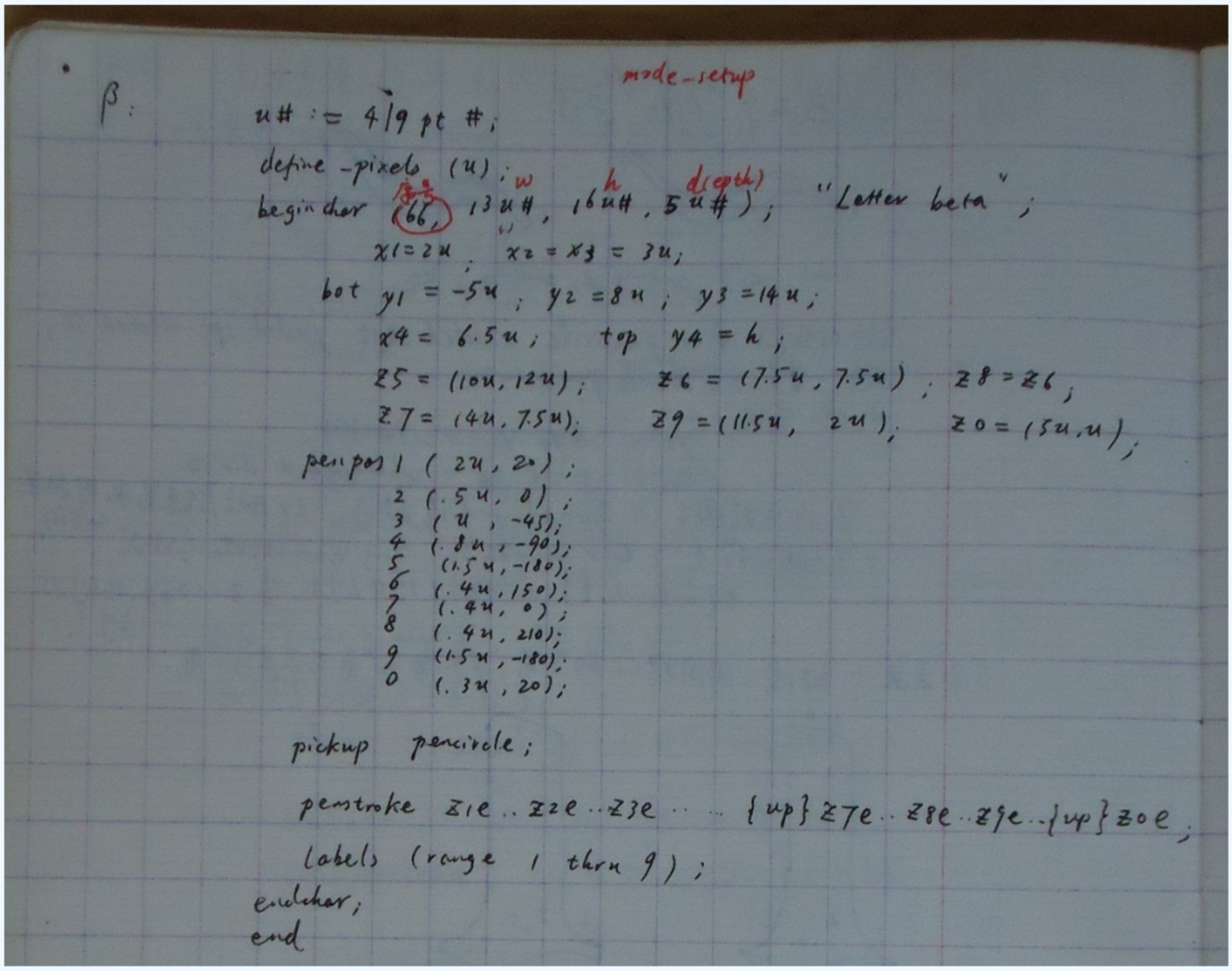
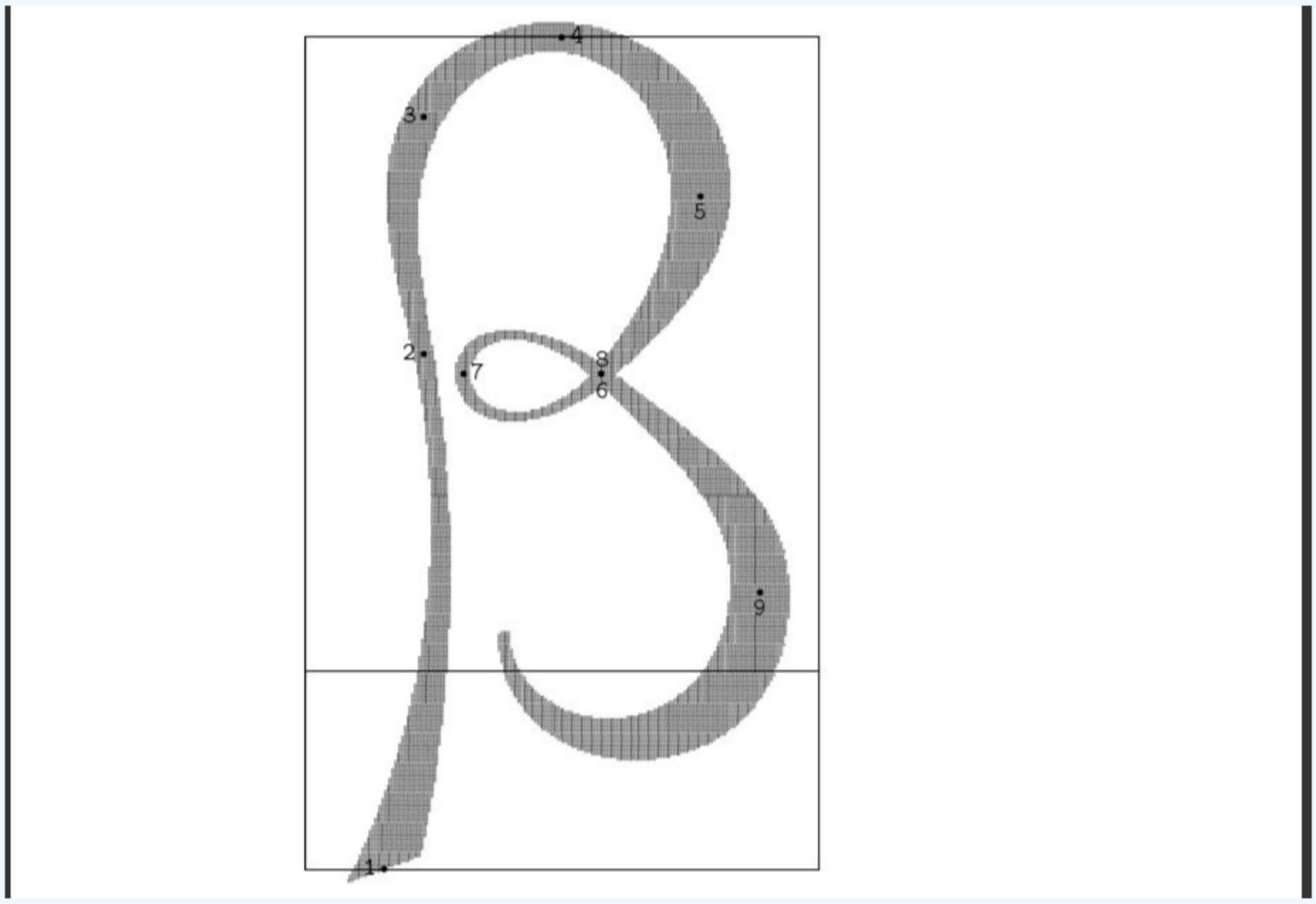
本站Web首页的制作：使用METAFONT

