lang.postexhyphenchar[language, [number]]**

Set the post-ex-phynchar for explicit hyphenation.
**boolean = lang.hyphenate[node, [node]]**

Hyphenate a node list.

### Lua table

There are 65536 bytecode registers, that are saved in the format file. Assignments are always global.
**function = lua.getbytecode[number]**

Return a previously stored function from a bytecode register.
**lua.setbytecode[number, function]**

Save a function in a bytecode register.
They also be accessed via the virtual array [lua.bytecode]. The virtual array lua.name[1] can be used to give names to lua chunks. To use lua.name[1], set lua.name[1] = 'testname' and 'directlua{rubbish}.'

### Metapost table

**string = mplib.version()**

Returns the mplib version.
**mpinstance = mplib.new[number]**

Create a new metapost instance.
**mpdata = mp.execute(string)**

Execute metapost code in the instance.
**mpdata = mp.finish()**

Finish a metapost instance.
The return value of mp.execute and mp.finish is a table with a few possible keys (only status is always guaranteed to be present).
**log string output to the ‘log’ stream**

**term string output to the ‘term’ stream**

**error string output to the ‘error’ stream (only used for ‘out of memory’)**

**status number the return value: 0=good, 1=warning, 2=errors, 3=fatal error**

**fig table an array of generated figures (if any)**

Handling of fig objects would take too much room here, please see the big reference manual.

```plaintext
table = mp:statistics()  Returns some statistics for this metapost instance.
number = mp:char_width[string, number]  Report a character’s width.
number = mp:char_height[string, number]  Report a character’s height.
number = mp:char_depth[string, number]  Report a character’s depth.
```

### Node table

**table = node.types()**

Return the list of node types.
**table = node.whatsits()**

Return the list of whatsit types.
**boolean = node.is_node[any]**

Return true if the object is a <node>.
**number = node.id[string]**

Convert a node type string into a node id number.
**number = node.subtype[string]**

Convert a whatsit type string into a node subtype number.
**string = node.type[number]**

Convert a node id number into a node type string.
**table = node.fields[number, [number]]**

Report the fields a node type understands. The optional argument is needed for whatsits.
**boolean = node.has_field[node, string]**

Return true if the node understands the named field.
**node = node.new[number, [number]]**

Create a new node with id and (optional) subtype.
**node.free[node]**

Release a node.
**node.flush_list[node]**

Release a list of nodes.
**node = node.copy[node]**

Copy a node.
**node = node.copy_list[node, [node]]**

Copy a list node.
**number, number = node.hpack[node, [number], [string], [string]]**

Pack a node list into a horizontal list. The number is the desired size, the first string is either “exact” or “additional” (modifies the first string), the second string is the desired direction.
**node, number = node.vpack[node, [number], [string], [string]]**

Pack a node list into a vertical list. Arguments as for node.hpack
**number, number, number = node.dimensions[number, [number], [number], [number], [node]]**

Return the natural dimensions of a (horizontal) node list. The 3 optional numbers represent glue_set, glue_sign, and glue_order. The calculation stops just before the optional node (default end of list)
**node = node.mlist_to_hlist[node, string, boolean]**

Recursively convert a math list into a horizontal list. The string differentiates display and inline, the boolean whether penalties are inserted
**node = node.slide[node]**

Move to the last node of a list while fixing next and prev pointers.
**node = node.tail[node]**

Return the last node in a list.
**number = node.length[node, [node]]**

Return the length of a node list. Processing stops just before the optional node.
**number = node.count[number, [node]]**

Return the count of nodes with a specific id in a node list. Processing stops just before the optional node.
**node = node.traverse[node]**

Iterate over a node list.
**node = node.traverse_id[number, node]**

Iterate over nodes with id matching the number in a node list.
**node, node = node.remove[node, node]**

Extract and remove a second node from the list that starts in the first node.
**node, node = node.insert_before[node, node, node]**

Insert the third node just before the second node in the list that starts at the first node.
**node, node = node.insert_after[node, node, node]**

Insert the third node just after the second node in the list that starts at the first node.
**node = node.first_glyph[node, [node]]**

Return the first character node in a list. Processing stops just before the optional node.
node, node, boolean = node.ligaturing(node, [node])
Apply the internal ligaturing routine to a node list. Processing stops just before the optional node.
node, node, boolean = node.kerning(node, [node])
Apply the internal kerning routine to a node list. Processing stops just before the optional node.

node.unprotect_glyphs(node)
Mark all characters in a node list as being processed glyphs.

node.protect_glyphs(node)
Mark all processed glyphs in a node list as being characters.

node = node.last_node()
Pops and returns the last node on the current output list.

node.write(node)
Appends a node to the current output list.

boolean = node.protrusion_skippable(node)
Return true if the node could be skipped for protrusion purposes.
node = node.next(node)
Returns the next node.
node = node.prev(node)
Returns the previous node.

number = node.has_attribute(node, number, [number])
Return an attribute value for a node, if it has one. The optional number tests for a specific value.

node.set_attribute(node, number, value)
Set an attribute value for a node. The optional number tests for a specific value.

node.unset_attribute(node, number, [number])
Unset an attribute value for a node. The optional number tests for a specific value.

