The luamplib package

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

2013/09/24 v2.01

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.

The package needs to be in PDF mode in order to output something, as PDF specials are not supported by the DVI format and tools.

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

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.

Using this package is easy: in Plain, type your metapost code between the macros `mplibcode` and `endmplibcode`, and in LaTeX in the `mplibcode` environment.

There are (basically) two formats for metapost: *plain* and *mpfun*. 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. ConTEXt uses `metapost`.

```

luamplib          = luamplib or { }

```

Identification.

```

local luamplib    = luamplib
luamplib.showlog  = luamplib.showlog or false
luamplib.lastlog  = ""

local err, warn, info, log = luatexbase.provides_module({
    name          = "luamplib",
    version       =  2.01,
    date          = "2013/09/24",
    description   = "Lua package to typeset Metapost with LuaTeX's MPLib.",
})


```

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

```

local format, abs = string.format, math.abs

local stringgsub    = string.gsub
local stringfind    = string.find
local stringmatch   = string.match
local stringgmatch  = string.gmatch
local tableconcat   = table.concat
local texsprint     = tex.sprint

local mplib = require ('mplib')
local kpse  = require ('kpse')

local file = file
if not file then

```

This is a small trick for LATEX. In LATEX 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`.

```

```

```
  file = { }

  function file.replacesuffix(filename, suffix)
      return (stringgsub(filename,"%.[%a%d]+$","")) .. "." .. suffix
  end

  function file.stripsuffix(filename)
      return (stringgsub(filename,"%.[%a%d]+$",""))
  end
end
```

As the finder function for `mplib`, 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).

```

local mpkpse = kpse.new("luatex", "mpost")

local function finder(name, mode, ftype)
    if mode == "w" then
        return name
    else
        return mpkpse:find_file(name,ftype)
    end
end
luamplib.finder = finder

```

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.

```

function luamplib.resetlastlog()
    luamplib.lastlog = ""
end

```

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

```
local mplibone = tonumber(mplib.version()) <= 1.50

if mplibone then

    luamplib.make = luamplib.make or function(name,mem_name,dump)
        local t = os.clock()
        local mpx = mplib.new {
            ini_version = true,
            find_file = luamplib.finder,
            job_name = file.stripsuffix(name)
        }
        mpx:execute(format("input %s ;",name))
        if dump then
            mpx:execute("dump ;")
            info(”format %s made and dumped for %s in %0.3f seconds”,mem_name,name,os.clock()-t)
        else
            info(”%s read in %0.3f seconds”,name,os.clock()-t)
        end
        return mpx
    end

    function luamplib.load(name)
        local mem_name = file.replacesuffix(name,”mem”)
        local mpx = mplib.new {
            ini_version = false,
            mem_name = mem_name,
            find_file = luamplib.finder
        }
        if not mpx and type(luamplib.make) == ”function” then
            -- when i have time i’ll locate the format and dump
            mpx = luamplib.make(name,mem_name)
        end
        if mpx then
            info(”using format %s”,mem_name,false)
            return mpx, nil
        else
            return nil, { status = 99, error = ”out of memory or invalid format” }
        end
    end

else