首页链接到中文版和英文版，为了显示各自的特点，指示中文版处使用中文汉字书法图片作背景，指示英文版处又想以一定的字体风格来 显示“English”单词的这几个字母，让字母带点汉字隶书的风格。所以制作首页时的主要工作，就是“English”的几个字母的字体设计。 这时就用上METAFONT了，边学边做。 从Christophe Grandsire的The METAFONT tutorial(mftut.pdf) （wikipedia Metafont词条中有链接）开始，其中第一个例子是：



```
mode_setup
beta:
u# := 4/9 pt #;
define_pixels (u);                    w    h    depth
beginchar (66, 13u#, 16u#, 5u#); "Letter beta";
    x1=2u;  x2 = x3 = 3u;
bot y1 = -5u;  y2 = 8u;  y3 = 14u;
    x4 = 6.5u;     top y4 = h;
    z5 = (10u, 12u);     z6 = (7.5u, 7.5u);  z8 = z6;
    z7 = (4u, 7.5u);     z9 = (11.5u, 2u);   z0 = (5u, u);
penpos1 (2u, 20);
       2 (.5u, 0);
       3 (u, -45);
       4 (.8u, -90);
       5 (1.5u, -180);
       6 (.4u, 150);
       7 (.4u, 0);
       8 (.4u, 210);
       9 (1.5u, -180);
       0 (.3u, 20);
pickup pencircle;
penstroke z1e .. z2e .. z3e .. .. {up}z7e .. z8e .. z9e .. {up}z0e;
    labels (range 1 thru 9);
endchar;
end
```