Pdf table

number = pdf.immediateobj([number], [string], string, [string])
Write an object to the PDF file immediately. The optional number is an object id, the first optional string is "file", "stream", or "filestream", the second optional string contains stream attributes for the latter two cases.

pdf.mapfile(string) Register a font map file.

pdf.mapline(string) Register a font map line.

number = pdf.object([number], [string], string, [string])
Write an object to the PDF file. See "pdf.immediateobj" for arguments.

pdf.reftobj(number) Reference an object, so that it will be written out.

number = pdf.pageref(number) Return the page reference object number.

pdf.print([string], string)
Write directly to the PDF file (use in \latex{}). The optional string is one of "direct" or "page".

number = pdf.reserveobj()
Reserve an object number in the PDF backend.

pdf.registerannot(number)
Register an annotation in the PDF backend.

Status table

table = status.list[]
Returns a table with various status items.

key |
---|---
pdf.gone | written PDF bytes
pdf.ptr | not yet written PDF bytes
dvi.gone | written DVI bytes
dvi.ptr | not yet written DVI bytes
total_pages | number of written pages
output_file_name | name of the PDF or DVI file
log_name | name of the log file
var_used | variable (one|-|word) memory in use
dyn_used | token (multi|-|word) memory in use
str.ptr | number of strings
init_str.ptr | number of \texttt{INTeX} strings
max_strings | maximum allowed strings
pool.ptr | string pool index

Typesetting table

tex.set(string, string, value)
Set a named internal register. Also accepts a predefined csname string.

value = tex.get(string)
Get a named internal register. Also accepts a predefined csname string.
Many of \texttt{LUATEX}'s internal parameters can be queried and set this way, but not nearly all. The big reference manual has an extensive list.

tex.setattribute(string, number, value)
Set an attribute register. Also accepts a predefined csname string.

number = tex.getattribute(number)
Get an attribute register. Also accepts a predefined csname string.

tex.setbbox(string, number, node)
Set a box register. Also accepts a predefined csname string.

node = tex.getbbox(number)
Get a box register. Also accepts a predefined csname string.

tex.setcount(string, number, value)
Set a counter register. Also accepts a predefined csname string.

number = tex.getcount(number)
Get a counter register. Also accepts a predefined csname string.


```
text.setdimen([string], number, number)
  Set a dimen register. Also accepts a predefined csname string.
  number = tex.getdimen(number)
  Get a dimen register. Also accepts a predefined csname string.

text.setskiptext(string, number, node)
  Set a skip register. Also accepts a predefined csname string.
  node = tex.getskiptext(number)
  Get a skip register. Also accepts a predefined csname string.

text.settoks([string], string, string, number)
  Set atoks register. Also accepts a predefined csname string.
  number = tex.gettoks([string], number, number)
  Get atoks register. Also accepts a predefined csname string.

text.setucode([string], number, number)
  Set an uppercasecode.

text.setsfcode([string], number, number)
  Set a lowercasecode.

text.setlccode([string], number, number)
  Get a lowercasecode.

text.getucode([string], number, number)
  Get an uppercasecode.

text.getsfcode([string], number, number)
  Get a lowercasecode.

text.getlccode([string], number, number)
  Get a lowercasecode.

text.getcatcode([string], number, number)
  Get a categorycode.

```

The itemscanalsobeaccesseddirectlyviavirtualarrays:
- `texЧgetdelcode` returns the tabledata:
- `texЧsetdelcode` sets the table.
- `texЧsetmathcode` sets the mathematical code.
- `texЧgetmathcode` gets the mathematical code.
- `texЧsetsfcode` sets the lowercase code.
- `texЧgetsfcode` gets the lowercase code.
- `texЧsetlccode` sets the lowercase code.
- `texЧgetlccode` gets the lowercase code.
- `texЧsetucode` sets the uppercase code.
- `texЧgetucode` gets the uppercase code.
- `texЧgetcatcode` gets the category code.

```
number = texЧextraprimatives([string], [string])
  Return all primitives in a (set of) extension identifiers. Valid identifiers
  are: "tex", "core", "etex", "pdf2tex", "omega", "aleph", and "luatex".
  table = texЧprimitives()
  Returns a table of all currently active primitives, with their meaning.
  number = texЧbadness(number, number)
  Compute a badness value.

