

Zpravodaj Československého sdružení uživatelů TeXu

Luigi Scarso

MicroTalk – pdfsplit

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 21 (2011), No. 2-4, 102–115

Persistent URL: <http://dml.cz/dmlcz/150179>

Terms of use:

© Československé sdružení uživatelů TeXu, 2011

Institute of Mathematics of the Czech Academy of Sciences provides access to digitized documents strictly for personal use. Each copy of any part of this document must contain these *Terms of use*.



This document has been digitized, optimized for electronic delivery and stamped with digital signature within the project *DML-CZ*:
The Czech Digital Mathematics Library <http://dml.cz>

Abstract: MicroTalk is a short and technical paper that shows some unusual, hopefully useful, ideas following the schema “figure to code”. The main topic is always typographic programming in ConTeXt & Lua. A bit of Lua code, the `\clip` macro and Leptonica extensions are the ingredients for this recipe to cook a pdfsplit macro that take a pdf and try to split into parts as the `\vsplit` does with `\vboxes`.

Key words: Lua, Leptonica, PDF slicing, SWIG, MuPDF, Sumatra.

Abstrakt: Článek poukazuje na způsob nařezání PDF na proužky tak, jak je to známé pomocí příkazu `\vsplit` s `\vboxy`.

Klíčová slova: Lua, Leptonica, vysekávání z PDF, SWIG, MuPDF, Sumatra.

References

- [1] Leptonica. Available at URL: <http://www.leptonica.com/>
- [2] MuPDF – a lightweight PDF viewer and toolkit written in portable C. Available at URL: <http://www.mupdf.com/>
- [3] Sumatra PDF. Available at URL: http://en.wikipedia.org/wiki/Sumatra_PDF

luigi (dot) scarso (at) gmail (dot) com
Padova, Italy

1 Slice a pdf

Let's start with a bit of Lua code:

```
\startluacode
document.lscarlo = document.lscarlo or {}
function document.lscarlo.LuaSliceIt(Fig,H,W,L,FigOpt)
  local H = math.floor(string.gsub(H,"pt","")*2^16)
  local W = math.floor(string.gsub(W,"pt","")*2^16)
  local L = math.floor(string.gsub(L,"pt","")*2^16)
  local vh = 0
  local step = L
  local h = step
  local S = ""
  local FigOpt = FigOpt or ""
  while vh <= H do
    S =string.format("{\\clip[voffset=\\%dsp,
                      width=\\%dsp,
                      height=\\%dsp]
                      {\\externalfigure[\\%s][\\%s]}}
                      \\par\\nointerlineskip\\blank[1sp]",
                      vh,W,h,Fig,FigOpt)
    tex.sprint(tex.ctxcatcodes,S)
    vh = vh + step
  end
  if math.mod(H,step) > 0 then
    vh = vh - step + math.mod(H,step)
    S =string.format("{\\clip[voffset=\\%dsp,
                      width=\\%dsp,
                      height=\\%dsp]
                      {\\externalfigure[\\%s][\\%s]}}
                      \\par\\nointerlineskip\\blank[1sp]",
                      vh,W,math.mod(H,step),Fig,FigOpt)
    tex.sprint(tex.ctxcatcodes,S)
  end
end
\stopluacode
```