```

These are the versions called with sufficiently recent mplib.

```

    local preamble = [[
        boolean mplib ; mplib := true ;
        let dump = endinput ;
        input %s ;
    ]]

    luamplib.make = luamplib.make or function()
    end

    function luamplib.load(name)
        local mpx = mplib.new {
            ini_version = true,
            find_file = luamplib.finder,
        }
        local result
        if not mpx then
            result = { status = 99, error = ”out of memory”}
        else
            result = mpx:execute(format(preamble, file.replacesuffix(name,”mp”)))
        end
        luamplib.reporterror(result)
        return mpx, result
    end

end

local currentformat = ”plain”

local function setformat (name) --- used in .sty
    currentformat = name
end
luamplib.setformat = setformat


luamplib.reporterror = function (result)
    if not result then
        err(”no result object returned”)
    elseif result.status > 0 then
        local t, e, l = result.term, result.error, result.log
        if t then
            info(t)
        end
        if e then
            err(e)
        end
        if not t and not e and l then
            luamplib.lastlog = luamplib.lastlog .. ”\n ” .. l
            log(l)
        else
            err(”unknown, no error, terminal or log messages”)
        end
    else
        return false
    end
    return true
end

local function process_indeed (mpx, data)
    local converted, result = false, {}
    local mpx = luamplib.load(mpx)
    if mpx and data then
        local result = mpx:execute(data)
        if not result then
            err(”no result object returned”)
        elseif result.status > 0 then
            err(”%s”,(result.term or ”no-term”) .. ”\n” .. (result.error or ”no-error”))
        elseif luamplib.showlog then
            luamplib.lastlog = luamplib.lastlog .. ”\n” .. result.term
            info(”%s”,result.term or ”no-term”)
        elseif result.fig then
            converted = luamplib.convert(result)
        else
            err(”unknown error, maybe no beginfig/endfig”)
        end
    else
        err(”Mem file unloadable. Maybe generated with a different version of mplib?”)
    end
    return converted, result
end
local process = function (data)
    return process_indeed(currentformat, data)
end
luamplib.process = process

local function getobjects(result,figure,f)
    return figure:objects()
end

local function convert(result, flusher)
    luamplib.flush(result, flusher)
    return true -- done
end
luamplib.convert = convert

local function pdf_startfigure(n,llx,lly,urx,ury)
```

The following line has been slightly modified by Kim.

```
    texsprint(format(”\\mplibstarttoPDF{%f}{%f}{%f}{%f}”,llx,lly,urx,ury))
end

local function pdf_stopfigure()
    texsprint(”\\mplibstoptoPDF”)
end

local function pdf_literalcode(fmt,...) -- table
    texsprint(format(”\\mplibtoPDF{%s}”,format(fmt,...)))
end
luamplib.pdf_literalcode = pdf_literalcode

local function pdf_textfigure(font,size,text,width,height,depth)
    text = text:gsub(”.”,”\\hbox{%1}”) -- kerning happens in metapost
```