这生成一个字母beta：



陆陆续续了解到一些更多的命令，觉得可能能用到，罗列如下：

draw ?

whatever [z1, z2].

dir 90 angle vector

shifted (x, y) scaled rotated around reflected about

pickup pencircle xscaled 10 yscaled 35 rotated (45)

pickup pencircle scaled 15 xscaled Phi rotated angle (Phi, 1);

draw z1{right} .. z2{up} ..{left} z3;

fill

filldraw

drawdot

最能体现隶书风格的笔画在字母E和g上。但套用最多的模式是笔划起始终止处的衬线，所以对衬线的画法做了较详细的了解。摘抄翻译了一些 The METAFONTbook中讲到衬线画法的内容。
书中先是在p152讲到一处画衬线。然后在p162开始较详细地讲了画衬线的方法。
p152处：

p152. 先写波浪号"tilde"字符.

draw z1 .. controls z2 and z3 .. z4

左边的例子用的 pencircle xscaled .8pt yscaled .2pt rotated 50,
右边一样，只是用的是 pensquare.

z2, z3 定义：

$y_2 - y_1 = y_4 - y_3 = 3(y_4 - y_1)$;
$z_2 - z_1 = z_4 - z_3 = whatever * dir 50$

第一个方程用了一个老的美国书法技法，即开始/结束笔画用执笔的方向；
第二个方程是一个数学上的技法，基于 Bernshtein 多项式
t[0,3, -2,1] 从0到1到0到1 当t从0到.25到.75到1.

下面来画一个衬线，用同样的两支笔，只是用 20° 角而不是 50° 角.

可用 'filldraw' 命令来画：

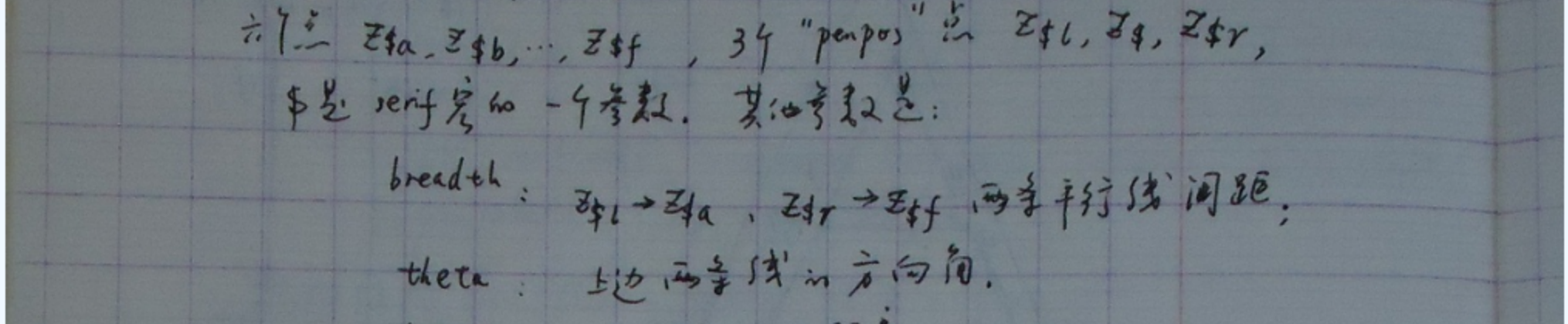
filldraw z1 .. controls z2 .. z3
--(flex(z3, .5[z3,z4] + dishing, z4)) shifted (0, -epsilon)
-- z4 .. controls z5 .. z6 --cycle.

flex(z1,z2,z3)
表示路径
z1 .. z2{z3-z1} .. z3
flex(z1,z2,z3,z4)
表示
z1 .. z2{z4-z1}
.. z3{z4-z1} .. z4

dishing 参数在 z3 和 z4 之间产生一个微小的升起，flex被用epsilon降低来避免"strange paths"的危险，否则可能会被导致在z3或z4的"tiny loops". 但该例子最有趣的是用了双控制点，z2和z5.（由于'controls z2'等同于controls z2 and z2'.）. 这些点如下决定：

