

# The luamplib package

Hans Hagen, Taco Hoekwater, Elie Roux, Philipp Gesang and Kim Dohyun  
Maintainer: LuaLaTeX Maintainers — Support: <lualatex-dev@tug.org>

2021/03/11 v2.20.7

## Abstract

Package to have metapost code typeset directly in a document with LuaTeX.

## 1 Documentation

This package aims at providing a simple way to typeset directly metapost code in a document with LuaTeX. LuaTeX is built with the lua mplib library, that runs metapost code. This package is basically a wrapper (in Lua) for the Lua mplib functions and some TeX functions to have the output of the mplib functions in the pdf.

In the past, the package required PDF mode in order to output something. Starting with version 2.7 it works in DVI mode as well, though DVIPDFMx is the only DVI tool currently supported.

The metapost figures are put in a TeX hbox with dimensions adjusted to the metapost code.

Using this package is easy: in Plain, type your metapost code between the macros \mplibcode and \endmplibcode, and in L<sup>A</sup>T<sub>E</sub>X in the mplibcode environment.

The code is from the luatex-mplib.lua and luatex-mplib.tex files from ConTeXt, they have been adapted to L<sup>A</sup>T<sub>E</sub>X and Plain by Elie Roux and Philipp Gesang, new functionalities have been added by Kim Dohyun. The changes are:

- a L<sup>A</sup>T<sub>E</sub>X environment
- all TeX macros start by mplib
- use of luatexbase for errors, warnings and declaration
- possibility to use btex ... etex to typeset TeX code. texttext() is a more versatile macro equivalent to TEX() from TEX.mp. TEX() is also allowed and is a synonym of texttext().

N.B. Since v2.5, btex ... etex input from external mp files will also be processed by luamplib.

N.B. Since v2.20, verbatimtex ... etex from external mp files will be also processed by luamplib. Warning: This is a change from previous version.

Some more changes and cautions are:

**\mplibforcehmode** When this macro is declared, every mplibcode figure box will be typeset in horizontal mode, so \centering, \raggedleft etc will have effects. \mplibnoforcehmode, being default, reverts this setting. (Actually these commands redefine \prependtomplibbox. You can define this command with anything suitable before a box.)

**\mpliblegacybehavior{enable}** By default, \mpliblegacybehavior{enable} is already declared, in which case a verbatimex ... etex that comes just before beginfig() is not ignored, but the T<sub>E</sub>X code will be inserted before the following mplib hbox. Using this command, each mplib box can be freely moved horizontally and/or vertically. Also, a box number might be assigned to mplib box, allowing it to be reused later (see test files).

```
\mplibcode
verbatimex \moveright 3cm etex; beginfig(0); ... endfig;
verbatimex \leavevmode etex; beginfig(1); ... endfig;
verbatimex \leavevmode\lower 1ex etex; beginfig(2); ... endfig;
verbatimex \endgraf\moveright 1cm etex; beginfig(3); ... endfig;
\endmplibcode
```

N.B. \endgraf should be used instead of \par inside verbatimex ... etex.

By contrast, T<sub>E</sub>X code in VerbatimTeX(...) or verbatimex ... etex between beginfig() and endfig will be inserted after flushing out the mplib figure.

```
\mplibcode
D := sqrt(2)**7;
beginfig(0);
draw fullcircle scaled D;
VerbatimTeX("\gdef\Dia{" & decimal D & "}");
endfig;
\endmplibcode
diameter: \Dia bp.
```

**\mpliblegacybehavior{disable}** If \mpliblegacybehavior{disabled} is declared by user, any verbatimex ... etex will be executed, along with btex ... etex, sequentially one by one. So, some T<sub>E</sub>X code in verbatimex ... etex will have effects on btex ... etex codes that follows.

```
\begin{mplibcode}
beginfig(0);
draw btex ABC etex;
verbatimex \bfseries etex;
draw btex DEF etex shifted (1cm,0); % bold face
draw btex GHI etex shifted (2cm,0); % bold face
endfig;
\end{mplibcode}
```

**About figure box metrics** Notice that, after each figure is processed, macro `\MPwidth` stores the width value of latest figure; `\MPheight`, the height value. Incidentally, also note that `\MPllx`, `\MPlly`, `\MPurx`, and `\MPury` store the bounding box information of latest figure without the unit bp.

**`\everymplib`, `\everyendmplib`** Since v2.3, new macros `\everymplib` and `\everyendmplib` re-define token lists `\everymplibtoks` and `\everyendmplibtoks` respectively, which will be automatically inserted at the beginning and ending of each `mplib` code.

```
\everymplib{ beginfig(0); }
\everyendmplib{ endfig; }
\mplibcode % beginfig/endfig not needed
  draw fullcircle scaled 1cm;
\endmplibcode
```

**`\mpdim`** Since v2.3, `\mpdim` and other raw  $\TeX$  commands are allowed inside `mplib` code. This feature is inspired by `gmp.sty` authored by Enrico Gregorio. Please refer the manual of `gmp` package for details.

```
\begin{mplibcode}
  draw origin--(\mpdim{\linewidth},0) withpen pencircle scaled 4
  dashed evenly scaled 4 withcolor \mpcolor{orange};
\end{mplibcode}
```

N.B. Users should not use the protected variant of `btex ... etex` as provided by `gmp` package. As `luamplib` automatically protects  $\TeX$  code inbetween, `btex` is not supported here.

**`\mpcolor`** With `\mpcolor` command, color names or expressions of `color`/`xcolor` packages can be used inside `mplibcode` environment (after `withcolor` operator), though `luamplib` does not automatically load these packages. See the example code above. For spot colors, `(x)spotcolor` (in PDF mode) and `xespotcolor` (in DVI mode) packages are supported as well.

**`\mplibnumbersystem`** Users can choose `numbersystem` option since v2.4. The default value scaled can be changed to double or decimal by declaring `\mplibnumbersystem{double}` or `\mplibnumbersystem{decimal}`. For details see <http://github.com/lualatex/luamplib/issues/21>.

**Settings regarding cache files** To support `btex ... etex` in external `.mp` files, `luamplib` inspects the content of each and every `.mp` input files and makes caches if necessary, before returning their paths to Lua $\TeX$ 's `mplib` library. This would make the compilation time longer wastefully, as most `.mp` files do not contain `btex ... etex` command. So `luamplib` provides macros as follows, so that users can give instruction about files that do not require this functionality.

- `\mplibmakenocache{<filename>[,<filename>,...]}`

- `\mplibcancelnocache{<filename>[,<filename>,...]}`

where `<filename>` is a file name excluding `.mp` extension. Note that `.mp` files under `$TEXMFMAIN/metapost/base` and `$TEXMFMAIN/metapost/context/base` are already registered by default.

By default, cache files will be stored in `$TEXMFVAR/luamplib_cache` or, if it's not available, in the same directory as where pdf/dvi output file is saved. This however can be changed by the command `\mplibcachedir{<directory path>}`, where tilde (`~`) is interpreted as the user's home directory (on a windows machine as well). As backslashes (`\`) should be escaped by users, it would be easier to use slashes (`/`) instead.

**\mplibtexttextlabel** Starting with v2.6, `\mplibtexttextlabel{enable}` enables string labels typeset via `texttext()` instead of `infont` operator. So, `label("my text",origin)` thereafter is exactly the same as `label(texttext("my text"),origin)`. N.B. In the background, `luamplib` redefines `infont` operator so that the right side argument (the font part) is totally ignored. Every string label therefore will be typeset with current  $\TeX$  font. Also take care of char operator in the left side argument, as this might bring unpermitted characters into  $\TeX$ .

**\mplibcodeinherit** Starting with v2.9, `\mplibcodeinherit{enable}` enables the inheritance of variables, constants, and macros defined by previous `mplibcode` chunks. On the contrary, the default value `\mplibcodeinherit{disable}` will make each code chunks being treated as an independent instance, and never affected by previous code chunks.

**\mplibglobaltexttext** To inherit `btex ... etex` labels as well as `metapost` variables, it is necessary to declare `\mplibglobaltexttext{enable}` in advance. On this case, be careful that normal  $\TeX$  boxes can conflict with `btex ... etex` boxes, though this would occur very rarely. Notwithstanding the danger, it is a 'must' option to activate `\mplibglobaltexttext` if you want to use `graph.mp` with `\mplibcodeinherit` functionality.

```
\mplibcodeinherit{enable}
\mplibglobaltexttext{enable}
\everymplib{ beginfig(0);} \everyendmplib{ endfig;}
\mplibcode
  label(btex  $\sqrt{2}$  etex, origin);
  draw fullcircle scaled 20;
  picture pic; pic := currentpicture;
\endmplibcode
\mplibcode
  currentpicture := pic scaled 2;
\endmplibcode
```

**\mplibverbatim** Starting with v2.11, users can issue `\mplibverbatim{enable}`, after which the contents of `mplibcode` environment will be read verbatim. As a result, except for `\mpdim` and `\mpcolor`, all other  $\TeX$  commands outside `btex ... etex` or `verbatimtex ... etex` are not expanded and will be fed literally into the `mplib` process.

**luamplib.cfg** At the end of package loading, `luamplib` searches `luamplib.cfg` and, if found, reads the file in automatically. Frequently used settings such as `\everymplib` or `\mplibforcehmode` are suitable for going into this file.

There are (basically) two formats for metapost: *plain* and *metafun*. By default, the *plain* format is used, but you can set the format to be used by future figures at any time using `\mplibsetformat{<format name>}`.