The following line has been slightly modified by Kim.

```
    texsprint(format(”\\mplibtextext{%s}{%f}{%s}{%s}{%f}”,font,size,text,0,-( 7200/ 7227)/65536*depth)
end
luamplib.pdf_textfigure = pdf_textfigure

local bend_tolerance = 131/65536

local rx, sx, sy, ry, tx, ty, divider = 1, 0, 0, 1, 0, 0, 1

local function pen_characteristics(object)
local t = mplib.pen_info(object)
rx, ry, sx, sy, tx, ty = t.rx, t.ry, t.sx, t.sy, t.tx, t.ty
divider = sx*sy - rx*ry
return not (sx==1 and rx==0 and ry==0 and sy==1 and tx==0 and ty==0), t.width
end

local function concat(px, py) -- no tx, ty here
return (sy*px-ry*py)/divider,(sx*py-rx*px)/divider
end

local function curved(ith,pth)
local d = pth.left_x - ith.right_x
if abs(ith.right_x - ith.x_coord - d) <= bend_tolerance and abs(pth.x_coord - pth.left_x - d) <= be
erance then
d = pth.left_y - ith.right_y
if abs(ith.right_y - ith.y_coord - d) <= bend_tolerance and abs(pth.y_co-
ord - pth.left_y - d) <= bend_tolerance then
return false
end
end
return true
end

local function flushnormalpath(path,open)
local pth, ith
for i=1,#path do
pth = path[i]
if not ith then
pdf_literalcode(”%f %f m”,pth.x_coord,pth.y_coord)
elseif curved(ith,pth) then
pdf_literalcode(”%f %f %f %f %f %f c”,ith.right_x,ith.right_y,pth.left_x,pth.left_y,pth.x_c
else
pdf_literalcode(”%f %f l”,pth.x_coord,pth.y_coord)
end
ith = pth
end
if not open then
local one = path[1]
if curved(pth,one) then
pdf_literalcode(”%f %f %f %f %f %f c”,pth.right_x,pth.right_y,one.left_x,one.left_y,one.x_c
else
pdf_literalcode(”%f %f l”,one.x_coord,one.y_coord)
end
elseif #path == 1 then
-- special case .. draw point
local one = path[1]
pdf_literalcode(”%f %f l”,one.x_coord,one.y_coord)
end
return t
end

local function flushconcatpath(path,open)
pdf_literalcode("%f %f %f %f %f %f cm", sx, rx, ry, sy, tx ,ty)
local pth, ith
for i=1,#path do
pth = path[i]
if not ith then
   pdf_literalcode("%f %f m",concat(pth.x_coord,pth.y_coord))
elseif curved(ith,pth) then
local a, b = concat(ith.right_x,ith.right_y)
local c, d = concat(pth.left_x,pth.left_y)
pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(pth.x_coord, pth.y_co-
ord))
else
   pdf_literalcode("%f %f l",concat(pth.x_coord, pth.y_coord))
end
ith = pth
end
if not open then
local one = path[1]
if curved(pth,one) then
local a, b = concat(pth.right_x,pth.right_y)
local c, d = concat(one.left_x,one.left_y)
pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(one.x_coord, one.y_co-
ord))
else
pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
end
elseif #path == 1 then
-- special case .. draw point
local one = path[1]
pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
end
return t
end

```

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

```

local mplibcodepreamble = [[
vardef textext@#(expr n, w, h, d) =
    image(
        addto currentpicture doublepath unitsquare
        xscaled w
        yscaled (h+d)
        shifted (0,-d)
        withprescript "%%textext:"&decimal n&":"&decimal w&":"&decimal(h+d);
    )
enddef;
]]

local factor = 65536*(7227/7200)

local function settexboxes (data)
    local i = tex.count[14] -- newbox register
    for _,c,_ in stringgmatch(data,"(%A)btex(%A.-%A)etex(%A)") do
        i = i + 1
        c = stringgsub(c,"^%s*(.-)%s*$","%1")
        texsprint(format("\\setbox%i\\hbox{%s}",i,c))
    end
end

luamplib.settexboxes = settexboxes

local function gettexboxes (data)
    local i = tex.count[14] -- the same newbox register
    data = stringgsub(data,"(%A)btex%A.-%Aetex(%A)",
    function(pre,post)
        i = i + 1
        local box = tex.box[i]
        local boxmetr = format("textext(%i,%f,%f,%f)",
                        i,
                        box and (box.width/factor) or 0,
                        box and (box.height/factor) or 0,
                        box and (box.depth/factor) or 0)
        return pre .. boxmetr .. post
    end)
    return mplibcodepreamble .. data
end

luamplib.gettexboxes = gettexboxes

local function puttexboxes (object)
    local n,tw,th = stringmatch(
                    object.prescript,
                    "%%%%textext:(%d+):([%d%.%+%-]+):([%d%.%+%-]+)")
    if n and tw and th then
        local op = object.path
        local first, second, fourth = op[1], op[2], op[4]
        local tx, ty = first.x_coord, first.y_coord
        local sx, sy = (second.x_coord - tx)/tw, (fourth.y_coord - ty)/th
        local rx, ry = (second.y_coord - ty)/tw, (fourth.x_coord - tx)/th
        if sx == 0 then sx = 0.00001 end
        if sy == 0 then sy = 0.00001 end
        pdf_literalcode("q %f %f %f %f %f %f cm",sx,rx,ry,sy,tx,ty)
        texsprint(format("\\mplibputtextbox{%i}",n))
        pdf_literalcode("Q")
    end
end