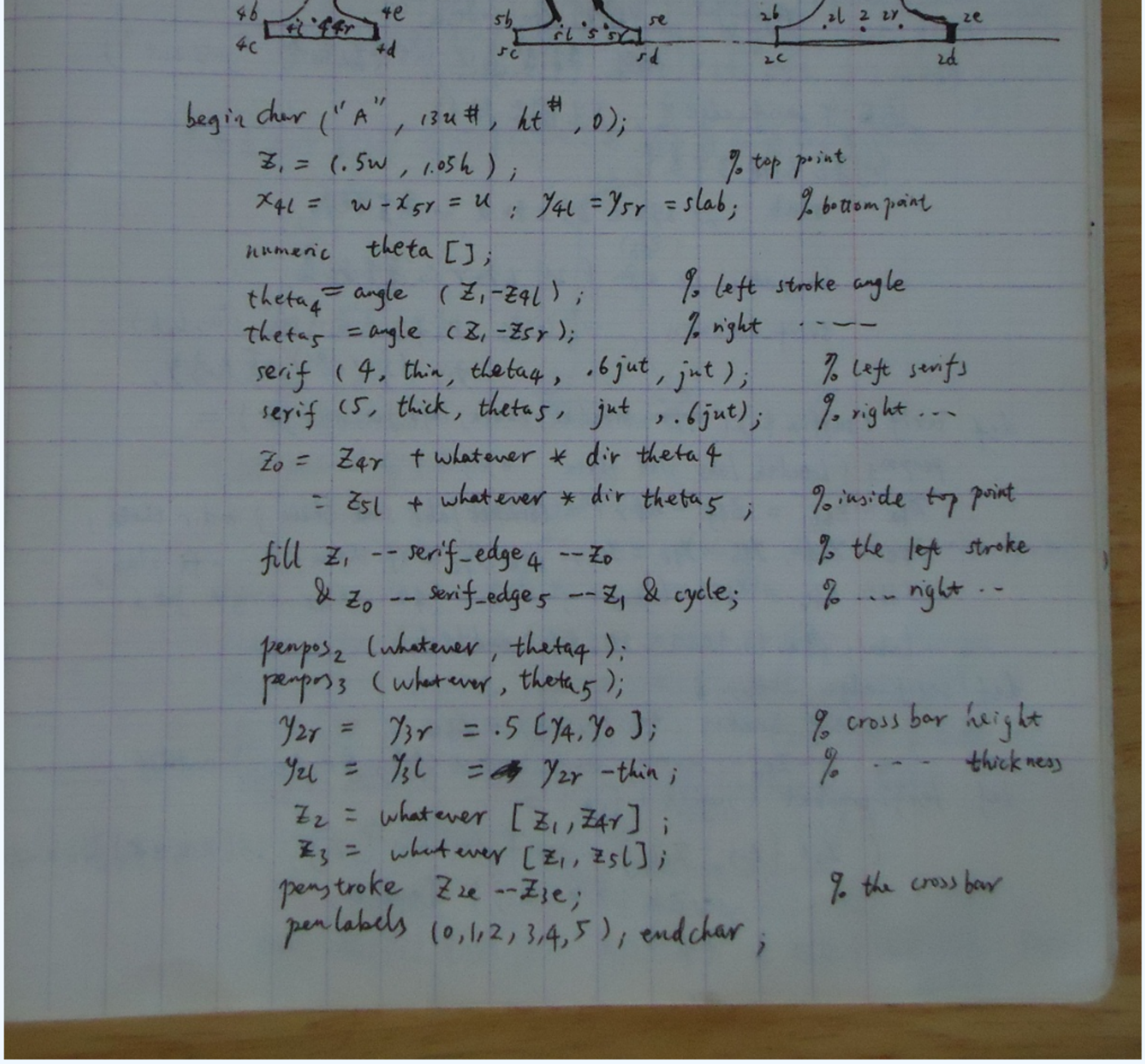
$x_2 = x_1$; $z_2 = z_3 + whatever * dir 20$,

再看p162处：
书中先后讲述了两种不同方法。
插图18b给出了一种待画衬线的形状（使用第一种方法画的）：



然后开始先后讲述：

用上面的例程，作者画了带该种衬线的A和I：



pen labels (1,2); endchar;

插图使用 cm thin = .5pt , thick = 1.1pt ;
u = .6pt , ht = 7pt;
slab = .25pt , jut = .9pt ;
bracket = pt , serif_darkness = $\frac{1}{3}$.