## 2 Implementation

### 2.1 Lua module

```

1
2 luatexbase.provides_module {
3   name      = "luamplib",
4   version   = "2.20.7",
5   date      = "2021/03/11",
6   description = "Lua package to typeset Metapost with LuaTeX's MPLib.",
7 }
8
9 local format, abs = string.format, math.abs
10
11 local err = function(...) return luatexbase.module_error ("luamplib", format(...)) end
12 local warn = function(...) return luatexbase.module_warning("luamplib", format(...)) end
13 local info = function(...) return luatexbase.module_info ("luamplib", format(...)) end
14

```

Use the `luamplib` namespace, since `mplib` is for the metapost library itself. ConT<sub>E</sub>Xt uses `metapost`.

```

15 luamplib      = luamplib or { }
16 local luamplib = luamplib
17
18 luamplib.showlog = luamplib.showlog or false
19

```

This module is a stripped down version of libraries that are used by ConT<sub>E</sub>Xt. Provide a few “shortcuts” expected by the imported code.

```

20 local tableconcat = table.concat
21 local textsprint  = tex.sprint
22 local textprint   = tex.tprint
23
24 local texget      = tex.get
25 local texgettoks  = tex.gettoks
26 local texgetbox   = tex.getbox
27 local texruntoks  = tex.runtoks

```

We don’t use `tex.scantoks` anymore. See below reagrding `tex.runtoks`.

```

    local texscantoks = tex.scantoks

```

```

28
29 if not texrun toks then
30   err("Your LuaTeX version is too old. Please upgrade it to the latest")
31 end
32
33 local mplib = require ('mplib')
34 local kpse = require ('kpse')
35 local lfs = require ('lfs')
36
37 local lfsattributes = lfs.attributes
38 local lfsisdir = lfs.isdir
39 local lfsmkdir = lfs.mkdir
40 local lfstouch = lfs.touch
41 local iopen = io.open
42

```

Some helper functions, prepared for the case when l-file etc is not loaded.

```

43 local file = file or { }
44 local replacesuffix = file.replacesuffix or function(filename, suffix)
45   return (filename:gsub("%.[%a%d]+$","")) .. "." .. suffix
46 end
47 local stripsuffix = file.stripsuffix or function(filename)
48   return (filename:gsub("%.[%a%d]+$",""))
49 end
50
51 local is_writable = file.is_writable or function(name)
52   if lfsisdir(name) then
53     name = name .. "_luamplib_temp_file_"
54     local fh = iopen(name,"w")
55     if fh then
56       fh:close(); os.remove(name)
57       return true
58     end
59   end
60 end
61 local mk_full_path = lfs.mkdir or function(path)
62   local full = ""
63   for sub in path:gmatch("(/*[^\\"/]+)") do
64     full = full .. sub
65     lfsmkdir(full)
66   end
67 end
68

```

btex ... etex in input .mp files will be replaced in finder. Because of the limitation of MPLib regarding make\_text, we might have to make cache files modified from input files.

```

69 local luamplibtime = kpse.find_file("luamplib.lua")
70 luamplibtime = luamplibtime and lfsattributes(luamplibtime,"modification")
71

```

```

72 local currenttime = os.time()
73
74 local outputdir
75 if lfstouch then
76   local texmfvar = kpse.expand_var('$TEXMFVAR')
77   if texmfvar and texmfvar ~= "" and texmfvar ~= '$TEXMFVAR' then
78     for _,dir in next, texmfvar:explode(os.type == "windows" and ";" or ":") do
79       if not lfsisdir(dir) then
80         mk_full_path(dir)
81       end
82       if is_writable(dir) then
83         local cached = format("%s/luamplib_cache",dir)
84         lfsmkdir(cached)
85         outputdir = cached
86         break
87       end
88     end
89   end
90 end
91 if not outputdir then
92   outputdir = "."
93   for _,v in ipairs(arg) do
94     local t = v:match("%-output%-directory=(.+)")
95     if t then
96       outputdir = t
97       break
98     end
99   end
100 end
101
102 function luamplib.getcachedir(dir)
103   dir = dir:gsub("##", "#")
104   dir = dir:gsub("^~",
105     os.type == "windows" and os.getenv("UserProfile") or os.getenv("HOME"))
106   if lfstouch and dir then
107     if lfsisdir(dir) then
108       if is_writable(dir) then
109         luamplib.cachedir = dir
110       else
111         warn("Directory '"..dir.."'" is not writable!")
112       end
113     else
114       warn("Directory '"..dir.."'" does not exist!")
115     end
116   end
117 end
118

```

Some basic MetaPost files not necessary to make cache files.

```

119 local noneedtoreplace = {

```

```

120 ["boxes.mp"] = true, -- ["format.mp"] = true,
121 ["graph.mp"] = true, ["marith.mp"] = true, ["mfplain.mp"] = true,
122 ["mpost.mp"] = true, ["plain.mp"] = true, ["rboxes.mp"] = true,
123 ["sarith.mp"] = true, ["string.mp"] = true, -- ["TEX.mp"] = true,
124 ["metafun.mp"] = true, ["metafun.mpiv"] = true, ["mp-abck.mpiv"] = true,
125 ["mp-apos.mpiv"] = true, ["mp-asnc.mpiv"] = true, ["mp-bare.mpiv"] = true,
126 ["mp-base.mpiv"] = true, ["mp-blob.mpiv"] = true, ["mp-butt.mpiv"] = true,
127 ["mp-char.mpiv"] = true, ["mp-chem.mpiv"] = true, ["mp-core.mpiv"] = true,
128 ["mp-crop.mpiv"] = true, ["mp-figs.mpiv"] = true, ["mp-form.mpiv"] = true,
129 ["mp-func.mpiv"] = true, ["mp-grap.mpiv"] = true, ["mp-grid.mpiv"] = true,
130 ["mp-grph.mpiv"] = true, ["mp-idea.mpiv"] = true, ["mp-luas.mpiv"] = true,
131 ["mp-mlib.mpiv"] = true, ["mp-node.mpiv"] = true, ["mp-page.mpiv"] = true,
132 ["mp-shap.mpiv"] = true, ["mp-step.mpiv"] = true, ["mp-text.mpiv"] = true,
133 ["mp-tool.mpiv"] = true,
134 }
135 luamplib.noneedtoreplace = noneedtoreplace
136

```

format.mp is much complicated, so specially treated.

```

137 local function replaceformatmp(file,newfile,ofmodify)
138   local fh = ioopen(file,"r")
139   if not fh then return file end
140   local data = fh:read("*all"); fh:close()
141   fh = ioopen(newfile,"w")
142   if not fh then return file end
143   fh:write(
144     "let normalinfont = infont;\n",
145     "primarydef str infont name = rawtexttext(str) enddef;\n",
146     data,
147     "vardef Fmant_(expr x) = rawtexttext(decimal abs x) enddef;\n",
148     "vardef Fexp_(expr x) = rawtexttext(\"$^{\"&decimal x&\"}$\") enddef;\n",
149     "let infont = normalinfont;\n"
150   ); fh:close()
151   lfstouch(newfile,currenttime,ofmodify)
152   return newfile
153 end
154

```

Replace btex ... etex and verbatimtex ... etex in input files, if needed.

```

155 local name_b = "%f[%a_]"
156 local name_e = "%f[^%a_]"
157 local btex_etex = name_b.."btex"..name_e.."s*(.)%s*"..name_b.."etex"..name_e
158 local verbatimtex_etex = name_b.."verbatimtex"..name_e.."s*(.)%s*"..name_b.."etex"..name_e
159
160 local function replaceinputmpfile (name,file)
161   local ofmodify = lfsattributes(file,"modification")
162   if not ofmodify then return file end
163   local cachedir = luamplib.cachedir or outputdir
164   local newfile = name:gsub("%W","_")
165   newfile = cachedir .."/luamplib_input"..newfile
166   if newfile and luamplibtime then

```



```

167     local nf = lfsattributes(newfile)
168     if nf and nf.mode == "file" and
169         ofmodify == nf.modification and luamplibtime < nf.access then
170         return nf.size == 0 and file or newfile
171     end
172 end
173
174 if name == "format.mp" then return replaceformatmp(file,newfile,ofmodify) end
175
176 local fh = ioopen(file,"r")
177 if not fh then return file end
178 local data = fh:read("*all"); fh:close()
179

```

“etex” must be followed by a space or semicolon as specified in Lua<sub>T</sub><sub>E</sub>X manual, which is not the case of standalone MetaPost though.

```

180 local count,cnt = 0,0
181 data, cnt = data:gsub(btex_etex, "btex %1 etex ") -- space
182 count = count + cnt
183 data, cnt = data:gsub(verbatimt看etex, "verbatim %1 etex;") -- semicolon
184 count = count + cnt
185
186 if count == 0 then
187     needtoreplace[name] = true
188     fh = ioopen(newfile,"w");
189     if fh then
190         fh:close()
191         lfstouch(newfile,currenttime,ofmodify)
192     end
193     return file
194 end
195
196 fh = ioopen(newfile,"w")
197 if not fh then return file end
198 fh:write(data); fh:close()
199 lfstouch(newfile,currenttime,ofmodify)
200 return newfile
201 end
202

```

As the finder function for MPLib, use the kpse library and make it behave like as if MetaPost was used. And replace it with cache files if needed.

```

203 local mpkpse = kpse.new(arg[0], "mpost")
204
205 local special_ftype = {
206     pfb = "type1 fonts",
207     enc = "enc files",
208 }
209
210 local function finder(name, mode, ftype)