```

End of btex – etex patch.

```

local function flush(result,flusher)
    if result then
        local figures = result.fig
        if figures then
            for f=1, #figures do
                info("flushing figure %s",f)
                local figure = figures[f]
                local objects = getobjects(result,figure,f)
                local fignum = tonumber(stringmatch(figure:filename(),"([%d]+)$") or fig-
ure:charcode() or 0)
                local miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
                local bbox = figure:boundingbox()
                local llx, lly, urx, ury = bbox[1], bbox[2], bbox[3], bbox[4] -- faster than un-
pack
                if urx < llx then
                    -- invalid
                    pdf_startfigure(fignum,0,0,0,0)
                    pdf_stopfigure()
                else
                    pdf_startfigure(fignum,llx,lly,urx,ury)
                    pdf_literalcode("q")
                    if objects then
                        for o=1,#objects do
                            local object        = objects[o]
                            local objecttype    = object.type
```

Change from ConTEXt code: the following 3 lines are part of the btex...etex patch.

```
                            local prescript     = object.prescript --- [be]tex patch
                            if prescript and stringfind(prescript,"%%%%textext:") then
                               puttexboxes(object)
                            elseif objecttype == "start_bounds" or objecttype == "stop_bounds" then
                                -- skip
                            elseif objecttype == "start_clip" then
                                pdf_literalcode("q")
                                flushnormalpath(object.path,t,false)
                                pdf_literalcode("W n")
                            elseif objecttype == "stop_clip" then
                                pdf_literalcode("Q")
                                miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
                            elseif objecttype == "special" then
                                -- not supported
                            elseif objecttype == "text" then
                                local ot = object.transform -- 3,4,5,6,1,2
                                pdf_literalcode("q %f %f %f %f %f %f cm",ot[3],ot[4],ot[5],ot[6],ot[1],
                                pdf_textfigure(object.font,object.dsize,object.text,object.width,object
                                pdf_literalcode("Q")
                            else
                                local cs = object.color
```

Change from ConTEXt code: the following 5 lines are part of the `btex...etex` patch.

```
                                local cr = nil
                                if cs and #cs > 0 then
                                    local start,stop = luamplib.colorconverter(cs)
                                    pdf_literalcode(start)
                                    cr = stop
                                end
                                local ml = object.miterlimit
                                if ml and ml ~= miterlimit then
                                    miterlimit = ml
                                    pdf_literalcode("%f M",ml)
                                end
                                local lj = object.linejoin
                                if lj and lj ~= linejoin then
                                    linejoin = lj
                                    pdf_literalcode("%i j",lj)
                                end
                                local lc = object.linecap
                                if lc and lc ~= linecap then
                                    linecap = lc
                                    pdf_literalcode("%i J",lc)
                                end
                                local dl = object.dash
                                if dl then
                                    local d = format("[%s] %i d",tableconcat(dl.dashes or {}," "),dl.o
                                    if d ~= dashed then
                                        dashed = d
                                        pdf_literalcode(dashed)
                                    end
                                elseif dashed then
                                   pdf_literalcode("[] 0 d")
                                   dashed = false
                                end
                                local path = object.path
                                local transformed, penwidth = false, 1
                                local open = path and path[1].left_type and path[#path].right_type
                                local pen = object.pen
                                if pen then
                                   if pen.type == 'elliptical' then
                                        transformed, penwidth = pen_characteristics(object) -- boolean, value
                                        pdf_literalcode("%f w",penwidth)
                                        if objecttype == 'fill' then
                                            objecttype = 'both'
                                        end
```

```
            else -- calculated by mplib itself
                 objecttype = ’fill’
            end
         end
         if transformed then
             pdf_literalcode(”q”)
         end
         if path then
             if transformed then
                 flushconcatpath(path,open)
             else
                 flushnormalpath(path,open)
             end
             if objecttype == ”fill” then
                 pdf_literalcode(”h f”)
             elseif objecttype == ”outline” then
                 pdf_literalcode((open and ”S”) or ”h S”)
             elseif objecttype == ”both” then
                 pdf_literalcode(”h B”)
             end
         end
         if transformed then
             pdf_literalcode(”Q”)
         end
         local path = object.htap
         if path then
             if transformed then
                 pdf_literalcode(”q”)
             end
             if transformed then
                 flushconcatpath(path,open)
             else
                 flushnormalpath(path,open)
             end
             if objecttype == ”fill” then
                 pdf_literalcode(”h f”)
             elseif objecttype == ”outline” then
                 pdf_literalcode((open and ”S”) or ”h S”)
             elseif objecttype == ”both” then
                 pdf_literalcode(”h B”)
             end
             if transformed then
                 pdf_literalcode(”Q”)
             end
         end
         if cr then
             pdf_literalcode(cr)
         end
     end
end
```

```
                    end
                    pdf_literalcode("Q")
                    pdf_stopfigure()
                end
            end
        end
    end
end
luamplib.flush = flush

local function colorconverter(cr)
    local n = #cr
    if n == 4 then
        local c, m, y, k = cr[1], cr[2], cr[3], cr[4]
        return format("%.3f %.3f %.3f %.3f k %.3f %.3f %.3f %.3f K",c,m,y,k,c,m,y,k), "0 g 0 G"
    elseif n == 3 then
        local r, g, b = cr[1], cr[2], cr[3]
        return format("%.3f %.3f %.3f rg %.3f %.3f %.3f RG",r,g,b,r,g,b), "0 g 0 G"
    else
        local s = cr[1]
        return format("%.3f g %.3f G",s,s), "0 g 0 G"
    end
end
luamplib.colorconverter = colorconverter
```

