

The luamplib package

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

2015/01/05 v2.9.0

Abstract

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

1 Documentation

This packages 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 \LaTeX in the `mplibcode` environment.

The code is from the `luatex-mplib.lua` and `luatex-mplib.tex` files from ConTeXt, they have been adapted to \LaTeX and Plain by Elie Roux and Philipp Gesang, new functionalities have been added by Kim Dohyun. The changes are:

- a \LaTeX 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`. However, `verbatimtex ... etex` will be entirely ignored in this case.

- `verbatimtex ... etex` (in \TeX file) that comes just before `beginfig()` is not ignored, but the \TeX code inbetween 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). All other `verbatimtex ... etex`'s are ignored. *E.G.*

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

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

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

```
\everymplib{ verbatimtex \leavevmode etex; beginfig(0); }
\everyendmplib{ endfig; }
\mplibcode % beginfig/endfig not needed; always in horizontal mode
  draw fullcircle scaled 1cm;
\endmplibcode
```

N.B. Many users have complained that `mplib` figures do not respect alignment commands such as `\centering` or `\raggedleft`. That's because `luamplib` does not force horizontal or vertical mode. If you want all `mplib` figures center- (or right-) aligned, please use `\everymplib` command with `\leavevmode` as shown above.

- 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. *E.G.*

```
\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.

- With `\mpcolor` command, color names or expressions of `color/xcolor` packages can be used inside `mplibcode` environment, though `luamplib` does not automatically load these packages. See the example code above. In PDF mode, `(x)spotcolor` package is supported as well.
- Users can choose `numbersystem` option since v2.4. The default value `scaled` can be changed to `double` by declaring `\mplibnumbersystem{double}`. For details see <http://github.com/lualatex/luamplib/issues/21>.
- 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 $\text{Lua}\TeX$'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.
- 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 .
- 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.

N.B. It does not work to pass across code chunks those variables containing `btex ... etex` pictures, as these are not METAPOST, but T_EX elements from the standpoint of `luamplib`. Likewise, `graph.mp` does not work properly with the inheritance functionality.

```
\mplibcodeinherit{enable}
\everymplib{ beginfig(0);} \everyendmplib{ endfig;}
A circle
\mplibcode
  u := 10;
  draw fullcircle scaled u;
\endmplibcode
and twice the size
\mplibcode
  draw fullcircle scaled 2u;
\endmplibcode
```

- 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 `\mplibcachedir` 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

Use the `luamplib` namespace, since `mplib` is for the metapost library itself. ConT_EXt uses metapost.

```
1
2 luamplib          = luamplib or { }
3
```

Identification.

```
4
5 local luamplib    = luamplib
6 luamplib.showlog  = luamplib.showlog or false
7 luamplib.lastlog  = ""
8
9 local err, warn, info, log = luatexbase.provides_module({
10  name      = "luamplib",
11  version   = "2.9.0",
12  date      = "2015/01/05",
13  description = "Lua package to typeset Metapost with LuaTeX's MPLib.",
14 })
```

15
16

This module is a stripped down version of libraries that are used by ConT_EXt. Provide a few “shortcuts” expected by the imported code.

```
17
18 local format, abs = string.format, math.abs
19
20 local stringgsub    = string.gsub
21 local stringfind    = string.find
22 local stringmatch   = string.match
23 local stringgmatch  = string.gmatch
24 local stringexplode = string.explode
25 local tableconcat   = table.concat
26 local textsprint    = tex.sprint
27
28 local mplib = require ('mplib')
29 local kpse  = require ('kpse')
30 local lfs   = require ('lfs')
31
32 local lfsattributes = lfs.attributes
33 local lfsisdir      = lfs.isdir
34 local lfsmkdir      = lfs.mkdir
35 local lfstouch      = lfs.touch
36 local ioopen        = io.open
37
38 local file = file
39 if not file then
```

This is a small trick for L^AT_EX. In L^AT_EX we read the metapost code line by line, but it needs to be passed entirely to process(), so we simply add the lines in data and at the end we call process(data).

A few helpers, taken from l-file.lua.

```
40 file = { }
41
42 function file.replacesuffix(filename, suffix)
43     return (stringgsub(filename,"%.[%a%d]+$","") .. "." .. suffix)
44 end
45
46 function file.stripsuffix(filename)
47     return (stringgsub(filename,"%.[%a%d]+$",""))
48 end
49 end
50
```

btex ... etex in input .mp files will be replaced in finder.

```
51 local is_writable = file.is_writable or function(name)
52     if lfsisdir(name) then
53         name = name .. "/_luam_plib_temp_file_"
54         local fh = ioopen(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.mkdirs or function(path)
62     local full = ""
63     for sub in stringgmatch(path,"(/*[^\s/]+)") do
64         full = full .. sub
65         lfsmkdir(full)
66     end
67 end
68
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,stringexplode(texmfvar,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 = stringmatch(v,"%-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 = stringgsub(dir,"###","#")
104     dir = stringgsub(dir,"^~",

```

```

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
119 local noneedtoreplace = {
120     ["boxes.mp"] = true,
121     -- ["format.mp"] = true,
122     ["graph.mp"] = true,
123     ["marith.mp"] = true,
124     ["mfplain.mp"] = true,
125     ["mpost.mp"] = true,
126     ["plain.mp"] = true,
127     ["rboxes.mp"] = true,
128     ["sarith.mp"] = true,
129     ["string.mp"] = true,
130     ["TEX.mp"] = true,
131     ["metafun.mp"] = true,
132     ["metafun.mpiv"] = true,
133     ["mp-abck.mpiv"] = true,
134     ["mp-apos.mpiv"] = true,
135     ["mp-asnc.mpiv"] = true,
136     ["mp-base.mpiv"] = true,
137     ["mp-butt.mpiv"] = true,
138     ["mp-char.mpiv"] = true,
139     ["mp-chem.mpiv"] = true,
140     ["mp-core.mpiv"] = true,
141     ["mp-crop.mpiv"] = true,
142     ["mp-figs.mpiv"] = true,
143     ["mp-form.mpiv"] = true,
144     ["mp-func.mpiv"] = true,
145     ["mp-grap.mpiv"] = true,
146     ["mp-grid.mpiv"] = true,
147     ["mp-grph.mpiv"] = true,
148     ["mp-idea.mpiv"] = true,
149     ["mp-mlib.mpiv"] = true,
150     ["mp-page.mpiv"] = true,
151     ["mp-shap.mpiv"] = true,
152     ["mp-step.mpiv"] = true,
153     ["mp-text.mpiv"] = true,
154     ["mp-tool.mpiv"] = true,

```