```

```

211 if mode == "w" then
212     return name
213 else
214     ftype = special_ftype[ftype] or ftype
215     local file = mpkpse:find_file(name,ftype)
216     if file then
217         if not lfstouch or ftype ~= "mp" or noneedtoreplace[name] then
218             return file
219         end
220         return replaceinputmpfile(name,file)
221     end
222     return mpkpse:find_file(name, name:match("%a+$"))
223 end
224 end
225 luamplib.finder = finder
226

```

Create and load MPLib instances. We do not support ancient version of MPLib any more. (Don't know which version of MPLib started to support `make_text` and `run_script`; let the users find it.)

```

227 if tonumber(mplib.version()) <= 1.50 then
228     err("luamplib no longer supports mplib v1.50 or lower. "..
229     "Please upgrade to the latest version of LuaTeX")
230 end
231
232 local preamble = [[
233     boolean mplib ; mplib := true ;
234     let dump = endinput ;
235     let normalfontsize = fontsize;
236     input %s ;
237 ]]
238
239 local function reporterror (result, indeed)
240     if not result then
241         err("no result object returned")
242     else
243         local t, e, l = result.term, result.error, result.log
244         local log = t or l or "no-term"
245         log = log:gsub("(Please type a command or say 'end%')", ""):gsub("\n+", "\n")
246         if result.status > 0 then
247             warn(log)
248             if result.status > 1 then
249                 err(e or "see above messages")
250             end
251         else

```

v2.6.1: now luamplib does not disregard `show` command, even when `luamplib.showlog` is false. Incidentally, it does not raise error but just prints a warning, even if output has no figure.

```

252     if log:find"\n>>" then

```

```

253     warn(log)
254     elseif log:find"%g" then
255         if luamplib.showlog then
256             info(log)
257         elseif indeed and not result.fig then
258             info(log)
259         end
260     end
261 end
262 return log
263 end
264 end
265
266 local function luamplibload (name)
267     local mpx = mplib.new {
268         ini_version = true,
269         find_file   = luamplib.finder,

```

Make use of `make_text` and `run_script`, which will co-operate with Lua $\TeX$ 's `tex.runtoks`. And we provide `numbersystem` option since v2.4. Default value "scaled" can be changed by declaring `\mplibnumbersystem{double}` or `\mplibnumbersystem{decimal}`. See <https://github.com/lualatex/luamplib/issues/21>.

```

270     make_text   = luamplib.maketext,
271     run_script  = luamplib.runscript,
272     math_mode   = luamplib.numbersystem,
273     extensions  = 1,
274 }

```

Append our own MetaPost preamble to the preamble above.

```

275 local preamble = preamble .. luamplib.mplibcodepreamble
276 if luamplib.legacy_verbatimtex then
277     preamble = preamble .. luamplib.legacyverbatimtexpreamble
278 end
279 if luamplib.texttextlabel then
280     preamble = preamble .. luamplib.texttextlabelpreamble
281 end
282 local result
283 if not mpx then
284     result = { status = 99, error = "out of memory" }
285 else
286     result = mpx:execute(format(preamble, replacesuffix(name,"mp")))
287 end
288 reporterror(result)
289 return mpx, result
290 end
291

```

plain or metafun, though we cannot support metafun format fully.

```

292 local currentformat = "plain"
293
294 local function setformat (name)

```

```

295 currentformat = name
296 end
297 luamplib.setformat = setformat
298

```

Here, excute each mplibcode data, ie \begin{mplibcode} ... \end{mplibcode}.

```

299 local function process_indeed (mpx, data)
300   local converted, result = false, {}
301   if mpx and data then
302     result = mpx:execute(data)
303     local log = reporterror(result, true)
304     if log then
305       if result.fig then
306         converted = luamplib.convert(result)
307       else
308         warn("No figure output. Maybe no beginfig/endfig")
309       end
310     end
311   else
312     err("Mem file unloadable. Maybe generated with a different version of mplib?")
313   end
314   return converted, result
315 end
316

```

v2.9 has introduced the concept of “code inherit”

```

317 luamplib.codeinherit = false
318 local mplibinstances = {}
319
320 local function process (data)

```

The workaround of issue #70 seems to be unnecessary, as we use make\_text now.

```

if not data:find(name_b.."beginfig%s*%([%+%-s]*%d[%.%d%s]*%)" then
  data = data .. "beginfig(-1);endfig;"
end

```

```

321 local standalone = not luamplib.codeinherit
322 local currfmt = currentformat .. (luamplib.numbersystem or "scaled")
323 .. tostring(luamplib.texttextlabel) .. tostring(luamplib.legacy_verbatimtex)
324 local mpx = mplibinstances[currfmt]
325 if mpx and standalone then
326   mpx:finish()
327 end
328 if standalone or not mpx then
329   mpx = luamplibload(currentformat)
330   mplibinstances[currfmt] = mpx
331 end
332 return process_indeed(mpx, data)
333 end
334

```

make\_text and some run\_script uses LuaTeX's tex.runtoks, which made possible running TeX code snippets inside \directlua.

```
335 local catlatex = luatexbase.registernumber("catcodetable@latex")
336 local catat11 = luatexbase.registernumber("catcodetable@atletter")
337
```

tex.scantoks sometimes fail to read catcode properly, especially \#, \&, or \%. After some experiment, we dropped using it. Instead, a function containing tex.script seems to work nicely.

```

    local function run_tex_code_no_use (str, cat)
        cat = cat or catlatex
        texscantoks("mplibtmptoks", cat, str)
        texruntoks("mplibtmptoks")
    end

338 local function run_tex_code (str, cat)
339     cat = cat or catlatex
340     texruntoks(function() texsprint(cat, str) end)
341 end
342
```

Indefinite number of boxes are needed for btex ... etex. So starts at somewhat huge number of box registry. Of course, this may conflict with other packages using many many boxes. (When codeinherit feature is enabled, boxes must be globally defined.) But I don't know any reliable way to escape this danger.

```
343 local tex_box_id = 2047

    For conversion of sp to bp.

344 local factor = 65536*(7227/7200)
345
346 local texttext_fmt = [[image(addto currentpicture doublepath unitsquare )].
347 [[xscaled %f yscaled %f shifted (0,-%f) ]].
348 [[withprescript "mplibtexboxid=%i:%f:%f")]]
349
350 local function process_tex_text (str)
351     if str then
352         tex_box_id = tex_box_id + 1
353         local global = luamplib.globaltexttext and "\\global" or ""
354         run_tex_code(format("%s\\setbox%i\\hbox{%s}", global, tex_box_id, str))
355         local box = texgetbox(tex_box_id)
356         local wd = box.width / factor
357         local ht = box.height / factor
358         local dp = box.depth / factor
359         return texttext_fmt:format(wd, ht+dp, dp, tex_box_id, wd, ht+dp)
360     end
361     return ""
362 end
363
```

Make color or xcolor's color expressions usable, with \mpcolor or mplibcolor. These commands should be used with graphical objects.

```

364 local mplibcolor_fmt = [[\begingroup\let\XC@mcolor\relax]]..
365 [[\def\set@color{\global\mplibtmptoks\expandafter{\current@color}}]]..
366 [[\color %s \endgroup]]
367
368 local function process_color (str)
369   if str then
370     if not str:find("{.-}") then
371       str = format("{%s}",str)
372     end
373     run_tex_code(mplibcolor_fmt:format(str), catat11)
374     return format('1 withprescript "MPLibOverrideColor=%s"', texgettoks"mplibtmptoks")
375   end
376   return ""
377 end
378

```

\mpdim is expanded before MPLib process, so code below will not be used for mplibcode data. But who knows anyone would want it in .mp input file. If then, you can say mplibdimen(".5\textwidth") for example.

```

379 local function process_dimen (str)
380   if str then
381     str = str:gsub("{(.+)}", "%1")
382     run_tex_code(format([[ \mplibtmptoks\expandafter{\the\dimexpr %s\relax}]], str))
383     return format("begingroup %s endgroup", texgettoks"mplibtmptoks")
384   end
385   return ""
386 end
387

```

Newly introduced method of processing verbatimtex ... etex. Used when \mpliblegacybehavior{false} is declared.

```

388 local function process_verbatimtex_text (str)
389   if str then
390     run_tex_code(str)
391   end
392   return ""
393 end
394

```

For legacy verbatimtex process. verbatimtex ... etex before beginfig() is not ignored, but the  $\TeX$  code is inserted just before the mplib box. And  $\TeX$  code inside beginfig() ... endfig is inserted after the mplib box.

```

395 local tex_code_pre_mplib = {}
396 luamplib.figid = 1
397 luamplib.in_the_fig = false
398
399 local function legacy_mplibcode_reset ()
400   tex_code_pre_mplib = {}

```

```

401 luamplib.figid = 1
402 end
403
404 local function process_verbatimtex_prefig (str)
405   if str then
406     tex_code_pre_mplib[luamplib.figid] = str
407   end
408   return ""
409 end
410
411 local function process_verbatimtex_infig (str)
412   if str then
413     return format('special "postmpbibverbtx=%s";', str)
414   end
415   return ""
416 end
417
418 local runscript_funcs = {
419   luamplibtext    = process_tex_text,
420   luamplibcolor   = process_color,
421   luamplibdimen   = process_dimen,
422   luamplibprefig  = process_verbatimtex_prefig,
423   luamplibinfig   = process_verbatimtex_infig,
424   luamplibverbtx  = process_verbatimtex_text,
425 }
426

```

For metafun format. see issue #79.

```

427 mp = mp or {}
428 local mp = mp
429 mp.mf_path_reset = mp.mf_path_reset or function() end
430 mp.mf_finish_saving_data = mp.mf_finish_saving_data or function() end
431

```

metafun 2021-03-09 changes crashes luamplib.

```

432 catcodes = catcodes or {}
433 local catcodes = catcodes
434 catcodes.numbers = catcodes.numbers or {}
435 catcodes.numbers.ctxcatcodes = catcodes.numbers.ctxcatcodes or ""
436 catcodes.numbers.texcatcodes = catcodes.numbers.texcatcodes or ""
437 catcodes.numbers.luacatcodes = catcodes.numbers.luacatcodes or ""
438 catcodes.numbers.notcatcodes = catcodes.numbers.notcatcodes or ""
439 catcodes.numbers.vrbcatcodes = catcodes.numbers.vrbcatcodes or ""
440 catcodes.numbers.prtcatcodes = catcodes.numbers.prtcatcodes or ""
441 catcodes.numbers.txtcatcodes = catcodes.numbers.txtcatcodes or ""
442

```

A function from ConT<sub>E</sub>Xt general.

```

443 local function mpprint(buffer,...)
444   for i=1,select("#",...) do
445     local value = select(i,...)