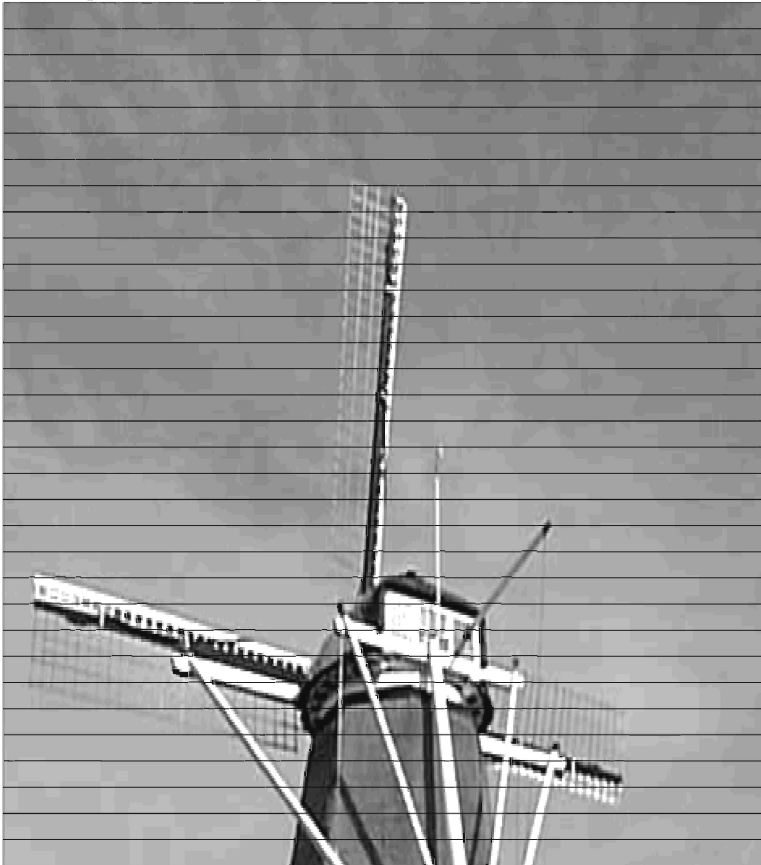
document.lscarlo.LuaSliceIt takes a pdf and “slices it” with slices of height L.

```

\bggroup
\newdimen\Hfig \newdimen\Wfig
\setbox1000=\hbox{\externalfigure[mill.png] [width=\textwidth,
                                                height=1.3\textheight]}
\Wfig=\wd1000 \Hfig=\dimexpr\ht1000+\dp1000\relax%
\ctxlua{document.lscarso.LuaSliceIt("mill.png","\the\Hfig",
                                     "\the\Wfig","\the\dimexpr 1.0\lineheight\relax",
                                     "width=\textwidth,height=1.3\textheight")}
\egroup

```

It's not a problem with images:





But uniform slicing is wrong with text:

$\sum_{x=0}^{10} \frac{1}{x+1} = \frac{83711}{27720} = 3.019877344877344877344877345$
$\sum_{x=0}^{100} \frac{1}{x+1} = \frac{1463919079740743966268954674710929768361083}{281670315928039407744716588098661706369472} = 5.197278507738630161795216686$
$\sum_{x=0}^{150} \frac{1}{x+1} = \frac{4195569667676135811153969815137073234944561746732919339914337201927}{749502901196827228266820481792118993292919127408542808267329424000} = 5.597803105200170187965715973$
$+ \sum_{x=0}^{30} \frac{-1^x}{2x+1} = \frac{58630135791001973169852284}{18472920064106597929865025} = 3.173842337190749408690224140$
$+ \sum_{x=0}^{200} \frac{-1^x}{2x+1} = 3.144914903558851799204586212$
$+ \sum_{x=0}^{300} \frac{-1^x}{2x+1} = 3.141925675689790151271200075$
$+ \sum_{x=0}^{3000} \frac{-1^x}{2x+1} = 3.141625985812043238153993692$
$+ \sum_{x=0}^{30000} \frac{-1^x}{2x+1} = 3.141593956917015489462612519$

$\sum_{x=0}^{10} \frac{1}{2x+1}$	
$4 \sum_{x=0}^{300000} \frac{-1^x}{2x+1} = 3.141592986923015460712643380$	

We can enumerate each slice, and observe that it depends only on the given step:
with `step = 1.0\lineheight` we have