```

155 ["mp-luas.mpiv"] = true,
156 }
157 luamplib.noneedtoreplace = noneedtoreplace
158
159 local function replaceformatmp(file,newfile,ofmodify)
160   local fh = ioopen(file,"r")
161   if not fh then return file end
162   local data = fh:read("*all"); fh:close()
163   fh = ioopen(newfile,"w")
164   if not fh then return file end
165   fh:write(
166     "let normalinfont = infont;\n",
167     "primarydef str infont name = rawtexttext(str) enddef;\n",
168     data,
169     "vardef Fmant_(expr x) = rawtexttext(decimal abs x) enddef;\n",
170     "vardef Fexp_(expr x) = rawtexttext("\$^{\"&decimal x&\")\$\"") enddef;\n",
171     "let infont = normalinfont;\n"
172   ); fh:close()
173   lfstouch(newfile,currenttime,ofmodify)
174   return newfile
175 end
176
177 local function replaceinputmpfile (name,file)
178   local ofmodify = lfsattributes(file,"modification")
179   if not ofmodify then return file end
180   local cachedir = luamplib.cachedir or outputdir
181   local newfile = stringgsub(name,"%W","_")
182   newfile = cachedir .."/luamplib_input_"..newfile
183   if newfile and luamplibtime then
184     local nf = lfsattributes(newfile)
185     if nf and nf.mode == "file" and ofmodify == nf.modification and luamplibtime < nf.access then
186       return nf.size == 0 and file or newfile
187     end
188   end
189   if name == "format.mp" then return replaceformatmp(file,newfile,ofmodify) end
190
191   local fh = ioopen(file,"r")
192   if not fh then return file end
193   local data = fh:read("*all"); fh:close()
194   data = stringgsub(data, "\n[\^\\n]-\\n",
195     function(str)
196       str = stringgsub(str,"([bem])tex%f[^A-Z_a-z]","%1!!!T!!!E!!!X!!!")
197       return str
198     end)
199   local count,cnt = 0,0
200   data,cnt = stringgsub(data,
201     "%f[A-Z_a-z]btex%f[^A-Z_a-z]%s*(.)%s*f[A-Z_a-z]etex%f[^A-Z_a-z]",
202     function(str)
203       str = stringgsub(str,"\\%%", "\\!\\!\\!PERCENT!\\!\\!")

```



```

204     str = stringgsub(str, "%%-\\n", "")
205     str = stringgsub(str, "%%-$", "")
206     str = stringgsub(str, "\\!!!!PERCENT!!!!", "\\%")
207     str = stringgsub(str, "[\\n\\r]%s*", " ")
208     str = stringgsub(str, "'", "&ditto&'")
209     return format("rawtexttext(\"%s\\\"", str)
210 end)
211 count = count + cnt
212 data, cnt = stringgsub(data,
213     "%f[A-Z_a-z]verbatimtex%f[^A-Z_a-z]%s*.-%s*f[A-Z_a-z]etex%f[^A-Z_a-z]",
214     "")
215 count = count + cnt
216 if count == 0 then
217     needtoreplace[name] = true
218     fh = ioopen(newfile, "w");
219     if fh then
220         fh:close()
221         lfstouch(newfile, currenttime, ofmodify)
222     end
223     return file
224 end
225 data = stringgsub(data, "([bem])!!!T!!!E!!!X!!!", "%1tex")
226 fh = ioopen(newfile, "w")
227 if not fh then return file end
228 fh:write(data); fh:close()
229 lfstouch(newfile, currenttime, ofmodify)
230 return newfile
231 end
232
233 local randomseed = nil

```

As the finder function for mpplib, use the kpse library and make it behave like as if MetaPost was used (or almost, since the engine name is not set this way—not sure if this is a problem).

```

234
235 local mpkpse = kpse.new("luatex", "mpost")
236
237 local function finder(name, mode, ftype)
238     if mode == "w" then
239         return name
240     else
241         local file = mpkpse:find_file(name, ftype)
242         if file then
243             if not lfstouch or ftype ~= "mp" or needtoreplace[name] then
244                 return file
245             end
246             return replaceinputmpfile(name, file)
247         end
248         return mpkpse:find_file(name, stringmatch(name, "[a-zA-Z]+$"))
249     end

```

```

250 end
251 luamplib.finder = finder
252

```

The rest of this module is not documented. More info can be found in the LuaTeX manual, articles in user group journals and the files that ship with ConTeXt.

```

253
254 function luamplib.resetlastlog()
255   luamplib.lastlog = ""
256 end
257

```

Below included is section that defines fallbacks for older versions of mplib.

```

258 local mplibone = tonumber(mplib.version()) <= 1.50
259
260 if mplibone then
261
262   luamplib.make = luamplib.make or function(name, mem_name, dump)
263     local t = os.clock()
264     local mpx = mplib.new {
265       ini_version = true,
266       find_file = luamplib.finder,
267       job_name = file.stripsuffix(name)
268     }
269     mpx:execute(format("input %s ;", name))
270     if dump then
271       mpx:execute("dump ;")
272       info("format %s made and dumped for %s in %0.3f seconds", mem_name, name, os.clock()-t)
273     else
274       info("%s read in %0.3f seconds", name, os.clock()-t)
275     end
276     return mpx
277   end
278
279   function luamplib.load(name)
280     local mem_name = file.replacesuffix(name, "mem")
281     local mpx = mplib.new {
282       ini_version = false,
283       mem_name = mem_name,
284       find_file = luamplib.finder
285     }
286     if not mpx and type(luamplib.make) == "function" then
287       -- when i have time i'll locate the format and dump
288       mpx = luamplib.make(name, mem_name)
289     end
290     if mpx then
291       info("using format %s", mem_name, false)
292       return mpx, nil
293     else
294       return nil, { status = 99, error = "out of memory or invalid format" }
295     end
296   end
297

```

```

295     end
296 end
297
298 else
299

```

These are the versions called with sufficiently recent mplib.

```

300 local preamble = [[
301     boolean mplib ; mplib := true ;
302     let dump = endinput ;
303     let normalfontsize = fontsize;
304     input %s ;
305 ]]
306
307 luamplib.make = luamplib.make or function()
308 end
309
310 function luamplib.load(name)
311     local mpx = mplib.new {
312         ini_version = true,
313         find_file = luamplib.finder,

```

Provides numbersystem option since v2.4. Default value "scaled" can be changed by declaring \mplibnumbersystem{double}. See <https://github.com/lualatex/luamplib/issues/21>.

```

314         math_mode = luamplib.numbersystem,
315         random_seed = randomseed,
316     }

```

Append our own preamble to the preamble above.

```

317     local preamble = preamble .. luamplib.mplibcodepreamble
318     if luamplib.texttextlabel then
319         preamble = preamble .. luamplib.texttextlabelpreamble
320     end
321     local result
322     if not mpx then
323         result = { status = 99, error = "out of memory"}
324     else
325         result = mpx:execute(format(preamble, file.replacesuffix(name,"mp")))
326     end
327     luamplib.reporterror(result)
328     return mpx, result
329 end
330
331 end
332
333 local currentformat = "plain"
334
335 local function setformat (name) --- used in .sty
336     currentformat = name

```

```

337 end
338 luamplib.setformat = setformat
339
340
341 luamplib.reporterror = function (result)
342   if not result then
343     err("no result object returned")
344   else
345     local t, e, l = result.term, result.error, result.log
346     local log = stringgsub(t or l or "no-term", "%s+", "\n")
347     luamplib.lastlog = luamplib.lastlog .. "\n " .. (l or t or "no-log")
348     if result.status > 0 then
349       warn("%s", log)
350       if result.status > 1 then
351         err("%s", e or "see above messages")
352       end
353     end
354     return log
355   end
356 end
357
358 local function process_indeed (mpx, data, indeed)
359   local converted, result = false, {}
360   if mpx and data then
361     result = mpx:execute(data)
362     local log = luamplib.reporterror(result)
363     if indeed and log then
364       if luamplib.showlog then
365         info("%s", luamplib.lastlog)
366         luamplib.resetlastlog()
367       elseif result.fig then

```

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.