```

```

446     if value ~= nil then
447         local t = type(value)
448         if t == "number" then
449             buffer[#buffer+1] = format("%.16f",value)
450         elseif t == "string" then
451             buffer[#buffer+1] = value
452         elseif t == "table" then
453             buffer[#buffer+1] = "(" .. tableconcat(value,",") .. ")"
454         else -- boolean or whatever
455             buffer[#buffer+1] = tostring(value)
456         end
457     end
458 end
459 end
460
461 function luamplib.runscript (code)
462     local id, str = code:match("(.-){(.+)}")
463     if id and str and str ~= "" then
464         local f = runscript_funcs[id]
465         if f then
466             local t = f(str)
467             if t then return t end
468         end
469     end
470     local f = loadstring(code)
471     if type(f) == "function" then
472         local buffer = {}
473         function mp.print(...)
474             mpprint(buffer,...)
475         end
476         local result = f()
477         buffer = tableconcat(buffer,"")
478         if buffer and buffer ~= "" then
479             return buffer
480         end
481         return result or ""
482     end
483     return ""
484 end
485

```

make\_text must be one liner, so comment sign is not allowed.

```

486 local function protecttexcontents (str)
487     return str:gsub("\\%", "\0PerCent\0")
488           :gsub("%%.-\n", "")
489           :gsub("%%.-$", "")
490           :gsub("%zPerCent%z", "\\%")
491           :gsub("%s+", " ")
492 end
493

```



```

494 luamplib.legacy_verbatimex = true
495
496 function luamplib.maketext (str, what)
497   if str and str ~= "" then
498     str = protecttexcontents(str)
499     if what == 1 then
500       if not str:find("\\documentclass"..name_e) and
501         not str:find("\\begin%s*{document}") and
502         not str:find("\\documentstyle"..name_e) and
503         not str:find("\\usepackage"..name_e) then
504         if luamplib.legacy_verbatimex then
505           if luamplib.in_the_fig then
506             return process_verbatimex_infig(str)
507           else
508             return process_verbatimex_prefig(str)
509           end
510         else
511           return process_verbatimex_text(str)
512         end
513       end
514     else
515       return process_tex_text(str)
516     end
517   end
518   return ""
519 end
520

```

#### Our MetaPost preambles

```

521 local mplibcodepreamble = [[
522 texscriptmode := 2;
523 def rawtexttext (expr t) = runscript("luamplibtext{"&t&"}") enddef;
524 def mplibcolor (expr t) = runscript("luamplibcolor{"&t&"}") enddef;
525 def mplibdimen (expr t) = runscript("luamplibdimen{"&t&"}") enddef;
526 def VerbatimTeX (expr t) = runscript("luamplibverbtex{"&t&"}") enddef;
527 if known context_mlib:
528   defaultfont := "cmtt10";
529   let infont = normalinfont;
530   let fontsize = normalfontsize;
531   vardef thelabel@#(expr p,z) =
532     if string p :
533       thelabel@#(p infont defaultfont scaled defaultscale,z)
534     else :
535       p shifted (z + labeloffset*mfun_laboff@# -
536         (mfun_labxf@#*lrcorner p + mfun_labyf@#*ulcorner p +
537         (1-mfun_labxf@#-mfun_labyf@#)*llcorner p))
538     fi
539   enddef;
540 def graphicstext primary filename =
541   if (readfrom filename = EOF):

```

```

542     errmessage "Please prepare '"&filename&'" in advance with"&
543     " 'pstoedit -ssp -dt -f mpost yourfile.ps "&filename&""";
544     fi
545     closefrom filename;
546     def data_mpy_file = filename enddef;
547     mfun_do_graphic_text (filename)
548 enddef;
549 else:
550     vardef texttext@# (text t) = rawtexttext (t) enddef;
551 fi
552 def externalfigure primary filename =
553     draw rawtexttext("\includegraphics{'& filename &}")
554 enddef;
555 def TEX = texttext enddef;
556 ]]
557 luamplib.mplibcodepreamble = mpplibcodepreamble
558
559 local legacyverbatimtexpreamble = [[
560 def specialVerbatimTeX (text t) = runscript("luamplibprefig{'&t&}") enddef;
561 def normalVerbatimTeX (text t) = runscript("luamplibinfig{'&t&}") enddef;
562 let VerbatimTeX = specialVerbatimTeX;
563 extra_beginfig := extra_beginfig & " let VerbatimTeX = normalVerbatimTeX;"&
564 "runscript(" &ditto& "luamplib.in_the_fig=true" &ditto& ");";
565 extra_endfig := extra_endfig & " let VerbatimTeX = specialVerbatimTeX;"&
566 "runscript(" &ditto&
567 "luamplib.in_the_fig=false luamplib.figid=luamplib.figid+1" &ditto& ");";
568 ]]
569 luamplib.legacyverbatimtexpreamble = legacyverbatimtexpreamble
570
571 local texttextlabelpreamble = [[
572 primarydef s infont f = rawtexttext(s) enddef;
573 def fontsize expr f =
574     begingroup
575     save size; numeric size;
576     size := mpplibdimen("1em");
577     if size = 0: 10pt else: size fi
578     endgroup
579 enddef;
580 ]]
581 luamplib.texttextlabelpreamble = texttextlabelpreamble
582

```

When \mplibverbatim is enabled, do not expand mpplibcode data.

```

583 luamplib.verbatiminput = false
584

```

Do not expand btex ... etex, verbatimtex ... etex, and string expressions.

```

585 local function protect_expansion (str)
586     if str then
587         str = str:gsub("\\", "!!!Control!!!")
588         :gsub("%%", "!!!Comment!!!")

```

```

589         :gsub("#", "!!!HashSign!!!")
590         :gsub("{", "!!!LBrace!!!")
591         :gsub("}", "!!!RBrace!!!")
592     return format("\unexpanded{%s}",str)
593 end
594 end
595
596 local function unprotect_expansion (str)
597     if str then
598         return str:gsub("!!!Control!!!", "\\")
599         :gsub("!!!Comment!!!", "%")
600         :gsub("!!!HashSign!!!", "#")
601         :gsub("!!!LBrace!!!", "{")
602         :gsub("!!!RBrace!!!", "}")
603     end
604 end
605
606 local function process_mplibcode (data)
    This is needed for legacy behavior regarding verbatimtex
607     legacy_mplibcode_reset()
608
609     local everymplib    = texgettoks'everymplibtoks' or ''
610     local everyendmplib = texgettoks'everyendmplibtoks' or ''
611     data = format("\n%s\n%s\n%s\n",everymplib, data, everyendmplib)
612     data = data:gsub("\r","\n")
613
614     data = data:gsub("\mpcolor%s+{.-%b{}}", "mplibcolor(\\"%1\\)")
615     data = data:gsub("\mpdim%s+{b{}}", "mplibdimen(\\"%1\\)")
616     data = data:gsub("\mpdim%s+(\\"%a+)", "mplibdimen(\\"%1\\)")
617
618     data = data:gsub(btex_etex, function(str)
619         return format("btex %s etex ", -- space
620             luamplib.verbatiminput and str or protect_expansion(str))
621     end)
622     data = data:gsub(verbatimtex_etex, function(str)
623         return format("verbatimtex %s etex;", -- semicolon
624             luamplib.verbatiminput and str or protect_expansion(str))
625     end)
626

```

If not mplibverbatim, expand mplibcode data, so that users can use  $\TeX$  codes in it. It has turned out that no comment sign is allowed.

```

627     if not luamplib.verbatiminput then
628         data = data:gsub("\".-\\\"", protect_expansion)
629
630         data = data:gsub("\\"%%", "\0Percent\0")
631         data = data:gsub("%%. -\\n", "")
632         data = data:gsub("%zPercent%z", "\\"%%)
633
634     run_tex_code(format("\mplibtmptoks\expanded{{%s}}",data))

```

```

635 data = texgettoks"mplibtmptoks"
Next line to address issue #55
636 data = data:gsub("##", "#")
637 data = data:gsub("\.-\\"", unprotect_expansion)
638 data = data:gsub(btex_etex, function(str)
639     return format("btex %s etex", unprotect_expansion(str))
640 end)
641 data = data:gsub(verbatimtex_etex, function(str)
642     return format("verbatimtex %s etex", unprotect_expansion(str))
643 end)
644 end
645
646 process(data)
647 end
648 luamplib.process_mplibcode = process_mplibcode
649

```

For parsing prescript materials.

```

650 local further_split_keys = {
651     mplibtexboxid = true,
652     sh_color_a    = true,
653     sh_color_b    = true,
654 }
655
656 local function script2table(s)
657     local t = {}
658     for _,i in ipairs(s:explode("\13+")) do
659         local k,v = i:match("(.-)=(.*)") -- v may contain = or empty.
660         if k and v and k ~= "" then
661             if further_split_keys[k] then
662                 t[k] = v:explode(":")
663             else
664                 t[k] = v
665             end
666         end
667     end
668     return t
669 end
670