$\sum_{x=0}^{10} \frac{1}{x+1} = \frac{83711}{27720} = 3.019877344877344877344877345$	001
	002
$\sum_{x=0}^{100} \frac{1}{x+1} = \frac{1463919079240743966268954674710929768361083}{281670315928038407744716598098661706369472} = 5.197278507738630161795216686$	003
	004
$\sum_{x=0}^{100} \frac{1}{x+1} = \frac{4195569667676135811153969815137073234944561746732919339914337201927}{749502901196827228266820481792118993292919127408542808267329424000} = 5.597803105200170187965715973$	005
	006
$\sum_{x=0}^{10} \frac{-1^x}{2x+1} = \frac{58630135791001973169852284}{18472920064106597929865025} = 3.173842337190749408690224140$	007
	008
$\sum_{x=0}^{300} \frac{-1^x}{2x+1} = 3.144914903558851799204586212$	009
	010
$\sum_{x=0}^{3000} \frac{-1^x}{2x+1} = 3.141925075039790154271200075$	011
	012
$\sum_{x=0}^{30000} \frac{-1^x}{2x+1} = 3.141625985812043238153992692$	013
	014
$4 \sum_{x=0}^{300000} \frac{-1^x}{2x+1} = 3.141595986912015480462612518$	015
	016
$4 \sum_{x=0}^{300000} \frac{-1^x}{2x+1} = 3.141592986923015460712643380$	017
	018
	019
	020

It's now clear that 002, 004, 006... are good break points while 001, 003, 005... must be avoided.. if we collect all good break points we can then subdivide the pdf into the right \vboxes. These break points depend on `lineheight`, but we can choose another `step`, let's say `2mm`, so that slices are independent from the body font:

```
\bgroup
\newdimen\Hfig \newdimen\Wfig
\setbox1000=\hbox{\externalfigure[pari_example.pdf][width=\textwidth]}
\Wfig=\wd1000 \Hfig=\dimexpr\ht1000+\dp1000\relax%
\ctxlua[document.lscarlo.LuaSliceItNR("pari_example.pdf", "\the\Hfig",
"\the\Wfig", "\the\dimexpr 2mm\relax",
"width=\textwidth")]
\egroup
```

$\sum_{x=0}^{10} \frac{1}{x+1} = \frac{83711}{27720} = 3.019877344877344877344877345$	001
	002
	003
	004
$\sum_{x=0}^{100} \frac{1}{x+1} = \frac{1463919079240743966268954674710929768361083}{281670315928038407744716598098661706369472} = 5.197278507738630161795216686$	005
	006
	007
	008
	009
	010

$\sum_{x=1}^{170} \frac{1}{x+1} = \frac{4195569667676135811153969815137073234944561746732919339914337201927}{749502901196827220266820481792110993292919127408542808267329424000} = 5.597803105200170187965715973$	011
$\sum_{x=0}^{10} \frac{1-x}{2x+1} = \frac{58630135791001973169852204}{18472920064106597929865025} = 3.173842337190749408690224140$	012
$\sum_{x=0}^{300} \frac{-1^x}{2x+1} = 3.14491490358051799204586212$	012
$\sum_{x=0}^{1000} \frac{-1^x}{2x+1} = 3.141925875839790151271200075$	012
$\sum_{x=0}^{30000} \frac{-1^x}{2x+1} = 3.141625985812043238153993692$	018
$\sum_{x=0}^{300000} \frac{-1^x}{2x+1} = 3.141595986912015488462612519$	020
$\sum_{x=0}^{3000000} \frac{-1^x}{2x+1} = 3.141592986923015460712643380$	021
	022
	023
	025
	026
	027
	028
	030
	031
	032
	034
	035
	036
	038
	040
	042

We group together slices 1, 5, 9, 13, 14, 18, 22, 26, 31, 35, 39:

$\sum_{x=0}^{10} \frac{1}{x+1} = \frac{83711}{27720} = 3.019877344877344877344877345$	001
$\sum_{x=0}^{100} \frac{1}{x+1} = \frac{1463919079240743966268954674710929768361083}{281670315928038407744716588098661706369472} = 5.197278507738630161795216686$	005
$\sum_{x=0}^{150} \frac{1}{x+1} = \frac{4195569667676135811153969815137073234944561746732919339914337201927}{749502901196827220266820481792110993292919127408542808267329424000} = 5.597803105200170187965715973$	009
$\sum_{x=0}^{10} \frac{-1^x}{2x+1} = \frac{58630135791001973169852204}{18472920064106597929865025} = 3.173842337190749408690224140$	012
$\sum_{x=0}^{300} \frac{-1^x}{2x+1} = 3.14491490358051799204586212$	012
$\sum_{x=0}^{1000} \frac{-1^x}{2x+1} = 3.141925875839790151271200075$	018
$\sum_{x=0}^{30000} \frac{-1^x}{2x+1} = 3.141625985812043238153993692$	022
$\sum_{x=0}^{300000} \frac{-1^x}{2x+1} = 3.141595986912015488462612519$	026
$\sum_{x=0}^{3000000} \frac{-1^x}{2x+1} = 3.141592986923015460712643380$	031
	035
	039
	042

Now slicing works well:

$\sum_{x=0}^{10} \frac{1}{x+1} = \frac{83711}{27720} = 3.019877344877344877344877345$	001
$\sum_{x=0}^{100} \frac{1}{x+1} = \frac{1463919079240743966268954674710929768361083}{281670315928038407744716588098661706369472} = 5.197278507738630161795216686$	005
$\sum_{x=0}^{150} \frac{1}{x+1} = \frac{4195569667676135811153969815137073234944561746732919339914337201927}{749502901196827220266820481792110993292919127408542808267329424000} = 5.597803105200170187965715973$	009
$\sum_{x=0}^{10} \frac{-1^x}{2x+1} = \frac{58630135791001973169852204}{18472920064106597929865025} = 3.173842337190749408690224140$	012
	012
	018

$4 \sum_{x=0}^{300} \frac{-1^x}{2x+1} = 3.144914903558851799204586212$	022
$4 \sum_{x=0}^{3000} \frac{-1^x}{2x+1} = 3.141925875839790151271200075$	026
$4 \sum_{x=0}^{30000} \frac{-1^x}{2x+1} = 3.141625985812043238153993692$	031
$4 \sum_{x=0}^{300000} \frac{-1^x}{2x+1} = 3.141595986912015488462612519$	035
$4 \sum_{x=0}^{3000000} \frac{-1^x}{2x+1} = 3.141592986923015460712643380$	039
	042

```

\bggroup
\newdimen\Hfig \newdimen\Wfig
\setbox1000=\hbox{\externalfigure[pari_example.pdf][width=\textwidth]}
\Wfig=\wd1000 \Hfig=\dimexpr\ht1000+\dp1000\relax%
\startluacode
document.lscarlo.GoodPoints = {1,5,9,13,14,18,22,26,31,35,39}
\stopluacode
\ctxlua{document.lscarlo.LuaSliceItAndCollect("pari_example.pdf",
"\the\Hfig","\the\Wfig","\the\dimexpr 2mm\relax",
"width=\textwidth","")}
\egroup

function document.lscarlo.LuaSliceItAndCollect(Fig,H,W,L,FigOpt)
local H = math.floor(string.gsub(H,"pt","")*2^16)
local W = math.floor(string.gsub(W,"pt","")*2^16)
local L = math.floor(string.gsub(L,"pt","")*2^16)
local vh = 0
local step = L
local h = step
local S = ""
local FigOpt = FigOpt or ""
local NR = 0
local Payload = "{\\ruledhbox{\\clip[voffset=\\dsp,width=\\dsp,height=\\dsp]
{\\externalfigure[\\s][\\s]}\\llap{\\tex\\%03d}}}"
\\par\\nointerlineskip\\blank[1sp]"
local GoodPoint = document.lscarlo.GoodPoints or {}
local j = 1
local Vboxes = {}
local prevvh = 0
while vh < H do
NR =NR +1
vh = vh + step
if GoodPoint[j] == NR then
Vboxes[#Vboxes+1] = {from=prevvh,to=vh,mark=NR}
--tex.sprint(tex.ctxcatcodes,"(",prevvh,",",vh,") ")