```

368         if stringfind(log, "\n>>") then info("%s", log) end
369         converted = luamplib.convert(result)
370       else
371         info("%s", log)
372         warn("No figure output. Maybe no beginfig/endfig")
373       end
374     end
375   else
376     err("Mem file unloadable. Maybe generated with a different version of mplib?")
377   end
378   return converted, result
379 end
380

```

v2.9 has introduced the concept of ‘code inherit’

```

381 luamplib.codeinherit = false
382 local mplibinstances = {}
383 local process = function (data, indeed)
384   local standalone, firstpass = not luamplib.codeinherit, not indeed
385   local currfmt = currentformat .. luamplib.numbersystem
386   currfmt = firstpass and currfmt or (currfmt.."2")
387   local mpx = mplibinstances[currfmt]
388   if standalone or not mpx then
389     randomseed = firstpass and math.random(65535) or randomseed
390     mpx = luamplib.load(currentformat)
391     mplibinstances[currfmt] = mpx
392   end
393   return process_indeed(mpx, data, indeed)
394 end
395 luamplib.process = process
396
397 local function getobjects(result, figure, f)
398   return figure:objects()
399 end
400
401 local function convert(result, flusher)
402   luamplib.flush(result, flusher)
403   return true -- done
404 end
405 luamplib.convert = convert
406
407 local function pdf_startfigure(n, llx, lly, urx, ury)

```

The following line has been slightly modified by Kim.

```

408   texsprint(format("\\mplibstarttoPDF{%f}{%f}{%f}{%f}", llx, lly, urx, ury))
409 end
410
411 local function pdf_stopfigure()
412   texsprint("\\mplibstoptoPDF")
413 end
414
415 local function pdf_literalcode(fmt, ...) -- table
416   texsprint(format("\\mplibtoPDF{%s}", format(fmt, ...)))
417 end
418 luamplib.pdf_literalcode = pdf_literalcode
419
420 local function pdf_textfigure(font, size, text, width, height, depth)

```

The following three lines have been modified by Kim.

```

421   -- if text == "" then text = "\0" end -- char(0) has gone
422   text = text:gsub(".", function(c)
423     return format("\\hbox{\\char%i}", string.byte(c)) -- kerning happens in meta-
      post
424   end)
425   texsprint(format("\\mplibtexttext{%s}{%f}{%s}{%s}{%f}", font, size, text, 0, -( 7200/ 7227)/65536*depth))
426 end

```

```

427 luamplib.pdf_textfigure = pdf_textfigure
428
429 local bend_tolerance = 131/65536
430
431 local rx, sx, sy, ry, tx, ty, divider = 1, 0, 0, 1, 0, 0, 1
432
433 local function pen_characteristics(object)
434   local t = mplib.pen_info(object)
435   rx, ry, sx, sy, tx, ty = t.rx, t.ry, t.sx, t.sy, t.tx, t.ty
436   divider = sx*sy - rx*ry
437   return not (sx==1 and rx==0 and ry==0 and sy==1 and tx==0 and ty==0), t.width
438 end
439
440 local function concat(px, py) -- no tx, ty here
441   return (sy*px-ry*py)/divider, (sx*py-rx*px)/divider
442 end
443
444 local function curved(ith,pth)
445   local d = pth.left_x - ith.right_x
446   if abs(ith.right_x - ith.x_coord - d) <= bend_tolerance and abs(pth.x_coord - pth.left_x - d) <= bend_tolerance then
447     d = pth.left_y - ith.right_y
448     if abs(ith.right_y - ith.y_coord - d) <= bend_tolerance and abs(pth.y_coord - pth.left_y - d) <= bend_tolerance then
449       return false
450     end
451   end
452   return true
453 end
454
455 local function flushnormalpath(path,open)
456   local pth, ith
457   for i=1,#path do
458     pth = path[i]
459     if not ith then
460       pdf_literalcode("%f %f m",pth.x_coord,pth.y_coord)
461     elseif curved(ith,pth) then
462       pdf_literalcode("%f %f %f %f %f %f c",ith.right_x,ith.right_y,pth.left_x,pth.left_y,pth.x_coord,pth.y_coord)
463     else
464       pdf_literalcode("%f %f l",pth.x_coord,pth.y_coord)
465     end
466     ith = pth
467   end
468   if not open then
469     local one = path[1]
470     if curved(pth,one) then
471       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)
472     else
473       pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
474     end

```

```

475 elseif #path == 1 then
476   -- special case .. draw point
477   local one = path[1]
478   pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
479 end
480 return t
481 end
482
483 local function flushconcatpath(path,open)
484 pdf_literalcode("%f %f %f %f %f %f cm", sx, rx, ry, sy, tx ,ty)
485 local pth, ith
486 for i=1,#path do
487   pth = path[i]
488   if not ith then
489     pdf_literalcode("%f %f m",concat(pth.x_coord,pth.y_coord))
490   elseif curved(ith,pth) then
491     local a, b = concat(ith.right_x,ith.right_y)
492     local c, d = concat(pth.left_x,pth.left_y)
493     pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(pth.x_coord, pth.y_co-
ord))
494   else
495     pdf_literalcode("%f %f l",concat(pth.x_coord, pth.y_coord))
496   end
497   ith = pth
498 end
499 if not open then
500   local one = path[1]
501   if curved(pth,one) then
502     local a, b = concat(pth.right_x,pth.right_y)
503     local c, d = concat(one.left_x,one.left_y)
504     pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(one.x_coord, one.y_co-
ord))
505   else
506     pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
507   end
508 elseif #path == 1 then
509   -- special case .. draw point
510   local one = path[1]
511   pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
512 end
513 return t
514 end
515

```

Below code has been contributed by Dohyun Kim. It implements btex / etex functions.

v2.1: texttext() is now available, which is equivalent to TEX() macro from TEX.mp.

TEX() is synonym of texttext() unless TEX.mp is loaded.

v2.2: Transparency and Shading

v2.3: \everymplib, \everyendmplib, and allows naked T_EX commands.

```

516 local further_split_keys = {

```

```

517 ["MPlibTEXboxID"] = true,
518 ["sh_color_a"]    = true,
519 ["sh_color_b"]    = true,
520 }
521
522 local function script2table(s)
523   local t = {}
524   for _,i in ipairs(stringexplode(s,"\13+")) do
525     local k,v = stringmatch(i,"(.-)=(.*)") -- v may contain = or empty.
526     if k and v and k ~= "" then
527       if further_split_keys[k] then
528         t[k] = stringexplode(v,":")
529       else
530         t[k] = v
531       end
532     end
533   end
534   return t
535 end
536
537 local mplibcodepreamble = [[
538 vardef rawtexttext (expr t) =
539   if unknown TEXBOX_:
540     image( special "MPlibmkTEXbox="&t;
541       addto currentpicture doublepath unitsquare; )
542   else:
543     TEXBOX_ := TEXBOX_ + 1;
544     if known TEXBOX_wd_[TEXBOX_]:
545       image ( addto currentpicture doublepath unitsquare
546         xscaled TEXBOX_wd_[TEXBOX_]
547         yscaled (TEXBOX_ht_[TEXBOX_] + TEXBOX_dp_[TEXBOX_])
548         shifted (0, -TEXBOX_dp_[TEXBOX_])
549         withprescript "MPlibTEXboxID=" &
550           decimal TEXBOX_ & ":" &
551           decimal TEXBOX_wd_[TEXBOX_] & ":" &
552           decimal(TEXBOX_ht_[TEXBOX_]+TEXBOX_dp_[TEXBOX_]); )
553     else:
554       image( special "MPlibTEXError=1"; )
555   fi
556 fi
557 enddef;
558 if known context_mlib:
559   defaultfont := "cmtt10";
560   let infont = normalinfont;
561   let fontsize = normalfontsize;
562   vardef thelabel@#(expr p,z) =
563     if string p :
564       thelabel@#(p infont defaultfont scaled defaultscale,z)
565     else :
566       p shifted (z + labeloffset*mfun_laboff@# -

```