```

Codes below for inserting PDF literals are mostly from ConTeXt general, with small changes when needed.

```

671 local function getobjects(result,figure,f)
672     return figure:objects()
673 end
674
675 local function convert(result, flusher)
676     luamplib.flush(result, flusher)
677     return true -- done
678 end

```

```

679 luamplib.convert = convert
680
681 local function pdf_startfigure(n,llx,lly,urx,ury)
682   texsprintf(format("\mplibstarttoPDF{%f}{%f}{%f}",llx,lly,urx,ury))
683 end
684
685 local function pdf_stopfigure()
686   texsprintf("\mplibstoptoPDF")
687 end
688
        tex.tprint with catcode regime -2, as sometimes # gets doubled in the argument of
pdfliteral.
689 local function pdf_literalcode(fmt,...) -- table
690   texsprintf({"\mplibtoPDF{",{-2,format(fmt,...)},{"}"}))
691 end
692
693 local function pdf_textfigure(font,size,text,width,height,depth)
694   text = text:gsub(".",function(c)
695     return format("\hbox{\char%i}",string.byte(c)) -- kerning happens in metapost
696   end)
697   texsprintf(format("\mplibtexttext{%s}{%f}{%s}{%s}{%f}",font,size,text,0,-( 7200/ 7227)/65536*depth))
698 end
699
700 local bend_tolerance = 131/65536
701
702 local rx, sx, sy, ry, tx, ty, divider = 1, 0, 0, 1, 0, 0, 1
703
704 local function pen_characteristics(object)
705   local t = mplib.pen_info(object)
706   rx, ry, sx, sy, tx, ty = t.rx, t.ry, t.sx, t.sy, t.tx, t.ty
707   divider = sx*sy - rx*ry
708   return not (sx==1 and rx==0 and ry==0 and sy==1 and tx==0 and ty==0), t.width
709 end
710
711 local function concat(px, py) -- no tx, ty here
712   return (sy*px-ry*py)/divider,(sx*py-rx*px)/divider
713 end
714
715 local function curved(ith,pth)
716   local d = pth.left_x - ith.right_x
717   if abs(ith.right_x - ith.x_coord - d) <= bend_tolerance and abs(pth.x_coord - pth.left_x - d) <= bend_tolerance then
718     d = pth.left_y - ith.right_y
719     if abs(ith.right_y - ith.y_coord - d) <= bend_tolerance and abs(pth.y_coord - pth.left_y - d) <= bend_tolerance then
720       return false
721     end
722   end
723   return true
724 end
725

```

```

726 local function flushnormalpath(path,open)
727   local pth, ith
728   for i=1,#path do
729     pth = path[i]
730     if not ith then
731       pdf_literalcode("%f %f m",pth.x_coord,pth.y_coord)
732     elseif curved(ith,pth) then
733       pdf_literalcode("%f %f %f %f %f c",ith.right_x,ith.right_y,pth.left_x,pth.left_y,pth.x_coord,pth.y_coord)
734     else
735       pdf_literalcode("%f %f l",pth.x_coord,pth.y_coord)
736     end
737     ith = pth
738   end
739   if not open then
740     local one = path[1]
741     if curved(pth,one) then
742       pdf_literalcode("%f %f %f %f %f %f c",pth.right_x,pth.right_y,one.left_x,one.left_y,one.x_coord,one.y_coord )
743     else
744       pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
745     end
746   elseif #path == 1 then -- special case .. draw point
747     local one = path[1]
748     pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
749   end
750 end
751
752 local function flushconcatpath(path,open)
753   pdf_literalcode("%f %f %f %f %f %f cm", sx, rx, ry, sy, tx ,ty)
754   local pth, ith
755   for i=1,#path do
756     pth = path[i]
757     if not ith then
758       pdf_literalcode("%f %f m",concat(pth.x_coord,pth.y_coord))
759     elseif curved(ith,pth) then
760       local a, b = concat(ith.right_x,ith.right_y)
761       local c, d = concat(pth.left_x,pth.left_y)
762       pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(pth.x_coord, pth.y_coord))
763     else
764       pdf_literalcode("%f %f l",concat(pth.x_coord, pth.y_coord))
765     end
766     ith = pth
767   end
768   if not open then
769     local one = path[1]
770     if curved(pth,one) then
771       local a, b = concat(pth.right_x,pth.right_y)
772       local c, d = concat(one.left_x,one.left_y)
773       pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(one.x_coord, one.y_coord))
774     else
775       pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))

```

```

776     end
777 elseif #path == 1 then -- special case .. draw point
778     local one = path[1]
779     pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
780 end
781 end
782
    dvipdfmx is supported, though nobody seems to use it.
783 local pdfoutput = tonumber(texget("outputmode")) or tonumber(texget("pdfoutput"))
784 local pdfmode = pdfoutput > 0
785
786 local function start_pdf_code()
787     if pdfmode then
788         pdf_literalcode("q")
789     else
790         texsprint("\\special{pdf:bcontent}") -- dvipdfmx
791     end
792 end
793 local function stop_pdf_code()
794     if pdfmode then
795         pdf_literalcode("Q")
796     else
797         texsprint("\\special{pdf:econtent}") -- dvipdfmx
798     end
799 end
800

```

Now we process hboxes created from `btex ... etex` or `texttext(...)` or `TEX(...)`, all being the same internally.

```

801 local function put_tex_boxes (object,prescript)
802     local box = prescript.mplibtexboxid
803     local n,tw,th = box[1],tonumber(box[2]),tonumber(box[3])
804     if n and tw and th then
805         local op = object.path
806         local first, second, fourth = op[1], op[2], op[4]
807         local tx, ty = first.x_coord, first.y_coord
808         local sx, rx, ry, sy = 1, 0, 0, 1
809         if tw ~= 0 then
810             sx = (second.x_coord - tx)/tw
811             rx = (second.y_coord - ty)/tw
812             if sx == 0 then sx = 0.00001 end
813         end
814         if th ~= 0 then
815             sy = (fourth.y_coord - ty)/th
816             ry = (fourth.x_coord - tx)/th
817             if sy == 0 then sy = 0.00001 end
818         end
819         start_pdf_code()
820         pdf_literalcode("%f %f %f %f %f %f cm",sx,rx,ry,sy,tx,ty)
821         texsprint(format("\\mplibputtextbox{%i}",n))
822     end
823 end
824

```

```

822     stop_pdf_code()
823 end
824 end
825
      Colors and Transparency
826 local pdf_objs = {}
827 local token, getpagers, setpagers = newtoken or token
828 local pgf = { bye = "pgfutil@everybye", extgs = "pgf@sys@addpdfresource@extgs@plain" }
829
830 if pdfmode then -- repeat luaotfload-colors
831   getpagers = pdf.getpagersources or function() return pdf.pagersources end
832   setpagers = pdf.setpagersources or function(s) pdf.pagersources = s end
833 else
834   texsprint("\\special{pdf:obj @MPlibTr<<>>}",
835             "\\special{pdf:obj @MPlibSh<<>>}")
836 end
837
838 local function update_pdfobjs (os)
839   local on = pdf_objs[os]
840   if on then
841     return on, false
842   end
843   if pdfmode then
844     on = pdf.immediateobj(os)
845   else
846     on = pdf_objs.cnt or 0
847     pdf_objs.cnt = on + 1
848   end
849   pdf_objs[os] = on
850   return on, true
851 end
852
853 local transparency_modes = { [0] = "Normal",
854   "Normal",      "Multiply",    "Screen",      "Overlay",
855   "SoftLight",   "HardLight",   "ColorDodge",  "ColorBurn",
856   "Darken",      "Lighten",     "Difference",  "Exclusion",
857   "Hue",         "Saturation",   "Color",       "Luminosity",
858   "Compatible",
859 }
860
861 local function update_tr_res(res, mode, opaq)
862   local os = format("<</BM /%s/ca %.3f/CA %.3f/AIS false>>", mode, opaq, opaq)
863   local on, new = update_pdfobjs(os)
864   if new then
865     if pdfmode then
866       res = format("%s/MPlibTr%i %i 0 R", res, on, on)
867     else
868       if pgf.loaded then
869         texsprint(format("\\csname %s\\endcsname{/MPlibTr%i%s}", pgf.extgs, on, os))

```



```

870     else
871         texsprint(format("\\special{pdf:put @MPlibTr<</MPlibTr%i%s>>}",on,os))
872     end
873 end
874 end
875 return res,on
876 end
877
878 local function tr_pdf_pageresources(mode,opaq)
879 if token and pgf.bye and not pgf.loaded then
880     pgf.loaded = token.create(pgf.bye).cmdname == "assign_toks"
881     pgf.bye     = pgf.loaded and pgf.bye
882 end
883 local res, on_on, off_on = "", nil, nil
884 res, off_on = update_tr_res(res, "Normal", 1)
885 res, on_on  = update_tr_res(res, mode, opaq)
886 if pdfmode then
887     if res ~= "" then
888         if pgf.loaded then
889             texsprint(format("\\csname %s\\endcsname{%s}", pgf.extgs, res))
890         else
891             local tpr, n = getpageres() or "", 0
892             tpr, n = tpr:gsub("/ExtGState<<", "%1"..res)
893             if n == 0 then
894                 tpr = format("%s/ExtGState<<%s>>", tpr, res)
895             end
896             setpageres(tpr)
897         end
898     end
899 else
900     if not pgf.loaded then
901         texsprint(format("\\special{pdf:put @resources<</ExtGState @MPlibTr>>}"))
902     end
903 end
904 return on_on, off_on
905 end
906