```



```

        prevvh = vh
        j = j +1
    end
end
if math.mod(H,step) == 0 then H = H +1  end
if math.mod(H,step) > 0 then
    NR = NR +1
    vh = vh - step + math.mod(H,step)
    Vboxes[#Vboxes+1] = {from=prevvh,to=vh,mark=NR}
end
Payload = "\\ruledvbox{\\hbox{\\clip[voffset=\\%dsp,width=\\%dsp,height=\\%dsp]
{\\externalfigure[\\%s][\\%s]}\\llap{\\tfx\\%03d}}
\\par\\nointerlineskip\\blank[1sp]"
for i,v in ipairs(Vboxes) do
    S =string.format(Payload,v.from,W,v.to-v.from,Fig,FigOpt,v.mark)
    tex.sprint(tex.ctxcatcodes,S)
end

```

It's now rather trivial to modify this macro to build a `\vbox` to be used with `\vsplit`.

2 Leptonica

Leptonica (<http://www.leptonica.com>) is a pedagogically-oriented open source site containing software that is useful for image processing and image analysis applications. It's not difficult to build a binding for Lua with SWIG. To use Leptonica we must first convert the pdf in a black and white bitmap with

```
pdftoppm -mono -r 72 -f 1 -l 1 pari_example.pdf pari_example
```

(this saves page 1 in `pari_example-000001.pbm` with a resolution of 72dpi). Then we scan the bitmap with `lept_get_breaks` searching for a white row and we store the y coordinates in `GoodPoints`. In the end `LuaCollect` builds the appropriate vboxes.

```
\bgroup
\newdimen\Hfig \newdimen\Wfig
\setbox1000=\hbox{\externalfigure[pari_example.pdf][width=\textwidth]}
\Wfig=\wd1000\relax\Hfig=\dimexpr\ht1000+\dp1000\relax%
\executesystemcommand{pdftoppm -mono -r 72 -f 1 -l 1 pari_example.pdf
pari_example}
\ctxlua{document.lscarlo.GoodPoints =
    document.lscarlo.lept_get_breaks("pari_example-000001.pbm",72)}
\ctxlua{document.lscarlo.LuaCollect("pari_example.pdf",
    "\the\Hfig","\the\Wfig",[width=\textwidth])}
\egroup
```

Let's see what happens:

$\sum_{x=0}^{100} \frac{1}{x+1} = \frac{83711}{27720} = 3.019877344877344877344877345$
$\sum_{x=0}^{1000} \frac{1}{x+1} = \frac{1463919079240743966268954674710929768361083}{281670315928038407744716588098661706369472} = 5.197278507738630161795216686$
$\sum_{x=0}^{10000} \frac{1}{x+1} = \frac{4195569667676135811153969815137073234944561746732919339914337201927}{749502901196827228266820481792118993292919127408542808267329424000} = 5.597803105200170187965715973$
$\sum_{x=0}^{100} \frac{-1^x}{2x+1} = \frac{58630135791001973169852284}{10472920064106597929865025} = 3.173842337190749408690224140$
$\sum_{x=0}^{1000} \frac{-1^x}{2x+1} = 3.144914903558851799204586212$
$\sum_{x=0}^{10000} \frac{-1^x}{2x+1} = 3.141925875839790151271200075$
$\sum_{x=0}^{100000} \frac{-1^x}{2x+1} = 3.141625985812043238153993692$
$\sum_{x=0}^{1000000} \frac{-1^x}{2x+1} = 3.141595986912015488462612519$
$\sum_{x=0}^{10000000} \frac{-1^x}{2x+1} = 3.141592986923015460712643380$