```

567      (mfun_labxf@#*lrcorner p + mfun_labyf@#*ulcorner p +
568      (1-mfun_labxf@#-mfun_labyf@#)*llcorner p))
569    fi
570  enddef;
571  def graphicstext primary filename =
572    if (readfrom filename = EOF):
573      errmessage "Please prepare '"&filename&'" in advance with"&
574      " 'pstoedit -ssp -dt -f mpost yourfile.ps "&filename&""";
575    fi
576    closefrom filename;
577    def data_mpy_file = filename enddef;
578    mfun_do_graphic_text (filename)
579  enddef;
580  if unknown TEXBOX_: def mfun_do_graphic_text text t = enddef; fi
581 else:
582   vardef texttext@# (text t) = rawtexttext (t) enddef;
583 fi
584 def externalfigure primary filename =
585   draw rawtexttext("\includegraphics{"& filename &}")
586 enddef;
587 def TEX = texttext enddef;
588 def fontmapfile primary filename = enddef;
589 def specialVerbatimTeX (text t) = special "MPLibVerbTeX="&t; enddef;
590 def ignoreVerbatimTeX (text t) = enddef;
591 let VerbatimTeX = specialVerbatimTeX;
592 extra_beginfig := extra_beginfig & " let VerbatimTeX = ignoreVerbatimTeX;" ;
593 extra_endfig := extra_endfig & " let VerbatimTeX = specialVerbatimTeX;" ;
594 ]]
595 luamplib.mplibcodepreamble = mplibcodepreamble
596
597 local texttextlabelpreamble = [[
598 primarydef s infont f = rawtexttext(s) enddef;
599 def fontsize expr f =
600   begingroup
601     save size,pic; numeric size; picture pic;
602     pic := rawtexttext("\hskip\pdffontsize\font");
603     size := xpart urcorner pic - xpart llcorner pic;
604     if size = 0: 10pt else: size fi
605   endgroup
606 enddef;
607 ]]
608 luamplib.texttextlabelpreamble = texttextlabelpreamble
609
610 local function protecttexttext(data)
611   local everymplib = tex.toks['everymplibtoks'] or ''
612   local everyendmplib = tex.toks['everyendmplibtoks'] or ''
613   data = "\n" .. everymplib .. "\n" .. data .. "\n" .. everyendmplib
614   data = stringgsub(data, "\r", "\n")
615   data = stringgsub(data, "\n[\n]-\n",
616   function(str)

```

```

617     str = stringgsub(str,"%%","!!!!!!PERCENT!!!!!!")
618     str = stringgsub(str,"([bem])tex%f[^A-Z_a-z]","%1!!!!T!!!!E!!!!X!!!!")
619     return str
620 end)
621 data = stringgsub(data,
622   "%f[A-Z_a-z]btex%f[^A-Z_a-z]%s*(.)%s%f[A-Z_a-z]etex%f[^A-Z_a-z]",
623   function(str)
624     str = stringgsub(str,"\\%", "\\!!!!!!PERCENT!!!!!!")
625     str = stringgsub(str,"%%.-\\n", "")
626     str = stringgsub(str,"%%.-$", "")
627     str = stringgsub(str,"'", "'&ditto&'")
628     str = stringgsub(str,"\\n%s*", " ")
629     return format("rawtexttext(\\\"%s\\\")", str)
630 end)
631 data = stringgsub(data,
632   "%f[A-Z_a-z]verbatimtex%f[^A-Z_a-z]%s*(.)%s%f[A-Z_a-z]etex%f[^A-Z_a-z]",
633   function(str)
634     str = stringgsub(str,"\\%", "\\!!!!!!PERCENT!!!!!!")
635     str = stringgsub(str,"%%.-\\n", "")
636     str = stringgsub(str,"%%.-$", "")
637     str = stringgsub(str,"'", "'&ditto&'")
638     str = stringgsub(str,"\\n%s*", " ")
639     return format("VerbatimTeX(\\\"%s\\\")", str)
640 end)
641 data = stringgsub(data, "\\^[^\\n]-\\",
642   function(str)
643     str = stringgsub(str,"([bem])!!!!T!!!!E!!!!X!!!!","%1tex")
644     str = stringgsub(str,"{", "!!!!!!LEFTBRCE!!!!!!")
645     str = stringgsub(str,"}", "!!!!!!RGHTBRCE!!!!!!")
646     str = stringgsub(str,"#", "!!!!!!SHARPE!!!!!!")
647     return format("\\detokenize{%s}", str)
648 end)
649 data = stringgsub(data,"%%.-\\n", "")
650 luamplib.mpxcolors = {}
651 data = stringgsub(data, "\\mpcolor%s*{(.)}",
652   function(str)
653     local cnt = #luamplib.mpxcolors + 1
654     luamplib.mpxcolors[cnt] = format(
655       "\\expandafter\\mplibcolor\\csname mpxcolor%i\\endcsname{%s}", cnt, str)
656     return format("\\csname mpxcolor%i\\endcsname", cnt)
657   end)
658 texpstr(data)
659 end
660
661 luamplib.protecttexttext = protecttexttext
662
663 local TeX_code_t = {}
664
665 local function domakeTEXboxes (data)
666   local num = 255 -- output box

```

```

667 if data and data.fig then
668     local figures = data.fig
669     for f=1, #figures do
670         TeX_code_t[f] = nil
671         local figure = figures[f]
672         local objects = getobjects(data, figure, f)
673         if objects then
674             for o=1, #objects do
675                 local object = objects[o]
676                 local prescript = object.prescript
677                 prescript = prescript and script2table(prescript)
678                 local str = prescript and prescript.MPLibmkTEXbox
679                 if str then
680                     num = num + 1
681                     texsprintf(format("\\setbox%i\\hbox{%s}", num, str))
682                 end

```

verbatimtex ... etex before beginfig() is not ignored, but the TeX code inbetween is inserted before the mplib box.