```

Shading with metafun format. (maybe legacy way)

```

907 local shading_res
908
909 local function shading_initialize ()
910     shading_res = {}
911     if pdfmode and luatexbase.callbacktypes.finish_pdffile then -- ltuatex
912         local shading_obj = pdf.reserveobj()
913         setpageres(format("%s/Shading %i 0 R",getpageres() or "",shading_obj))
914         luatexbase.add_to_callback("finish_pdffile", function()
915             pdf.immediateobj(shading_obj,format("<<%s>>",tableconcat(shading_res)))
916             end, "luamplib.finish_pdffile")
917         pdf_objs.finishpdf = true

```

```

918 end
919 end
920
921 local function sh_pdfpageresources(shtype, domain, colorspace, colora, colorb, coordinates)
922 if not shading_res then shading_initialize() end
923 local os = format("<</FunctionType 2/Domain [ %s ]/C0 [ %s ]/C1 [ %s ]/N 1>>",
924 domain, colora, colorb)
925 local funcobj = pdfmode and format("%i 0 R", update_pdfobjs(os)) or os
926 os = format("<</ShadingType %i/ColorSpace /%s/Function %s/Coords [ %s ]/Extend [ true true ]/AntiAlias true>>",
927 shtype, colorspace, funcobj, coordinates)
928 local on, new = update_pdfobjs(os)
929 if pdfmode then
930 if new then
931 local res = format("/MPLibSh%i %i 0 R", on, on)
932 if pdf_objs.finishpdf then
933 shading_res[#shading_res+1] = res
934 else
935 local pageres = getpageres() or ""
936 if not pageres:find("/Shading<<.*>>") then
937 pageres = pageres.."/Shading<<>>"
938 end
939 pageres = pageres:gsub("/Shading<<","%1"..res)
940 setpageres(pageres)
941 end
942 end
943 else
944 if new then
945 texsprint(format("\\special{pdf:put @MPLibSh<</MPLibSh%i%s>>}", on, os))
946 end
947 texsprint(format("\\special{pdf:put @resources<</Shading @MPLibSh>>}"))
948 end
949 return on
950 end
951
952 local function color_normalize(ca, cb)
953 if #cb == 1 then
954 if #ca == 4 then
955 cb[1], cb[2], cb[3], cb[4] = 0, 0, 0, 1-cb[1]
956 else -- #ca = 3
957 cb[1], cb[2], cb[3] = cb[1], cb[1], cb[1]
958 end
959 elseif #cb == 3 then -- #ca == 4
960 cb[1], cb[2], cb[3], cb[4] = 1-cb[1], 1-cb[2], 1-cb[3], 0
961 end
962 end
963
964 local prev_override_color
965
966 local function do_preobj_color(object, prescript)

```

```

transparency
967 local opaq = prescript and prescript.tr_transparency
968 local tron_no, troff_no
969 if opaq then
970     local mode = prescript.tr_alternative or 1
971     mode = transparency_modes[tonumber(mode)]
972     tron_no, troff_no = tr_pdf_pageresources(mode, opaq)
973     pdf_literalcode("/MPLibTr%i gs", tron_no)
974 end

color
975 local override = prescript and prescript.MPLibOverrideColor
976 if override then
977     if pdfmode then
978         pdf_literalcode(override)
979         override = nil
980     else
981         texsprint(format("\\special{color push %s}", override))
982         prev_override_color = override
983     end
984 else
985     local cs = object.color
986     if cs and #cs > 0 then
987         pdf_literalcode(luamplib.colorconverter(cs))
988         prev_override_color = nil
989     elseif not pdfmode then
990         override = prev_override_color
991         if override then
992             texsprint(format("\\special{color push %s}", override))
993         end
994     end
995 end

shading
996 local sh_type = prescript and prescript.sh_type
997 if sh_type then
998     local domain = prescript.sh_domain
999     local centera = prescript.sh_center_a:explode()
1000     local centerb = prescript.sh_center_b:explode()
1001     for _, t in pairs({centera, centerb}) do
1002         for i, v in ipairs(t) do
1003             t[i] = format("%.4f", v)
1004         end
1005     end
1006     centera = tableconcat(centera, " ")
1007     centerb = tableconcat(centerb, " ")
1008     local colora = prescript.sh_color_a or {0};
1009     local colorb = prescript.sh_color_b or {1};
1010     for _, t in pairs({colora, colorb}) do
1011         for i, v in ipairs(t) do

```

```

1012         t[i] = format("%.3f",v)
1013     end
1014 end
1015 if #colora > #colorb then
1016     color_normalize(colora,colorb)
1017 elseif #colorb > #colora then
1018     color_normalize(colorb,colora)
1019 end
1020 local colorspace
1021 if #colorb == 1 then colorspace = "DeviceGray"
1022 elseif #colorb == 3 then colorspace = "DeviceRGB"
1023 elseif #colorb == 4 then colorspace = "DeviceCMYK"
1024 else return troff_no,override
1025 end
1026 colora = tableconcat(colora, " ")
1027 colorb = tableconcat(colorb, " ")
1028 local shade_no
1029 if sh_type == "linear" then
1030     local coordinates = tableconcat({centera,centerb}," ")
1031     shade_no = sh_pdfpageresources(2,domain,colorspace,colora,colorb,coordinates)
1032 elseif sh_type == "circular" then
1033     local radiusa = format("%.f",prescript.sh_radius_a)
1034     local radiusb = format("%.f",prescript.sh_radius_b)
1035     local coordinates = tableconcat({centera,radiusa,centerb,radiusb}," ")
1036     shade_no = sh_pdfpageresources(3,domain,colorspace,colora,colorb,coordinates)
1037 end
1038 pdf_literalcode("q /Pattern cs")
1039 return troff_no,override,shade_no
1040 end
1041 return troff_no,override
1042 end
1043
1044 local function do_postobj_color(tr,over,sh)
1045     if sh then
1046         pdf_literalcode("W n /MPLibSh%s sh Q",sh)
1047     end
1048     if over then
1049         texsprintf("\\special{color pop}")
1050     end
1051     if tr then
1052         pdf_literalcode("/MPLibTr%i gs",tr)
1053     end
1054 end
1055

```

Finally, flush figures by inserting PDF literals.

```

1056 local function flush(result,flusher)
1057     if result then
1058         local figures = result.fig
1059         if figures then

```

```

1060     for f=1, #figures do
1061         info("flushing figure %s",f)
1062         local figure = figures[f]
1063         local objects = getobjects(result,figure,f)
1064         local fignum = tonumber(figure:filename():match("([%d]+)$") or figure:charcode() or 0)
1065         local miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
1066         local bbox = figure:boundingbox()
1067         local llx, lly, urx, ury = bbox[1], bbox[2], bbox[3], bbox[4] -- faster than unpack
1068         if urx < llx then

```

luamplib silently ignores this invalid figure for those that do not contain `beginfig ... endfig`.  
(issue #70) Original code of ConTeXt general was:

```

-- invalid
pdf_startfigure(fignum,0,0,0,0)
pdf_stopfigure()

```

```

1069     else

```

For legacy behavior. Insert ‘pre-fig’  $\TeX$  code here, and prepare a table for ‘in-fig’ codes.

```

1070         if tex_code_pre_mplib[f] then
1071             texpstr(tex_code_pre_mplib[f])
1072         end
1073         local TeX_code_bot = {}
1074         pdf_startfigure(fignum,llx,lly,urx,ury)
1075         start_pdf_code()
1076         if objects then
1077             local savedpath = nil
1078             local savedhtap = nil
1079             for o=1,#objects do
1080                 local object      = objects[o]
1081                 local objecttype  = object.type

```

The following 5 lines are part of `btex...etex` patch. Again, colors are processed at this stage.

```

1082         local prescript      = object.prescript
1083         prescript = prescript and script2table(prescript) -- prescript is now a table
1084         local tr_opaq,cr_over,shade_no = do_preobj_color(object,prescript)
1085         if prescript and prescript.mplibtexboxid then
1086             put_tex_boxes(object,prescript)
1087         elseif objecttype == "start_bounds" or objecttype == "stop_bounds" then --skip
1088             elseif objecttype == "start_clip" then
1089                 local evenodd = not object.istext and object.postscript == "evenodd"
1090                 start_pdf_code()
1091                 flushnormalpath(object.path,false)
1092                 pdf_literalcode(evenodd and "W* n" or "W n")
1093             elseif objecttype == "stop_clip" then
1094                 stop_pdf_code()
1095                 miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
1096             elseif objecttype == "special" then