第二种方法 第2种方法基于 Ch16 末尾 in 例2 (即 P152 例)前)

在此假设 broad_pen 是一个 'pensquare xscaled px yscaled py rotated phi'
(px>py, phi是小角) 填充一个较大区域时会用这支笔制作
较粗(浓厚)笔画; serif 例程被给出 z_{4l} 和 z_{4r} 间距 xx.

有一个 pair 变量 dishing 控制 z_{4c} 和 z_{4d} 间 so curvature.

上、下衬线是类似的，但最好把它们写成分开的宏.

```
def bot_serif (suffix $) (expr xx, theta, left_jut, right_jut) =
  penpos$ (xx,0); z$a - z$l = z$f - z$r = (bracket/abs sind theta)*dirtheta;
  y$c = top y$l; y$d = y$r; x$c = x$l - left_jut; x$d = x$r + right_jut;
  z$b = z$l + whatever * dir theta = z$c + whatever * dir phi;
  z$e = z$r + ~~~~~~ = z$d + ~~~~~ - phi;
  labels ($a, $b, $c, $d, $e, $f ) enddef;
def bot_serif_edge suffix $ =
  ( z$a .. controls z$b .. z$c
    -- (flex (z$c, .5[z$c, z$d] + dishing, z$d)) shifted (0, -epsilon)
    -- z$d .. controls z$e .. z$f ) enddef;
```

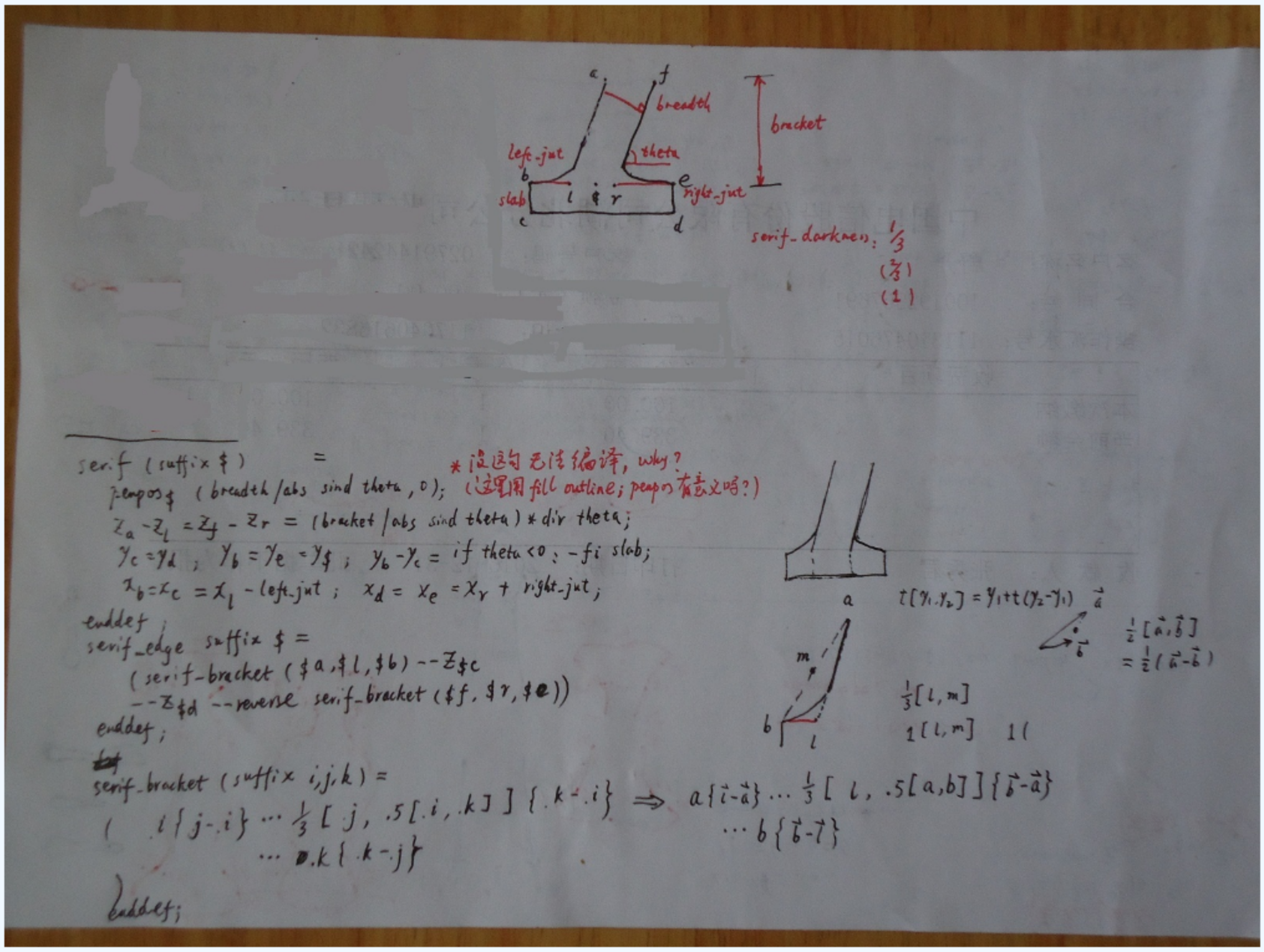
并接着开始讲第二种方法的例程（如上图下半部分），然后演示了用该例程画的又一种衬线的A和I：

```
z0 = z4r + whatever * dir theta4 = z5l + whatever * dir theta5;
fill draw z1 --bot_serif_edge4 --z0 & z0 -- bot_serif_edge5
    --z1 & cycle;
top y2 = top y3 = .45 bot y0; z2 = whatever[z1, z4r];
z3 = whatever[z1, z5l];
draw z2 --z3; penlabels(0,1,2,3,4,5); endchar;
beginchar("I", 6u#, ht#, 0); pickup broad_pen;
    x1 = x2 = .5w; y1 = h; y2 = 0;
    top_serif (1, xx, -90, 1.1jut, 1.1jut);
    bot_serif (2, xx, 90, 1.1jut, 1.1jut);
    fill draw bot_serif_edge2 --reverse top_serif_edge1 --cycle;
    penlabels(1,2); endchar;
```