```

683         local texcode = prescript and prescript.MPLibVerbTeX
684         if texcode and texcode ~= "" then
685             TeX_code_t[f] = texcode
686         end
687     end
688 end
689 end
690 end
691 end
692
693 local function makeTEXboxes (data)
694     data = stringgsub(data, "##", "#") -- restore # doubled in input string
695     data = stringgsub(data, "!!!!PERCENT!!!!", "%")
696     data = stringgsub(data, "!!!!LEFTBRCE!!!!", "{")
697     data = stringgsub(data, "!!!!RIGHTBRCE!!!!", "}")
698     data = stringgsub(data, "!!!!SHARPE!!!!", "#")
699     local _, result = process(data, false)
700     domakeTEXboxes(result)
701     return data
702 end
703
704 luamplib.makeTEXboxes = makeTEXboxes
705
706 local factor = 65536*(7227/7200)
707
708 local function processwithTEXboxes (data)
709     if not data then return end
710     local num = 255 -- output box
711     local preamble = format("TEXBOX_:=%i;\n", num)
712     while true do
713         num = num + 1

```

```

714     local box = tex.box[num]
715     if not box then break end
716     prepreamble = format(
717         "%sTEXBOX_wd_[%i]:=f;\nTEXBOX_ht_[%i]:=f;\nTEXBOX_dp_[%i]:=f;\n",
718         prepreamble,
719         num, box.width /factor,
720         num, box.height/factor,
721         num, box.depth /factor)
722     end
723     process(prepreamble .. data, true)
724 end
725 luamplib.processwithTEXboxes = processwithTEXboxes
726
727 local pdfmode = tex.pdfoutput > 0 and true or false
728
729 local function start_pdf_code()
730     if pdfmode then
731         pdf_literalcode("q")
732     else
733         texpstr("\special{pdf:bcontent}") -- dvipdfmx
734     end
735 end
736 local function stop_pdf_code()
737     if pdfmode then
738         pdf_literalcode("Q")
739     else
740         texpstr("\special{pdf:econtent}") -- dvipdfmx
741     end
742 end
743
744 local function putTEXboxes (object,prescript)
745     local box = prescript.MPlibTEXboxID
746     local n,tw,th = box[1],box[2],box[3]
747     if n and tw and th then
748         local op = object.path
749         local first, second, fourth = op[1], op[2], op[4]
750         local tx, ty = first.x_coord, first.y_coord
751         local sx, sy = (second.x_coord - tx)/tw, (fourth.y_coord - ty)/th
752         local rx, ry = (second.y_coord - ty)/tw, (fourth.x_coord - tx)/th
753         if sx == 0 then sx = 0.00001 end
754         if sy == 0 then sy = 0.00001 end
755         start_pdf_code()
756         pdf_literalcode("%f %f %f %f %f %f cm",sx,rx,ry,sy,tx,ty)
757         texpstr(format("\mplibputtextbox{%i}",n))
758         stop_pdf_code()
759     end
760 end
761

```

Transparency and Shading

```

762 local pdf_objs = {}
763
764 if not pdfmode then
765   texsprint("\\special{pdf:obj @MPlibTr<<>>}",
766     "\\special{pdf:obj @MPlibSh<<>>}")
767 end
768
769 -- objstr <string> => obj <number>, new <boolean>
770 local function update_pdfobjs (os)
771   local on = pdf_objs[os]
772   if on then
773     return on,false
774   end
775   if pdfmode then
776     on = pdf.immediateobj(os)
777   else
778     on = pdf_objs.cnt or 0
779     pdf_objs.cnt = on + 1
780   end
781   pdf_objs[os] = on
782   return on,true
783 end
784
785 local transparency_modes = { [0] = "Normal",
786   "Normal",      "Multiply",    "Screen",      "Overlay",
787   "SoftLight",   "HardLight",    "ColorDodge",  "ColorBurn",
788   "Darken",      "Lighten",      "Difference",   "Exclusion",
789   "Hue",         "Saturation",    "Color",       "Luminosity",
790   "Compatible",
791 }
792
793 local function update_tr_res(res,mode,opaque)
794   local os = format("<</BM /%s/ca %.3f/CA %.3f/AIS false>>",mode,opaque,opaque)
795   local on, new = update_pdfobjs(os)
796   if new then
797     if pdfmode then
798       res = format("%s/MPlibTr%i %i 0 R",res,on,on)
799     else
800       texsprint(format("\\special{pdf:put @MPlibTr<</MPlibTr%i%s>>}",on,os))
801     end
802   end
803   return res,on
804 end
805
806 local function tr_pdf_pageresources(mode,opaque)
807   local res, on_on, off_on = "", nil, nil
808   res, off_on = update_tr_res(res, "Normal", 1)
809   res, on_on = update_tr_res(res, mode, opaque)
810   if pdfmode then
811     if res ~= "" then

```

```

812     local tpr = tex.pdfpageresources -- respect luaotfload-colors
813     if not stringfind(tpr,"/ExtGState<<.*>>") then
814         tpr = tpr.."/ExtGState<<>>"
815     end
816     tpr = stringgsub(tpr,"/ExtGState<<","%1"..res)
817     tex.set("global","pdfpageresources",tpr)
818 end
819 else
820     texsprintf(format("\\special{pdf:put @resources<</ExtGState @MPLibTr>>}"))
821 end
822 return on_on, off_on
823 end
824
825 local shading_res
826 local getpageres = pdf.getpageresources or function() return pdf.pageresources end
827 local setpageres = pdf.setpageresources or function(s) pdf.pageresources = s end
828
829 local function shading_initialize ()
830     shading_res = {}
831     if pdfmode then
832         require('luatexbase.mcb')
833         if luatexbase.is_active_callback then -- luatexbase 0.7+
834             local shading_obj = pdf.reserveobj()
835             setpageres(format("%s/Shading %i 0 R",getpageres() or "",shading_obj))
836             luatexbase.add_to_callback("finish_pdffile", function()
837                 pdf.immediateobj(shading_obj,format("<<%s>>",tableconcat(shading_res)))
838             end, "luamplib.finish_pdffile")
839             pdf_objs.finishpdf = true
840         end
841     end
842 end
843
844 local function sh_pdfpageresources(shtype, domain, colorspace, colora, colorb, coordinates)
845     if not shading_res then shading_initialize() end
846     local os = format("<</FunctionType 2/Domain [ %s ]/C0 [ %s ]/C1 [ %s ]/N 1>>",
847         domain, colora, colorb)
848     local funcobj = pdfmode and format("%i 0 R",update_pdfobjs(os)) or os
849     os = format("<</ShadingType %i/ColorSpace /%s/Function %s/Coords [ %s ]/Extend [ true true ]/AntiAlias true>>",
850         shtype, colorspace, funcobj, coordinates)
851     local on, new = update_pdfobjs(os)
852     if pdfmode then
853         if new then
854             local res = format("/MPLibSh%i %i 0 R", on, on)
855             if pdf_objs.finishpdf then
856                 shading_res[#shading_res+1] = res
857             else
858                 local pageres = getpageres() or ""
859                 if not stringfind(pageres,"/Shading<<.*>>") then
860                     pageres = pageres.."/Shading<<>>"

```

```

861     end
862     pageres = stringgsub(pageres,"/Shading<<","%1"..res)
863     setpageres(pageres)
864     end
865     end
866 else
867     if new then
868         texsprint(format("\\special{pdf:put @MPlibSh<</MPlibSh%i%s>>}",on,os))
869     end
870     texsprint(format("\\special{pdf:put @resources<</Shading @MPlibSh>>}"))
871 end
872 return on
873 end
874
875 local function color_normalize(ca,cb)
876     if #cb == 1 then
877         if #ca == 4 then
878             cb[1], cb[2], cb[3], cb[4] = 0, 0, 0, 1-cb[1]
879         else -- #ca = 3
880             cb[1], cb[2], cb[3] = cb[1], cb[1], cb[1]
881         end
882     elseif #cb == 3 then -- #ca == 4
883         cb[1], cb[2], cb[3], cb[4] = 1-cb[1], 1-cb[2], 1-cb[3], 0
884     end
885 end
886
887 local prev_override_color
888
889 local function do_preobj_color(object,prescript)
890     -- transparency
891     local opaq = prescript and prescript.tr_transparency
892     local tron_no, troff_no
893     if opaq then
894         local mode = prescript.tr_alternative or 1
895         mode = transparency_modes[tonumber(mode)]
896         tron_no, troff_no = tr_pdf_pageresources(mode,opaq)
897         pdf_literalcode("/MPlibTr%i gs",tron_no)
898     end
899     -- color
900     local override = prescript and prescript.MPlibOverrideColor
901     if override then
902         if pdfmode then
903             pdf_literalcode(override)
904             override = nil
905         else
906             texsprint(format("\\special{color push %s}",override))
907             prev_override_color = override
908         end
909     else
910         local cs = object.color

```