```

Collect TeX codes that will be executed after flushing. Legacy behavior.

```

1097         if prescript and prescript.postmplibverbtex then
1098             TeX_code_bot[#TeX_code_bot+1] = prescript.postmplibverbtex
1099         end
1100     elseif objecttype == "text" then
1101         local ot = object.transform -- 3,4,5,6,1,2
1102         start_pdf_code()
1103         pdf_literalcode("%f %f %f %f %f %f cm",ot[3],ot[4],ot[5],ot[6],ot[1],ot[2])
1104         pdf_textfigure(object.font,object.dsize,object.text,object.width,object.height,object.depth)
1105         stop_pdf_code()
1106     else
1107         local evenodd, collect, both = false, false, false
1108         local postscript = object.postscript
1109         if not object.istext then
1110             if postscript == "evenodd" then
1111                 evenodd = true
1112             elseif postscript == "collect" then
1113                 collect = true
1114             elseif postscript == "both" then
1115                 both = true
1116             elseif postscript == "eoboth" then
1117                 evenodd = true
1118                 both = true
1119             end
1120         end
1121         if collect then
1122             if not savedpath then
1123                 savedpath = { object.path or false }
1124                 savedhtap = { object.htap or false }
1125             else
1126                 savedpath[#savedpath+1] = object.path or false
1127                 savedhtap[#savedhtap+1] = object.htap or false
1128             end
1129         else
1130             local ml = object.miterlimit
1131             if ml and ml ~= miterlimit then
1132                 miterlimit = ml
1133                 pdf_literalcode("%f M",ml)
1134             end
1135             local lj = object.linejoin
1136             if lj and lj ~= linejoin then
1137                 linejoin = lj
1138                 pdf_literalcode("%i j",lj)
1139             end
1140             local lc = object.linecap
1141             if lc and lc ~= linecap then
1142                 linecap = lc
1143                 pdf_literalcode("%i J",lc)
1144             end

```

```

1145         local dl = object.dash
1146     if dl then
1147         local d = format("[%s] %f d",tableconcat(dl.dashes or {}, " "),dl.offset)
1148         if d ~= dashed then
1149             dashed = d
1150             pdf_literalcode(dashed)
1151         end
1152     elseif dashed then
1153         pdf_literalcode("[ ] 0 d")
1154         dashed = false
1155     end
1156     local path = object.path
1157     local transformed, penwidth = false, 1
1158     local open = path and path[1].left_type and path[#path].right_type
1159     local pen = object.pen
1160     if pen then
1161         if pen.type == 'elliptical' then
1162             transformed, penwidth = pen_characteristics(object) -- boolean, value
1163             pdf_literalcode("%f w",penwidth)
1164             if objecttype == 'fill' then
1165                 objecttype = 'both'
1166             end
1167         else -- calculated by mplib itself
1168             objecttype = 'fill'
1169         end
1170     end
1171     if transformed then
1172         start_pdf_code()
1173     end
1174     if path then
1175         if savedpath then
1176             for i=1,#savedpath do
1177                 local path = savedpath[i]
1178                 if transformed then
1179                     flushconcatpath(path,open)
1180                 else
1181                     flushnormalpath(path,open)
1182                 end
1183             end
1184             savedpath = nil
1185         end
1186         if transformed then
1187             flushconcatpath(path,open)
1188         else
1189             flushnormalpath(path,open)
1190         end

```

Change from ConTeXt general: there was color stuffs.

```

1191         if not shade_no then -- conflict with shading
1192             if objecttype == "fill" then

```

```

1193         pdf_literalcode(evenodd and "h f*" or "h f")
1194     elseif objecttype == "outline" then
1195         if both then
1196             pdf_literalcode(evenodd and "h B*" or "h B")
1197         else
1198             pdf_literalcode(open and "S" or "h S")
1199         end
1200     elseif objecttype == "both" then
1201         pdf_literalcode(evenodd and "h B*" or "h B")
1202     end
1203 end
1204 end
1205 if transformed then
1206     stop_pdf_code()
1207 end
1208 local path = object.htap
1209 if path then
1210     if transformed then
1211         start_pdf_code()
1212     end
1213     if savedhtap then
1214         for i=1,#savedhtap do
1215             local path = savedhtap[i]
1216             if transformed then
1217                 flushconcatpath(path,open)
1218             else
1219                 flushnormalpath(path,open)
1220             end
1221         end
1222         savedhtap = nil
1223         evenodd = true
1224     end
1225     if transformed then
1226         flushconcatpath(path,open)
1227     else
1228         flushnormalpath(path,open)
1229     end
1230     if objecttype == "fill" then
1231         pdf_literalcode(evenodd and "h f*" or "h f")
1232     elseif objecttype == "outline" then
1233         pdf_literalcode(open and "S" or "h S")
1234     elseif objecttype == "both" then
1235         pdf_literalcode(evenodd and "h B*" or "h B")
1236     end
1237     if transformed then
1238         stop_pdf_code()
1239     end
1240 end
1241 end
1242 end

```



Added to ConTeXt general: color stuff. And execute legacy verbatimex code.

```

1243         do_postobj_color(tr_opaq,cr_over,shade_no)
1244     end
1245 end
1246 stop_pdf_code()
1247 pdf_stopfigure()
1248 if #TeX_code_bot > 0 then texsprint(TeX_code_bot) end
1249 end
1250 end
1251 end
1252 end
1253 end
1254 luamplib.flush = flush
1255
1256 local function colorconverter(cr)
1257     local n = #cr
1258     if n == 4 then
1259         local c, m, y, k = cr[1], cr[2], cr[3], cr[4]
1260         return format("%.3f %.3f %.3f %.3f k %.3f %.3f %.3f %.3f K",c,m,y,k,c,m,y,k), "0 g 0 G"
1261     elseif n == 3 then
1262         local r, g, b = cr[1], cr[2], cr[3]
1263         return format("%.3f %.3f %.3f rg %.3f %.3f %.3f RG",r,g,b,r,g,b), "0 g 0 G"
1264     else
1265         local s = cr[1]
1266         return format("%.3f g %.3f G",s,s), "0 g 0 G"
1267     end
1268 end
1269 luamplib.colorconverter = colorconverter

```

## 2.2 T<sub>E</sub>X package

First we need to load some packages.

```

1270 \bgroup\expandafter\expandafter\expandafter\egroup
1271 \expandafter\ifx\csname selectfont\endcsname\relax
1272     \input ltluatex
1273 \else
1274     \NeedsTeXFormat{LaTeX2e}
1275     \ProvidesPackage{luamplib}
1276     [2021/03/11 v2.20.7 mplib package for LuaTeX]
1277     \ifx\newluafunction\undefined
1278     \input ltluatex
1279     \fi
1280 \fi

```

Loading of lua code.

```

1281 \directlua{require("luamplib")}

```

Support older engine. Seems we don't need it, but no harm.

```

1282 \ifx\pdfoutput\undefined

```

```

1283 \let\pdfoutput\outputmode
1284 \protected\def\pdfliteral{\pdfextension literal}
1285 \fi

```

Unfortunately there are still packages out there that think it is a good idea to manually set `\pdfoutput` which defeats the above branch that defines `\pdfliteral`. To cover that case we need an extra check.

```

1286 \ifx\pdfliteral\undefined
1287 \protected\def\pdfliteral{\pdfextension literal}
1288 \fi

```

Set the format for metapost.

```

1289 \def\mplibsetformat#1{\directlua{luamplib.setformat("#1")}}

```

`luamplib` works in both PDF and DVI mode, but only DVIPDFMx is supported currently among a number of DVI tools. So we output a warning.

```

1290 \ifnum\pdfoutput>0
1291 \let\mplibtoPDF\pdfliteral
1292 \else
1293 \def\mplibtoPDF#1{\special{pdf:literal direct #1}}
1294 \ifcsname PackageWarning\endcsname
1295 \PackageWarning{luamplib}{take dvipdfmx path, no support for other dvi tools currently.}
1296 \else
1297 \write128{}
1298 \write128{luamplib Warning: take dvipdfmx path, no support for other dvi tools currently.}
1299 \write128{}
1300 \fi
1301 \fi

```

Make `mplibcode` typesetted always in horizontal mode.

```

1302 \def\mplibforcehmode{\let\prependtomplibbox\leavevmode}
1303 \def\mplibnoforcehmode{\let\prependtomplibbox\relax}
1304 \mplibnoforcehmode

```

Catcode. We want to allow comment sign in `mplibcode`.

```

1305 \def\mplibsetupcatcodes{%
1306 %catcode'\={12 %catcode'\}=12
1307 \catcode'\#=12 \catcode'\^=12 \catcode'\~=12 \catcode'\_ =12
1308 \catcode'\&=12 \catcode'\$=12 \catcode'\%=12 \catcode'\^M=12
1309 }

```

Make `btex...etex` box zero-metric.

```

1310 \def\mplibputtextbox#1{\vbox to 0pt{\vss\hbox to 0pt{\raise\dp#1\copy#1\hss}}}

```

The Plain-specific stuff.

```

1311 \bgroup\expandafter\expandafter\expandafter\egroup
1312 \expandafter\ifx\csname selectfont\endcsname\relax
1313 \def\mplibcode{%
1314 \begingroup
1315 \begingroup
1316 \mplibsetupcatcodes
1317 \mplibdocode

```

```

1318 }
1319 \long\def\mplibdocode#1\endmplibcode{%
1320   \endgroup
1321   \directlua{luamplib.process_mplibcode([====\unexpanded{#1}====])}%
1322   \endgroup
1323 }
1324 \else

    The  $\TeX$ -specific part: a new environment.

1325 \newenvironment{mplibcode}{%
1326   \mplibtmptoks}\ltxdomplibcode
1327 }{}
1328 \def\ltxdomplibcode{%
1329   \begingroup
1330   \mplibsetupcatcodes
1331   \ltxdomplibcodeindeed
1332 }
1333 \def\mplib@mplibcode{mplibcode}
1334 \long\def\ltxdomplibcodeindeed#1\end#2{%
1335   \endgroup
1336   \mplibtmptoks\expandafter{\the\mplibtmptoks#1}%
1337   \def\mplibtemp@a{#2}%
1338   \ifx\mplib@mplibcode\mplibtemp@a
1339     \directlua{luamplib.process_mplibcode([====\the\mplibtmptoks]====])}%
1340     \end{mplibcode}%
1341   \else
1342     \mplibtmptoks\expandafter{\the\mplibtmptoks\end{#2}}%
1343     \expandafter\ltxdomplibcode
1344   \fi
1345 }
1346 \fi

    User settings.

1347 \def\mpliblegacybehavior#1{\directlua{
1348   local s = string.lower("#1")
1349   if s == "enable" or s == "true" or s == "yes" then
1350     luamplib.legacy_verbatimex = true
1351   else
1352     luamplib.legacy_verbatimex = false
1353   end
1354 }}
1355 \def\mplibverbatim#1{\directlua{
1356   local s = string.lower("#1")
1357   if s == "enable" or s == "true" or s == "yes" then
1358     luamplib.verbatiminput = true
1359   else
1360     luamplib.verbatiminput = false
1361   end
1362 }}
1363 \newtoks\mplibtmptoks