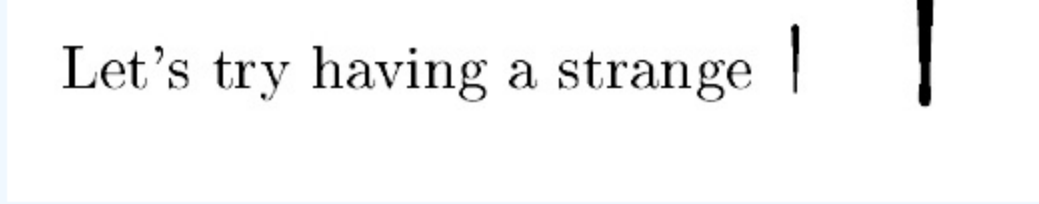
```
採用中, px = .8pt   py = .2pt   phi = 20   xx = .3pt
        u = .6pt    ht = 7pt    jut = .9pt  bracket = pt
dishing = (.25pt, 0) rotated 20.
```

其中说到第二种方法基于第16章末尾的例子，指的就是p152处。

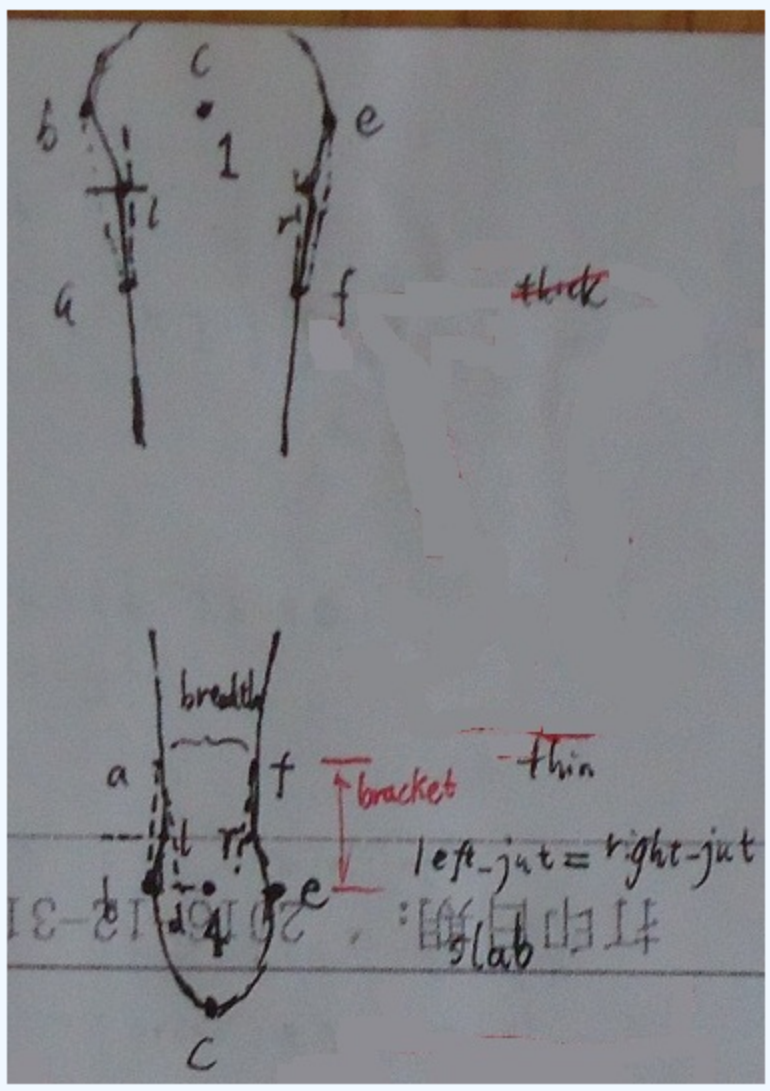
第一种方法是勾勒出轮廓线然后填充（使用fill命令来画），比较易于我理解，所以我用了第一种方法来画衬线。沿用原书中的宏和代码，作了一点改动即可用了。
先对作者原书中的例程作了一些理解：