```

911     if cs and #cs > 0 then
912         pdf_literalcode(luamplib.colorconverter(cs))
913         prev_override_color = nil
914     elseif not pdfmode then
915         override = prev_override_color
916         if override then
917             texsprint(format("\\special{color push %s}", override))
918         end
919     end
920 end
921 -- shading
922 local sh_type = prescript and prescript.sh_type
923 if sh_type then
924     local domain = prescript.sh_domain
925     local centera = stringexplode(prescript.sh_center_a)
926     local centerb = stringexplode(prescript.sh_center_b)
927     for _,t in pairs({centera,centerb}) do
928         for i,v in ipairs(t) do
929             t[i] = format("%f",v)
930         end
931     end
932     centera = tableconcat(centera, " ")
933     centerb = tableconcat(centerb, " ")
934     local colora = prescript.sh_color_a or {0};
935     local colorb = prescript.sh_color_b or {1};
936     for _,t in pairs({colora,colorb}) do
937         for i,v in ipairs(t) do
938             t[i] = format("%.3f",v)
939         end
940     end
941     if #colora > #colorb then
942         color_normalize(colora,colorb)
943     elseif #colorb > #colora then
944         color_normalize(colorb,colora)
945     end
946     local colorspace
947     if #colorb == 1 then colorspace = "DeviceGray"
948     elseif #colorb == 3 then colorspace = "DeviceRGB"
949     elseif #colorb == 4 then colorspace = "DeviceCMYK"
950     else return troff_no,override
951     end
952     colora = tableconcat(colora, " ")
953     colorb = tableconcat(colorb, " ")
954     local shade_no
955     if sh_type == "linear" then
956         local coordinates = tableconcat({centera,centerb}, " ")
957         shade_no = sh_pdfpageresources(2,domain,colorspace,colora,colorb,coordinates)
958     elseif sh_type == "circular" then
959         local radiusa = format("%f",prescript.sh_radius_a)
960         local radiusb = format("%f",prescript.sh_radius_b)

```



```

961     local coordinates = tableconcat({centera,radiusa,centerb,radiusb}," ")
962     shade_no = sh_pdfpageresources(3,domain,colorspace,colora,colorb,coordinates)
963     end
964     pdf_literalcode("q /Pattern cs")
965     return troff_no,override,shade_no
966 end
967 return troff_no,override
968 end
969
970 local function do_postobj_color(tr,over,sh)
971   if sh then
972     pdf_literalcode("W n /MPLibSh%s sh Q",sh)
973   end
974   if over then
975     texsprint("\special{color pop}")
976   end
977   if tr then
978     pdf_literalcode("/MPLibTr%i gs",tr)
979   end
980 end
981

```

End of btex – etex and Transparency/Shading patch.

```

982
983 local function flush(result,flusher)
984   if result then
985     local figures = result.fig
986     if figures then
987       for f=1, #figures do
988         info("flushing figure %s",f)
989         local figure = figures[f]
990         local objects = getobjects(result,figure,f)
991         local fignum = tonumber(stringmatch(figure:filename(),"([%d]+)$") or figure:charcode() or 0)
992         local miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
993         local bbox = figure:boundingbox()
994         local llx, lly, urx, ury = bbox[1], bbox[2], bbox[3], bbox[4] -- faster than unpack
995         if urx < llx then
996           -- invalid
997           pdf_startfigure(fignum,0,0,0,0)
998           pdf_stopfigure()
999         else

```

Insert verbatimtex code before mplib box.

```

1000     if TeX_code_t[f] then
1001       texsprint(TeX_code_t[f])
1002     end
1003     pdf_startfigure(fignum,llx,lly,urx,ury)
1004     start_pdf_code()

```

```

1005         if objects then
1006             for o=1,#objects do
1007                 local object      = objects[o]
1008                 local objecttype  = object.type

```

Change from ConT_EXt code: the following 5 lines are part of the btex...etex patch.
Again, colors are processed at this stage.

```

1009             local prescript      = object.prescript
1010             prescript = prescript and script2table(prescript) -- prescript is now a ta-
ble
1011             local tr_opaq,cr_over,shade_no = do_preobj_color(object,prescript)
1012             if prescript and prescript.MPlibTEXboxID then
1013                 putTEXboxes(object,prescript)
1014             elseif objecttype == "start_bounds" or objecttype == "stop_bounds" then
1015                 -- skip
1016             elseif objecttype == "start_clip" then
1017                 start_pdf_code()
1018                 flushnormalpath(object.path,t,false)
1019                 pdf_literalcode("W n")
1020             elseif objecttype == "stop_clip" then
1021                 stop_pdf_code()
1022                 miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
1023             elseif objecttype == "special" then
1024                 -- not supported
1025                 if prescript and prescript.MPlibTEXError then
1026                     warn("texttext() anomaly. Try disabling \\mplibtexttextlabel.")
1027                 end
1028             elseif objecttype == "text" then
1029                 local ot = object.transform -- 3,4,5,6,1,2
1030                 start_pdf_code()
1031                 pdf_literalcode("%f %f %f %f %f %f cm",ot[3],ot[4],ot[5],ot[6],ot[1],ot[2])
1032                 pdf_textfigure(object.font,object.dsize,object.text,object.width,object.height,object.d)
1033                 stop_pdf_code()
1034             else

```

Color stuffs are modified and moved to several lines above.

```

1035             local ml = object.miterlimit
1036             if ml and ml ~= miterlimit then
1037                 miterlimit = ml
1038                 pdf_literalcode("%f M",ml)
1039             end
1040             local lj = object.linejoin
1041             if lj and lj ~= linejoin then
1042                 linejoin = lj
1043                 pdf_literalcode("%i j",lj)
1044             end
1045             local lc = object.linecap
1046             if lc and lc ~= linecap then
1047                 linecap = lc
1048                 pdf_literalcode("%i J",lc)

```

```

1049         end
1050         local dl = object.dash
1051         if dl then
1052             local d = format("[%s] %i d",tableconcat(dl.dashes or {}, " "),dl.offset)
1053             if d ~= dashed then
1054                 dashed = d
1055                 pdf_literalcode(dashed)
1056             end
1057         elseif dashed then
1058             pdf_literalcode("[ ] 0 d")
1059             dashed = false
1060         end
1061         local path = object.path
1062         local transformed, penwidth = false, 1
1063         local open = path and path[1].left_type and path[#path].right_type
1064         local pen = object.pen
1065         if pen then
1066             if pen.type == 'elliptical' then
1067                 transformed, penwidth = pen_characteristics(object) -- boolean, value
1068                 pdf_literalcode("%f w",penwidth)
1069                 if objecttype == 'fill' then
1070                     objecttype = 'both'
1071                 end
1072             else -- calculated by mplib itself
1073                 objecttype = 'fill'
1074             end
1075         end
1076         if transformed then
1077             start_pdf_code()
1078         end
1079         if path then
1080             if transformed then
1081                 flushconcatpath(path,open)
1082             else
1083                 flushnormalpath(path,open)
1084             end
1085         end

```

Change from ConT_EXt code: color stuff