```
entered in, negative inside the output routine.

head node all the head of the current list
tail node all the tail of the current list
prevgraf number vmode number of lines in the previous paragraph
prevdepth number vmode depth of the previous paragraph
dirs node hmode internal use only
noad node mmode internal use only
delimptr node mmode internal use only
mathdir boolean mmode true when during math processing the \textdir is not the same as the surrounding \textdir
mathstyle number mmode the current \mathstyle

Texconfig table

This is a table that is created empty. A startup LUA script could fill this table with a number of settings that are read out by the executable after loading and executing the startup file.

<table>
<thead>
<tr>
<th>key</th>
<th>type</th>
<th>default</th>
<th>explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>kpse_init</td>
<td>boolean</td>
<td>true</td>
<td>false totally disables KPATHSEA initialisation</td>
</tr>
<tr>
<td>shell_escape</td>
<td>string</td>
<td>cf.web2c docs</td>
<td></td>
</tr>
<tr>
<td>shell_escape_commands</td>
<td>string</td>
<td>cf.web2c docs</td>
<td></td>
</tr>
<tr>
<td>string_vacancies</td>
<td>number</td>
<td>75000</td>
<td>cf.web2c docs</td>
</tr>
<tr>
<td>pool_free</td>
<td>number</td>
<td>5000</td>
<td>cf.web2c docs</td>
</tr>
<tr>
<td>max_strings</td>
<td>number</td>
<td>15000</td>
<td>cf.web2c docs</td>
</tr>
<tr>
<td>strings_free</td>
<td>number</td>
<td>100</td>
<td>cf.web2c docs</td>
</tr>
<tr>
<td>nest_size</td>
<td>number</td>
<td>50</td>
<td>cf.web2c docs</td>
</tr>
<tr>
<td>max_in_open</td>
<td>number</td>
<td>15</td>
<td>cf.web2c docs</td>
</tr>
<tr>
<td>param_size</td>
<td>number</td>
<td>60</td>
<td>cf.web2c docs</td>
</tr>
<tr>
<td>save_size</td>
<td>number</td>
<td>4000</td>
<td>cf.web2c docs</td>
</tr>
<tr>
<td>stack_size</td>
<td>number</td>
<td>300</td>
<td>cf.web2c docs</td>
</tr>
<tr>
<td>dvi_buf_size</td>
<td>number</td>
<td>16384</td>
<td>cf.web2c docs</td>
</tr>
<tr>
<td>error_line</td>
<td>number</td>
<td>79</td>
<td>cf.web2c docs</td>
</tr>
<tr>
<td>half_error_line</td>
<td>number</td>
<td>50</td>
<td>cf.web2c docs</td>
</tr>
<tr>
<td>max_print_line</td>
<td>number</td>
<td>79</td>
<td>cf.web2c docs</td>
</tr>
<tr>
<td>hash_extra</td>
<td>number</td>
<td>0</td>
<td>cf.web2c docs</td>
</tr>
<tr>
<td>pk_dpi</td>
<td>number</td>
<td>72</td>
<td>cf.web2c docs</td>
</tr>
<tr>
<td>trace_file_names</td>
<td>boolean</td>
<td>true</td>
<td>false disables \TeX’s normal file feedback</td>
</tr>
<tr>
<td>file_line_error</td>
<td>boolean</td>
<td>false</td>
<td>file:line style error messages</td>
</tr>
<tr>
<td>halt_on_error</td>
<td>boolean</td>
<td>false</td>
<td>abort run on the first encountered error</td>
</tr>
<tr>
<td>formatname</td>
<td>string</td>
<td></td>
<td>if no format name was given on the commandline, this will be used</td>
</tr>
<tr>
<td>jobname</td>
<td>string</td>
<td></td>
<td>as formatname.</td>
</tr>
</tbody>
</table>

IO table

texio.write([string], string)
Write a string to the log and/or terminal. The optional argument is "term", "term and log", or "log".
texio.write_nl([string], string)
Write a string to the log and/or terminal, starting on a new line. The optional argument is "term", "term and log", or "log".

Token table

A token is represented in LUA as a small table. For the moment, this table consists of three numeric entries:

<table>
<thead>
<tr>
<th>index</th>
<th>meaning</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>command code</td>
<td>this is a value between 0 and 130</td>
</tr>
<tr>
<td>2</td>
<td>command modifier</td>
<td>this is a value between 0 and 2^{21}</td>
</tr>
<tr>
<td>3</td>
<td>control sequence id</td>
<td>for commands that are not the result of control sequences, like letters and characters, it is zero, otherwise, it is a number pointing into the 'equivalence table'</td>
</tr>
</tbody>
</table>