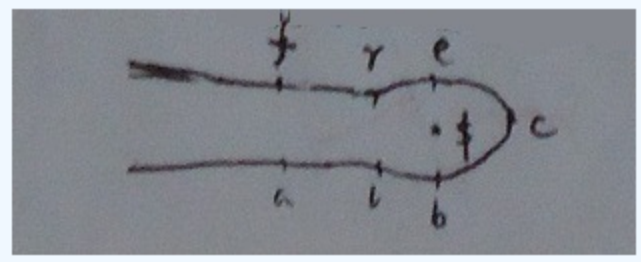


然后我大致想画成这样：



即起止处的笔画稍微粗一些。上面是竖，横也一样。勾勒用点的设置草图：





比原书中点的设置就少一个d点。
实现竖的最好例子莫过于字母l了：

tmp5\lowerL.mf:

```
u#:=.6pt#;
thin#:=.5pt#;
thick#:=1.1pt#;
ht#:=7pt#;
%slab#:=.25pt#;
slab#:=.8pt#;
%jut#:=.9pt#;
jut#:=.3pt#;
bracket#:=pt#;
define_pixels(u, ht, slab, jut, bracket);
define_blacker_pixels(thin, thick);

def serif_my(suffix $)(expr breadth, breadth_out, theta, left_jut, right_jut) =
 %penpos$(breadth / abs sind theta, 0);
  x$a = x$ - 0.5breadth;
  x$f = x$ + 0.5breadth;
  y$a = y$f = y$+bracket*(sind theta);

  x$b = x$ - left_jut ;
  x$e = x$ + right_jut;
  y$b = y$e = y$;

  x$c = x$;
  y$c = y$ - slab * sind theta;

  y$l = 0.4[y$b, y$a];
  y$r = 0.4[y$e, y$f];

  z$m = 0.4[z$b, z$a];
  y$n - y$b = 0.4(y$a - y$b);
  x$n-x$a = (0.5*(breadth_out-breadth))*(0.6*(y$a-y$b)) / (y1-y2);

  z$l = 0.5[z$m, z$n];
  x$r = x$ + x$ - x$l;
labels($a,$b,$c,$e,$f,$l,$r, $m, $n)
enddef;


def serif_edge_my suffix $ =
  (z$a{z$l-z$a}..z$l{z$b-z$l}...z$b..z$c..z$e{z$r-z$e}...z$r{z$f-z$r}..z$f)
enddef;



beginchar("l",13u#,16u#,0);"Letter l";

x1=x2=.5w;
y1=h;y2=0;

serif_my(1, thick,thin,  -90, 2.1jut, 2.1jut);
serif_my(2, thin, thick,  90, 0.9jut, 0.9jut);

fill serif_edge_my2 -- reverse serif_edge_my1 -- cycle;

penlabels(1,2);
endchar;
end
```