A good breakpoint is marked with an horizontal rule: a black stripe means that good points are very tight there.

```
require("leptonica")
function document.lscarsolua_pixGetPixel(pixs,x,y)
    local scratch = 0
    local res = ''
    scratch = leptonica.uti_getref_l_uint32()
    if (leptonica.pixGetPixel(pixs,x,y,scratch) == 0) then
        res = leptonica.uti_valref_l_uint32(scratch)
    else
        print("! Error on ",x,y)
        res = ''
    end
    return res
end

function document.lscarsol.ept_get_breaks(filename,res)
    -- os.execute([[pdftoppm -mono filename]])
    local mainName = filename
    local pixs = leptonica.pixRead(mainName)
    local lua_pixGetPixel = document.lscarsolua_pixGetPixel
    local GoodPoint = {} -- document.lscarsol.GoodPoints or {}
    if pixs then
        local w = leptonica.pixGetWidth(pixs)
        local h = leptonica.pixGetHeight(pixs)
        local d = leptonica.pixGetDepth(pixs)
        local out = ''
        local bit
        local storey = true
        local go
        GoodPoint[1] = 0
        for y=0,h-1 do
            out = ''
            storey = true
            for x=0,w-1 do
                bit = lua_pixGetPixel(pixs,x,y)
                if bit == 1 then
                    storey = false
                    break
                end
            end
            if storey and (math.floor(0.5+ (y/res) *72.27*2^16)
                ~= GoodPoint[#GoodPoint]) then
```

```

        GoodPoint[#GoodPoint +1] = math.floor(0.5+ (y/res)
*72.27*2^16)
        end
        storey = true
    end
    GoodPoint[#GoodPoint +1] = math.floor(0.5+ (h/res) *72.27*2^16)
else
    tex.sprint("ERROR")
end
return GoodPoint
end

function document.lscarso.LuaCollect(Fig,H,W,FigOpt)
    local H = math.floor(string.gsub(H,"pt","")*2^16)
    local W = math.floor(string.gsub(W,"pt","")*2^16)
    local vh = 0
    local h = step
    local S = ""
    local FigOpt = FigOpt or ""
    local NR = 0
    local Payload = ""
    local GoodPoint = document.lscarso.GoodPoints or {}
    local j = 1
    local Vboxes = {}
    local prevvh = 0
    local truey = GoodPoint[#GoodPoint]
    local y_ratio = H / truey
    for i,v in ipairs(GoodPoint) do
        if GoodPoint[i+1] == nil then
            break
        end
        NR = NR +1
        Vboxes[#Vboxes+1] = {from=v,to=GoodPoint[i+1],mark=NR}
    end
    Payload = "\\ruledvbox{\\hbox{\\clip[voffset=\\%dsp,width=\\%dsp,height=\\%dsp]
        {\\externalfigure[\\%s][\\%s]}}}}\\par\\nointerlineskip\\blank[1sp]"
    for i,v in ipairs(Vboxes) do
        S =string.format(Payload,y_ratio*v.from,W,y_ratio*(v.to-v.from),Fig,FigOpt)
        tex.sprint(tex.ctxcatcodes,S)
    end
end
end

```

3 MuPDF

We can replace the `pdftoppm` call by building a Lua binding to MuPDF, “a light-weight PDF viewer and toolkit written in portable C” (<http://www.mupdf.com>). Sumatra (http://en.wikipedia.org/wiki/Sumatra_PDF) is a fast pdfviewer based on MuPDF.

Despite MuPDF is a well written library and there are several examples that explain how to use it, it's not easy to write a converter.