```

1085         if not shade_no then ----- conflict with shading
1086             if objecttype == "fill" then
1087                 pdf_literalcode("h f")
1088             elseif objecttype == "outline" then
1089                 pdf_literalcode((open and "S") or "h S")
1090             elseif objecttype == "both" then
1091                 pdf_literalcode("h B")
1092             end
1093         end
1094     end
1095     if transformed then
1096         stop_pdf_code()

```

```

1097         end
1098         local path = object.htap
1099         if path then
1100             if transformed then
1101                 start_pdf_code()
1102             end
1103             if transformed then
1104                 flushconcatpath(path,open)
1105             else
1106                 flushnormalpath(path,open)
1107             end
1108             if objecttype == "fill" then
1109                 pdf_literalcode("h f")
1110             elseif objecttype == "outline" then
1111                 pdf_literalcode((open and "S") or "h S")
1112             elseif objecttype == "both" then
1113                 pdf_literalcode("h B")
1114             end
1115             if transformed then
1116                 stop_pdf_code()
1117             end
1118         end
1119         -- if cr then
1120         --     pdf_literalcode(cr)
1121         -- end
1122     end

```

Added to ConTeXt code: color stuff

```

1123         do_postobj_color(tr_opaq,cr_over,shade_no)
1124     end
1125 end
1126 stop_pdf_code()
1127 pdf_stopfigure()
1128 end
1129 end
1130 end
1131 end
1132 end
1133 luamplib.flush = flush
1134
1135 local function colorconverter(cr)
1136     local n = #cr
1137     if n == 4 then
1138         local c, m, y, k = cr[1], cr[2], cr[3], cr[4]
1139         return format("%.3f %.3f %.3f %.3f k %.3f %.3f %.3f %.3f K",c,m,y,k,c,m,y,k), "0 g 0 G"
1140     elseif n == 3 then
1141         local r, g, b = cr[1], cr[2], cr[3]
1142         return format("%.3f %.3f %.3f rg %.3f %.3f %.3f RG",r,g,b,r,g,b), "0 g 0 G"
1143     else
1144         local s = cr[1]

```

```

1145     return format("%.3f g %.3f G",s,s), "0 g 0 G"
1146   end
1147 end
1148 luamplib.colorconverter = colorconverter

```

2.2 T_EX package

```

1149  $\langle$ *package $\rangle$ 

```

First we need to load some packages.

```

1150 \bgroup\expandafter\expandafter\expandafter\egroup
1151 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
1152   \input luatexbase-modutils.sty
1153 \else
1154   \NeedsTeXFormat{LaTeX2e}
1155   \ProvidesPackage{luamplib}
1156   [2015/01/05 v2.9.0 mplib package for LuaTeX]
1157   \RequirePackage{luatexbase-modutils}
1158 \fi

```

Loading of lua code.

```

1159 \RequireLuaModule{luamplib}

```

Set the format for metapost.

```

1160 \def\mplibsetformat#1{%
1161   \directlua{luamplib.setformat("\luatexluaescapestring{#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.

```

1162 \ifnum\pdfoutput>0
1163   \let\mplibtoPDF\pdfliteral
1164 \else
1165   \def\mplibtoPDF#1{\special{pdf:literal direct #1}}
1166   \ifcsname PackageWarning\endcsname
1167     \PackageWarning{luamplib}{take dvipdfmx path, no support for other dvi tools currently.}
1168   \else
1169     \write16{}
1170     \write16{luamplib Warning: take dvipdfmx path, no support for other dvi tools currently.}
1171   \write16{}
1172 \fi
1173 \fi
1174 \def\mplibsetupcatcodes{%
1175   %catcode'\{=12 %catcode'\}=12
1176   \catcode'\#=12 \catcode'\^=12 \catcode'\-=12 \catcode'\_ =12
1177   \catcode'\&=12 \catcode'\$=12 \catcode'\%=12 \catcode'\^^M=12 \endlinechar=10
1178 }

```

Make btex...etex box zero-metric.

```

1179 \def\mplibputtextbox#1{\vbox to 0pt{\vss\hbox to 0pt{\raise\dp#1\copy#1\hss}}}
1180 \newcount\mplibstartlineno

```

```

1181 \def\mplibpostmpcatcodes{%
1182   \catcode'\{=12 \catcode'\}=12 \catcode'\#=12 \catcode'\%=12 }
1183 \def\mplibreplacenewlinebr{%
1184   \begingroup \mplibpostmpcatcodes \mplibdoreplacenewlinebr}
1185 \begingroup\lccode'\~='^^M \lowercase{\endgroup
1186   \def\mplibdoreplacenewlinebr#1^^J{\endgroup\luatexscantextokens{{}#1~}}

    The Plain-specific stuff.
1187 \bgroup\expandafter\expandafter\expandafter\egroup
1188 \expandafter\ifx\csname selectfont\endcsname\relax
1189 \def\mplibreplacenewlinecs{%
1190   \begingroup \mplibpostmpcatcodes \mplibdoreplacenewlinecs}
1191 \begingroup\lccode'\~='^^M \lowercase{\endgroup
1192   \def\mplibdoreplacenewlinecs#1^^J{\endgroup\luatexscantextokens{\relax#1~}}}
1193 \def\mplibcode{%
1194   \mplibstartlineno\inputlineno
1195   \begingroup
1196   \begingroup
1197   \mplibsetupcatcodes
1198   \mplibdocode
1199 }
1200 \long\def\mplibdocode#1\endmplibcode{%
1201   \endgroup
1202   \edef\mplibtemp{\directlua{luamplib.protecttexttext([===[\unexpanded{#1}]===])}}}%
1203   \directlua{ tex.sprint(table.concat(luamplib.mpxcolors)) }%
1204   \directlua{luamplib.tempdata = luamplib.makeTEXboxes([===[\mplibtemp]===])}%
1205   \directlua{luamplib.processwithTEXboxes(luamplib.tempdata)}%
1206   \endgroup
1207   \ifnum\mplibstartlineno<\inputlineno\expandafter\mplibreplacenewlinecs\fi
1208 }
1209 \else

    The  $\TeX$ -specific parts: a new environment.
1210 \newenvironment{mplibcode}{%
1211   \global\mplibstartlineno\inputlineno
1212   \toks@{}\ltxdomplibcode
1213 }{}
1214 \def\ltxdomplibcode{%
1215   \begingroup
1216   \mplibsetupcatcodes
1217   \ltxdomplibcodeindeed
1218 }
1219 \def\mplib@mplibcode{mplibcode}
1220 \long\def\ltxdomplibcodeindeed#1\end#2{%
1221   \endgroup
1222   \toks@\expandafter{\the\toks@#1}%
1223   \def\mplibtemp@a{#2}\ifx\mplib@mplibcode\mplibtemp@a
1224     \edef\mplibtemp{\directlua{luamplib.protecttexttext([===[\the\toks@]===])}}}%
1225     \directlua{ tex.sprint(table.concat(luamplib.mpxcolors)) }%
1226     \directlua{luamplib.tempdata=luamplib.makeTEXboxes([===[\mplibtemp]===])}%
1227     \directlua{luamplib.processwithTEXboxes(luamplib.tempdata)}%