2.2 TeX package

```
⟨*package⟩
```

First we need to load fancyvrb, to define the environment mplibcode.

```
\bgroup\expandafter\expandafter\expandafter\egroup
\expandafter\ifx\csname ProvidesPackage\endcsname\relax
  \input luatexbase-modutils.sty
\else
  \NeedsTeXFormat{LaTeX2e}
  \ProvidesPackage{luamplib}
    [2013/09/24 v2.01 mplib package for LuaTeX]
  \RequirePackage{luatexbase-modutils}
  \RequirePackage{pdftexcmds}
\fi
```

Loading of lua code.

```
\RequireLuaModule{luamplib}
```

Set the format for metapost.

```
\def\mplibsetformat#1{%
  \directlua{luamplib.setformat("\luatexluaescapestring{#1}")}}
```

MPLib only works in PDF mode, we don't do anything if we are in DVI mode, and we output a warning.

```
\ifnum\pdfoutput>0
```

```
    \let\mplibtoPDF\pdfliteral
\else
    %\def\MPLIBtoPDF#1{\special{pdf:literal direct #1}} % not ok yet
    \def\mplibtoPDF#1{}
    \expandafter\ifx\csname PackageWarning\endcsname\relax
      \write16{}
      \write16{Warning: MPLib only works in PDF mode, no figure will be output.}
      \write16{}
    \else
      \PackageWarning{mplib}{MPLib only works in PDF mode, no figure will be out-
put.}
    \fi
\fi
\def\mplibsetupcatcodes{%
  \catcode‘\{=12 \catcode‘\}=12 \catcode‘\#=12
  \catcode‘\^=12 \catcode‘\~=12 \catcode‘\_=12
  %catcode‘\%=12 %% don’t in Plain!
  \catcode‘\&=12 \catcode‘\$=12
}
```