```

`\everymplib` & `\everyendmplib`: macros redefining `\everymplibtoks` & `\everyendmplibtoks` respectively

```

1364 \newtoks\everymplibtoks
1365 \newtoks\everyendmplibtoks
1366 \protected\def\everymplib{%
1367   \begingroup
1368   \mplibsetupcatcodes
1369   \mplibdoeverymplib
1370 }
1371 \long\def\mplibdoeverymplib#1{%
1372   \endgroup
1373   \everymplibtoks{#1}%
1374 }
1375 \protected\def\everyendmplib{%
1376   \begingroup
1377   \mplibsetupcatcodes
1378   \mplibdoeveryendmplib
1379 }
1380 \long\def\mplibdoeveryendmplib#1{%
1381   \endgroup
1382   \everyendmplibtoks{#1}%
1383 }

```

Allow  $\TeX$  `dimen`/`color` macros. Now `runscript` does the job, so the following lines are not needed for most cases. But the macros will be expanded when they are used in another macro.

```

1384 \def\mpdim#1{ \mplibdimen("#1") }
1385 \def\mpcolor#1#{\domplibcolor{#1}}
1386 \def\domplibcolor#1#2{ \mplibcolor("#1{#2}") }

```

MPLib's number system. Now binary has gone away.

```

1387 \def\mplibnumbersystem#1{\directlua{
1388   local t = "#1"
1389   if t == "binary" then t = "decimal" end
1390   lua\mplib.numbersystem = t
1391 }}

```

Settings for `.mp` cache files.

```

1392 \def\mplibmakenocache#1{\mplibdomakenocache #1,*,}
1393 \def\mplibdomakenocache#1,{%
1394   \ifx\empty#1\empty
1395     \expandafter\mplibdomakenocache
1396   \else
1397     \ifx*#1\else
1398       \directlua{lua\mplib.noneedtoreplace["#1.mp"]=true}%
1399     \expandafter\expandafter\expandafter\mplibdomakenocache
1400   \fi
1401 \fi
1402 }
1403 \def\mplibcancelnocache#1{\mplibdocancelnocache #1,*,}

```

```

1404 \def\mplibdocancelnocache#1,{%
1405   \ifx\empty#1\empty
1406     \expandafter\mplibdocancelnocache
1407   \else
1408     \ifx*#1\else
1409       \directlua{luamplib.noneedtoreplace["#1.mp"]=false}%
1410       \expandafter\expandafter\expandafter\mplibdocancelnocache
1411     \fi
1412   \fi
1413 }
1414 \def\mplibcachedir#1{\directlua{luamplib.getcachedir("\unexpanded{#1}")}}

```

More user settings.

```

1415 \def\mplibtexttextlabel#1{\directlua{
1416   local s = string.lower("#1")
1417   if s == "enable" or s == "true" or s == "yes" then
1418     luamplib.texttextlabel = true
1419   else
1420     luamplib.texttextlabel = false
1421   end
1422 }}
1423 \def\mplibcodeinherit#1{\directlua{
1424   local s = string.lower("#1")
1425   if s == "enable" or s == "true" or s == "yes" then
1426     luamplib.codeinherit = true
1427   else
1428     luamplib.codeinherit = false
1429   end
1430 }}
1431 \def\mplibglobaltexttext#1{\directlua{
1432   local s = string.lower("#1")
1433   if s == "enable" or s == "true" or s == "yes" then
1434     luamplib.globaltexttext = true
1435   else
1436     luamplib.globaltexttext = false
1437   end
1438 }}

```

The followings are from ConTeXt general, mostly. We use a dedicated scratchbox.

```

1439 \ifx\mplibscratchbox\undefined \newbox\mplibscratchbox \fi

```

We encapsulate the literals.

```

1440 \def\mplibstarttoPDF#1#2#3#4{%
1441   \prependtomplibbox
1442   \hbox\bgroup
1443   \xdef\MPllx{#1}\xdef\MPlly{#2}%
1444   \xdef\MPurx{#3}\xdef\MPury{#4}%
1445   \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
1446   \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
1447   \parskip0pt%
1448   \leftskip0pt%

```

```

1449 \parindent0pt%
1450 \everypar{}%
1451 \setbox\mplibscratchbox\vbox\bgroup
1452 \noindent
1453 }
1454 \def\mplibstoptoPDF{%
1455 \egroup %
1456 \setbox\mplibscratchbox\hbox %
1457 {\hskip-\MPllx bp%
1458 \raise-\MPlly bp%
1459 \box\mplibscratchbox}%
1460 \setbox\mplibscratchbox\vbox to \MPheight
1461 {\vfill
1462 \hsize\MPwidth
1463 \wd\mplibscratchbox0pt%
1464 \ht\mplibscratchbox0pt%
1465 \dp\mplibscratchbox0pt%
1466 \box\mplibscratchbox}%
1467 \wd\mplibscratchbox\MPwidth
1468 \ht\mplibscratchbox\MPheight
1469 \box\mplibscratchbox
1470 \egroup
1471 }

```

Text items have a special handler.

```

1472 \def\mplibtexttext#1#2#3#4#5{%
1473 \begingroup
1474 \setbox\mplibscratchbox\hbox
1475 {\font\temp=#1 at #2bp%
1476 \temp
1477 #3}%
1478 \setbox\mplibscratchbox\hbox
1479 {\hskip#4 bp%
1480 \raise#5 bp%
1481 \box\mplibscratchbox}%
1482 \wd\mplibscratchbox0pt%
1483 \ht\mplibscratchbox0pt%
1484 \dp\mplibscratchbox0pt%
1485 \box\mplibscratchbox
1486 \endgroup
1487 }

```

Input luamplib.cfg when it exists.

```

1488 \openin0=luamplib.cfg
1489 \ifeof0 \else
1490 \closein0
1491 \input luamplib.cfg
1492 \fi

```

That's all folks!

## 3 The GNU GPL License v2

The GPL requires the complete license text to be distributed along with the code. I recommend the canonical source, instead: <http://www.gnu.org/licenses/old-licenses/gpl-2.0.html>. But if you insist on an included copy, here it is. You might want to zoom in.

<div>GNU GENERAL PUBLIC LICENSE</div> <div>Version 2, June 1991</div> <div>Copyright © 1989, 1991 Free Software Foundation, Inc.</div> <div>51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA</div> <div>Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.</div> <div>Preamble</div> <div>The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.</div> <div>When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.</div> <div>To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.</div> <div>For example, if you distribute copies of such a program, whether grants or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.</div> <div>We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.</div> <div>Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.</div> <div>Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.</div> <div>The precise terms and conditions for copying, distribution and modification follow.</div> <div>TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION</div> <div><div><div>1. This License applies to any program or other work which contains a notice placed by the copyright holder stating it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".</div><div>Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.</div><div>2. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.</div><div>You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.</div><div>3. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:</div><div><div><div>(a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.</div><div>(b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.</div><div>(c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)</div></div></div><div>These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be</div></div><div><div>on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.</div><div>Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.</div><div>In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.</div><div>4. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:</div><div><div><div>(a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or</div><div>(b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source code distribution, a copy of the complete corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or</div><div>(c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection 1 above.)</div></div></div><div>The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.</div><div>If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.</div><div>5. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.</div><div>6. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.</div><div>7. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.</div><div>8. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.</div><div>If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.</div><div>It is not the purpose of this section to induce you to infringe any patents or other property rights claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through this system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.</div><div>This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.</div><div>9. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.</div></div><div><div>10. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.</div><div>Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.</div><div>11. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.</div></div><div><div>NO WARRANTY</div><div>12. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.</div><div>13. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REPACKAGE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.</div></div><div><div>END OF TERMS AND CONDITIONS</div><div>Appendix: How to Apply These Terms to Your New Programs</div><div>If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.</div><div>To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty, and each file should have at least the "copyright" line and a pointer to where the full notice is found.</div><div><div>one line to give the program's name and a brief idea of what it does.</div><div>Copyright (C) yyyy name of author</div><div>This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.</div><div>This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.</div><div>You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.</div><div>Also add information on how to contact you by electronic and paper mail.</div><div>If the program is interactive, make it output a short notice like this when it starts in an interactive mode:</div><div>Gnomovision version 69, Copyright (C) yyyy name of author</div><div>Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.</div><div>This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.</div><div>The hypothetical commands show w and show c should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than show w and show c; they could even be mouse-clicks or menu items—whatever suits your program.</div><div>You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:</div><div>Yorodnyne, Inc., hereby disclaims all copyright interest in the program "Gnomovision" (which makes passes at compilers) written by James Hacker.</div><div>signature of Ty Coon, 1 April 1989</div><div>Ty Coon, President of Vice</div><div>This General Public License does not permit incorporating your program into proprietary programs. If your program is a subcomponent library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.</div></div></div></div> <div data-bbox="860 1845 889 1869" data-label="Page-Footer"><p>39</p></div>
--