```

```

1228 \end{mplibcode}%
1229 \ifnum\mplibstartlineno<\inputlineno
1230 \expandafter\expandafter\expandafter\mplibreplacelinebr
1231 \fi
1232 \else
1233 \toks@\expandafter{\the\toks@\end{#2}}\expandafter\ltxdomplibcode
1234 \fi
1235 }
1236 \fi

```

Support color/xcolor packages. User interface is: \mpcolor{teal}, for example.

```

1237 \def\mplibcolor#1#2{%
1238 \ifcsname\string\color @#2\endcsname
1239 \edef#1{1 withprescript
1240 "MplibOverrideColor=\csname\string\color @#2\endcsname"%
1241 \else
1242 \ifdefined\extractcolorspecs
1243 \extractcolorspecs{#2}\mplibtemp@a\mplibtemp@b
1244 \convertcolorspec\mplibtemp@a\mplibtemp@b{cmyk}\mplibtemp@c
1245 \edef#1{(\mplibtemp@c)}%
1246 \else
1247 \errmessage{Undefined color '#2'}%
1248 \fi
1249 \fi
1250 }

```

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

```

1251 \newtoks\everymplibtoks
1252 \newtoks\everyendmplibtoks
1253 \protected\def\everymplib{%
1254 \mplibstartlineno\inputlineno
1255 \begingroup
1256 \mplibsetupcatcodes
1257 \mplibdoeverymplib
1258 }
1259 \long\def\mplibdoeverymplib#1{%
1260 \endgroup
1261 \everymplibtoks{#1}%
1262 \ifnum\mplibstartlineno<\inputlineno\expandafter\mplibreplacelinebr\fi
1263 }
1264 \protected\def\everyendmplib{%
1265 \mplibstartlineno\inputlineno
1266 \begingroup
1267 \mplibsetupcatcodes
1268 \mplibdoeveryendmplib
1269 }
1270 \long\def\mplibdoeveryendmplib#1{%
1271 \endgroup
1272 \everyendmplibtoks{#1}%
1273 \ifnum\mplibstartlineno<\inputlineno\expandafter\mplibreplacelinebr\fi

```

```

1274 }
1275 \def\mpdim#1{ \begingroup \the\dimexpr #1\relax\space \endgroup } % gmp.sty
1276 \def\mplibnumbersystem#1{\directlua{luamplib.numbersystem = "#1"}}
1277 \def\mplibmakenocache#1{\mplibdomakenocache #1,*}
1278 \def\mplibdomakenocache#1,{%
1279   \ifx\empty#1\empty
1280     \expandafter\mplibdomakenocache
1281   \else
1282     \ifx*#1\else
1283       \directlua{luamplib.noneedtoreplace["#1.mp"]=true}%
1284       \expandafter\expandafter\expandafter\mplibdomakenocache
1285     \fi
1286   \fi
1287 }
1288 \def\mplibcancelnocache#1{\mplibdocancelnocache #1,*}
1289 \def\mplibdocancelnocache#1,{%
1290   \ifx\empty#1\empty
1291     \expandafter\mplibdocancelnocache
1292   \else
1293     \ifx*#1\else
1294       \directlua{luamplib.noneedtoreplace["#1.mp"]=false}%
1295       \expandafter\expandafter\expandafter\mplibdocancelnocache
1296     \fi
1297   \fi
1298 }
1299 \def\mplibcachedir#1{\directlua{luamplib.getcachedir("\unexpanded{#1}")}}
1300 \def\mplibtexttextlabel#1{%
1301   \begingroup
1302   \def\tempa{enable}\def\tempb{#1}%
1303   \ifx\tempa\tempb
1304     \directlua{luamplib.texttextlabel = true}%
1305   \else
1306     \directlua{luamplib.texttextlabel = false}%
1307   \fi
1308   \endgroup
1309 }
1310 \def\mplibcodeinherit#1{%
1311   \begingroup
1312   \def\tempa{enable}\def\tempb{#1}%
1313   \ifx\tempa\tempb
1314     \directlua{luamplib.codeinherit = true}%
1315   \else
1316     \directlua{luamplib.codeinherit = false}%
1317   \fi
1318   \endgroup
1319 }

```

We use a dedicated scratchbox.

```
1320 \ifx\mplibscratchbox\undefined \newbox\mplibscratchbox \fi
```

We encapsulate the literals.


```

1321 \def\mplibstarttoPDF#1#2#3#4{%
1322   \hbox\bgroup
1323   \xdef\MPllx{#1}\xdef\MPlly{#2}%
1324   \xdef\MPurx{#3}\xdef\MPury{#4}%
1325   \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
1326   \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
1327   \parskip0pt%
1328   \leftskip0pt%
1329   \parindent0pt%
1330   \everypar{}%
1331   \setbox\mplibscratchbox\vbox\bgroup
1332   \noindent
1333 }
1334 \def\mplibstoptoPDF{%
1335   \egroup %
1336   \setbox\mplibscratchbox\hbox %
1337     {\hskip-\MPllx bp%
1338      \raise-\MPlly bp%
1339      \box\mplibscratchbox}%
1340   \setbox\mplibscratchbox\vbox to \MPheight
1341     {\vfill
1342      \hsize\MPwidth
1343      \wd\mplibscratchbox0pt%
1344      \ht\mplibscratchbox0pt%
1345      \dp\mplibscratchbox0pt%
1346      \box\mplibscratchbox}%
1347   \wd\mplibscratchbox\MPwidth
1348   \ht\mplibscratchbox\MPheight
1349   \box\mplibscratchbox
1350   \egroup
1351 }

```

Text items have a special handler.

```

1352 \def\mplibtexttext#1#2#3#4#5{%
1353   \begingroup
1354   \setbox\mplibscratchbox\hbox
1355     {\font\temp=#1 at #2bp%
1356      \temp
1357      #3}%
1358   \setbox\mplibscratchbox\hbox
1359     {\hskip#4 bp%
1360      \raise#5 bp%
1361      \box\mplibscratchbox}%
1362   \wd\mplibscratchbox0pt%
1363   \ht\mplibscratchbox0pt%
1364   \dp\mplibscratchbox0pt%
1365   \box\mplibscratchbox
1366   \endgroup
1367 }

```

input luamplib.cfg when it exists

```
1368 \openin0=luamplib.cfg
1369 \ifeof0 \else
1370   \closein0
1371   \input luamplib.cfg
1372 \fi

      That's all folks!
1373 \end{package}
```

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.

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright © 1989, 1991 Free Software Foundation, Inc.

51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

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.

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.

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.

For example, if you distribute copies of such a program, whether gratis 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.

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.

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 author's reputations.

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.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

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".

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 it contains 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.

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.

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.

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:

(a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

(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 or no charge to all third parties under the terms of this License.

(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.)

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

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.

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.

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.

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:

(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,

(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 distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

(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 b above.)

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.

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.

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.

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 for 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.

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 to this License.

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.

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.

It is not the purpose of this section to induce you to infringe any patents or other property right 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 that 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.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

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.

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.

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.

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.

No WARRANTY

12. *BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHERE 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.*

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 RE-DISTRIBUTE 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.*

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

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.

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.

one line to give the program's name and a brief idea of what it does.
Copyright (C) yyyy name of author

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.

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.

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.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) yyyy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

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.

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:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program
"Gnomovision" (which makes passes at compilers) written by James
Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine 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.

35