The Plain-specific stuff.

```
\def\mplibputtextbox#1{\vbox to 0pt{\vss\hbox to 0pt{\raise\dp#1\box#1\hss}}}
\bgroup\expandafter\expandafter\expandafter\egroup
\expandafter\ifx\csname ProvidesPackage\endcsname\relax
\def\mplibcode{%
  \bgroup
  \mplibsetupcatcodes
  \mplibdocode %
}
\long\def\mplibdocode#1\endmplibcode{%
  \egroup
  \directlua{
    luamplib.settexboxes([===[\unexpanded{#1}]===])
  }%
  \directlua{
    local data = luamplib.gettexboxes([===[\unexpanded{#1}]===])
    luamplib.process(data)
  }%
}
\else
```

The LaTeX-specific parts. First a Hack for the catcodes in LaTeX.

```
\begingroup
\catcode‘\,=13
\catcode‘\-=13
\catcode‘\<=13
\catcode‘\>=13
\catcode‘\^^I=13
\catcode‘\‘=13 % must be last...
\gdef\FV@hack{%
  \def,{\string,}%
\def-{\string-}%
\def<{\string<}%
\def>{\string>}%
\def‘{\string‘}%
\def^^I{\string^^I}%
}
\endgroup
```

The LaTeX environment.

```
\newenvironment{mplibcode}{\toks@{}\ltxdomplibcode}{}
\def\ltxdomplibcode{%
  \bgroup
  \mplibsetupcatcodes
  \ltxdomplibcodeindeed %
}
%
\long\def\ltxdomplibcodeindeed#1\end{%
  \egroup
  \toks@\expandafter{\the\toks@#1}\ltxdomplibcodefinally%
}%
%
\def\ltxdomplibcodefinally#1{%
  \ifnum\pdf@strcmp{#1}{mplibcode}=\z@
    \directlua{luamplib.settexboxes([===[\the\toks@]===])}%
    \directlua{
      local data = luamplib.gettexboxes([===[\the\toks@]===])
      luamplib.process(data)
    }%
    \end{mplibcode}%
  \else
    \toks@\expandafter{\the\toks@\end{#1}}\expandafter\ltxdomplibcode
 \fi%
}
\fi
```

We use a dedicated scratchbox.

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

We encapsulate the litterals.

```
\def\mplibstarttoPDF#1#2#3#4{%
  \hbox\bgroup
  \xdef\MPllx{#1}\xdef\MPlly{#2}%
  \xdef\MPurx{#3}\xdef\MPury{#4}%
  \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
  \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
  \parskip0pt%
  \leftskip0pt%
  \parindent0pt%
  \everypar{}%
  \setbox\mplibscratchbox\vbox\bgroup
  \noindent
}
```

```
\def\mplibstoptoPDF{%
  \egroup %
  \setbox\mplibscratchbox\hbox %
    {\hskip-\MPllx bp%
     \raise-\MPlly bp%
     \box\mplibscratchbox}%
  \setbox\mplibscratchbox\vbox to \MPheight
    {\vfill
     \hsize\MPwidth
     \wd\mplibscratchbox0pt%
     \ht\mplibscratchbox0pt%
     \dp\mplibscratchbox0pt%
     \box\mplibscratchbox}%
  \wd\mplibscratchbox\MPwidth
  \ht\mplibscratchbox\MPheight
  \box\mplibscratchbox
  \egroup
}
```

Text items have a special handler.

```
\def\mplibtextext#1#2#3#4#5{%
  \begingroup
  \setbox\mplibscratchbox\hbox
    {\font\temp=#1 at #2bp%
     \temp
     #3}%
  \setbox\mplibscratchbox\hbox
    {\hskip#4 bp%
     \raise#5 bp%
     \box\mplibscratchbox}%
  \wd\mplibscratchbox0pt%
  \ht\mplibscratchbox0pt%
  \dp\mplibscratchbox0pt%
  \box\mplibscratchbox
  \endgroup
}
```

That's all folks!

⟨/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 authors' 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 saying 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 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.

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 at 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 (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.

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 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.*

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 redistribute 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.