What follow is only a part of an almost literal translation into Lua of the C program `pdfdraw.c` and even so the debugging is difficult.

```
function drawpbm(pagenum)
    local xref = pdfdraw.targetpdf.xref
    local drawrotate = pdfdraw.drawrotate
    local drawzoom = pdfdraw.drawzoom
    local drawbands = pdfdraw.drawbands
    local drawpattern= pdfdraw.drawpattern
    drawloadpage(pagenum)
    local drawpage = pdfdraw.drawpage
    local drawcache = pdfdraw.drawcache
    local bbox,w,h, bh
    local pix
    local fd,name
    name = pdfdraw.outname
    local ctm = mupdf.fz_identity()
    ctm = mupdf.fz_concat(ctm, mupdf.fz_translate(0, - drawpage.mediabox.y1))
    ctm = mupdf.fz_concat(ctm, mupdf.fz_scale(drawzoom, - drawzoom))
    ctm = mupdf.fz_concat(ctm, mupdf.fz_rotate(drawrotate + drawpage.rotate))
    bbox = mupdf.fz_roundrect(mupdf.fz_transformrect(ctm, drawpage.mediabox))
    w = bbox.x1 - bbox.x0;
    h = bbox.y1 - bbox.y0;
    bh = h / drawbands;
    if NotNil(drawpattern) then
        fd = io.open(name,'wb')
        if (fd == nil) then
            die(fz_throw("ioerror: could not create raster file '%s'", name));
        end
        fd:write(string.format("P4\n%d %d\n", w, h));
    end
    pix = mupdf.fz_newpixmap(mupdf.pdf_devicergb, bbox.x0, bbox.y0, w, bh)
    mupdf.fz_clearpixmap(pix, 0xFF);
    for b = 0, drawbands-1 do
        local dev, error
        dev = mupdf.fz_newdrawdevice(drawcache, pix)
```

```

error = mupdf.pdf_runcontentstream(dev, ctm, xref, drawpage.resources, drawpage.contents)
if (error>0) then
    die(fz_rethrow(error, "cannot draw page %d in PDF file '%s'", pagenum, basename))
end
mupdf.fz_freedevice(dev)
if NotNil(drawpattern) then
    for y = 0, pix.h -1 do
        local column = y * pix.w * 4
        local dst = {}
        local bit = 0
        local r = 0
        for x = 0, pix.w-1 do
            local v = (1+(mupdf.uti_samples_arr_getitem(pix.samples,(x * 4 + 1) +
column)))*77
            v = v + (1+(mupdf.uti_samples_arr_getitem(pix.samples,(x * 4 + 2) + column)))*150
            v = v + (1+(mupdf.uti_samples_arr_getitem(pix.samples,(x * 4 + 3) + column)))*28
            v = math.floor(v/256)
            local d = 1
            if v > 200 then d = 0 end
            r = r + (d*2^(7-bit))
            bit = bit +1
            if bit == 8 then
                bit = 0
                dst[#dst+1] = string.char(r)
                r = 0
            end
        end
        if bit > 0 then
            dst[#dst+1] = string.char(r)
        end
        fd:write(table.concat(dst))
    end
end
pix.y = pix.y + bh;
if (pix.y + pix.h > bbox.y1) then
    pix.h = bbox.y1 - pix.y;
end
end
mupdf.fz_droppixmap(pix)
if NotNil(drawpattern) then
    fd:close()
end
drawfreepage()
print()
end

```

The complete binding is truly more difficult to debug: even if SWIG does an excellent work, it's necessary to manage the details of implementation of the library. In this case `pdftopbm` does already an excellent work, it's well tested, stable and ready to use; the call of an external program is not so expensive compared to a plug in.

4 Conclusion

Lua code is much nearer to plain standard Lua: ConT_EXt offers some shortcuts that is better to learn. For example `math.floor(string.gsub("10pt", "pt", "")*2^16)` can be replaced by `string.todimen("10pt")` and possibly other things related to print.

The algorithm that finds a good breakpoint is rather simple: for example the are some unwanted good breakpoints between a Σ and the its subscript. This leads to break an object that should not be broken even if it has some white rows inside. A practical solution is to consider a line with thickness grater than 1 pixel (but it's better to use metric dimensions).

These ideas are also valid for scanned text but then we must take care of noise (and Leptonica can help a lot here).

Finally, it's better to consider carefully the opportunity of a binding. As shown with MuPFD, sometimes the traditional way is still the best way.

About μ Talk

μ Talk is a short and technical paper that shows some unusual, hopefully useful, ideas following the schema "figure $\rightarrow$ code". The main topic is always typographic programming in ConT_EXt & Lua.