多增了一个参数breadth_out是因为笔画两端粗细不一样，用来表示对端的宽度（是thick还是thin）。

对于横，我发现字母调用宏例程的代码基本上把角度改一下就可以了，宏中则把x坐标改为相应的y坐标。字母r被设计成下面的竖加上面（点）实现为一

短横， 比较字母l正好多一短横，拿来作例子，代码如下：

tmp5\lowerI.mf:

```
u#:=.6pt#;
thin#:=.5pt#;
thick#:=1.1pt#;
ht#:=7pt#;
%slab#:=.25pt#;
slab#:=.8pt#;
%jut#:=.9pt#;
jut#:=.3pt#;
bracket#:=pt#;
define_pixels(u, ht, slab, jut, bracket);
define_blacker_pixels(thin, thick);

def serif_my(suffix $)(expr breadth, breadth_out, theta, left_jut, right_jut) =
 %penpos$(breadth / abs sind theta, 0);
  x$a = x$ - 0.5breadth;
  x$f = x$ + 0.5breadth;
  y$a = y$f = y$+bracket*(sind theta);

  x$b = x$ - left_jut ;
  x$e = x$ + right_jut;
  y$b = y$e = y$;

  x$c = x$;
  y$c = y$ - slab * sind theta;

  y$l = 0.4[y$b, y$a];
  y$r = 0.4[y$e, y$f];

  z$m = 0.4[z$b, z$a];
  y$n - y$b = 0.4(y$a - y$b);
  x$n-x$a = (0.5*(breadth_out-breadth))*(0.6*(y$a-y$b)) / (y1-y2);

  z$l = 0.5[z$m, z$n];
  x$r = x$ + x$ - x$l;
labels($a,$b,$c,$e,$f,$l,$r, $m, $n)
enddef;

def serif_my_horizontal(suffix $)(expr breadth, breadth_out, theta, left_jut, right_jut) =
 %penpos$(breadth / abs sind theta, 0);
  y$a = y$ - 0.5breadth;
  y$f = y$ + 0.5breadth;
  x$a = x$f = x$+bracket*(sind theta);

  y$b = y$ - left_jut ;
  y$e = y$ + right_jut;
  x$b = x$e = x$;

  y$c = y$;
  x$c = x$ - slab * sind theta;

  x$l = 0.4[x$b, x$a];
  x$r = 0.4[x$e, x$f];

  z$m = 0.4[z$b, z$a];
  x$n - x$b = 0.4(x$a - x$b);
  y$n-y$a = (0.5*(breadth_out-breadth))*(0.6*(x$a-x$b)) / (x4-x5);

  z$l = 0.5[z$m, z$n];
  y$r = y$ + y$ - y$l;
labels($a,$b,$c,$e,$f,$l,$r, $m, $n)
enddef;

def serif_edge_my suffix $ =
  (z$a{z$l-z$a}..z$l{z$b-z$l}...z$b..z$c..z$e{z$r-z$e}...z$r{z$f-z$r}..z$f)
enddef;



beginchar("i",13u#,16u#,0);"Letter i";

x1=x2=.5w;
y1=h/2;y2=0;


serif_my(1, thick,thin,  -90, 2.1jut, 2.1jut);
serif_my(2, thin, thick,  90, 0.9jut, 0.9jut);
fill serif_edge_my2 -- reverse serif_edge_my1 -- cycle;


z3 = z1 + (0, 0.2h);
x4 = x3 - 0.2w; x5 = x3+0.2w;
y4 = y5 = y3 ;
serif_my_horizontal(4, 0.7thick,thin,  90, 2.1jut, 2.1jut);
serif_my_horizontal(5, thin, 0.7thick, -90, 0.9jut, 0.9jut);
fill serif_edge_my5 -- reverse serif_edge_my4 -- cycle;


penlabels(1,2,3,4,5);
endchar;
end
```

其中可见serif_my宏完全没有改变，serif_my_horizontal宏相应于serif_my宏的改变则如下图：

```
def serif_my(suffix $)(expr breadth, breadth_out, theta, left_jut, right_jut) =
 %penpos$(breadth / abs sind theta, 0);
  x$a = x$ - 0.5breadth;
  x$f = x$ + 0.5breadth;
  y$a = y$f = y$+bracket*(sind theta);

  x$b = x$ - left_jut ;
  x$e = x$ + right_jut;
  y$b = y$e = y$;

  x$c = x$;
  y$c = y$ - slab * sind theta;

  y$l = 0.4[y$b, y$a];
  y$r = 0.4[y$e, y$f];
```

```
def serif_my_horizontal(suffix $)(expr breadth, breadth_out, theta, left_jut, ri
 %penpos$(breadth / abs sind theta, 0);
  y$a = y$ - 0.5breadth;
  y$f = y$ + 0.5breadth;
  x$a = x$f = x$+bracket*(sind theta);

  y$b = y$ - left_jut ;
  y$e = y$ + right_jut;
  x$b = x$e = x$;

  y$c = y$;
  x$c = x$ - slab * sind theta;

  x$l = 0.4[x$b, x$a];
  x$r = 0.4[x$e, x$f];
```

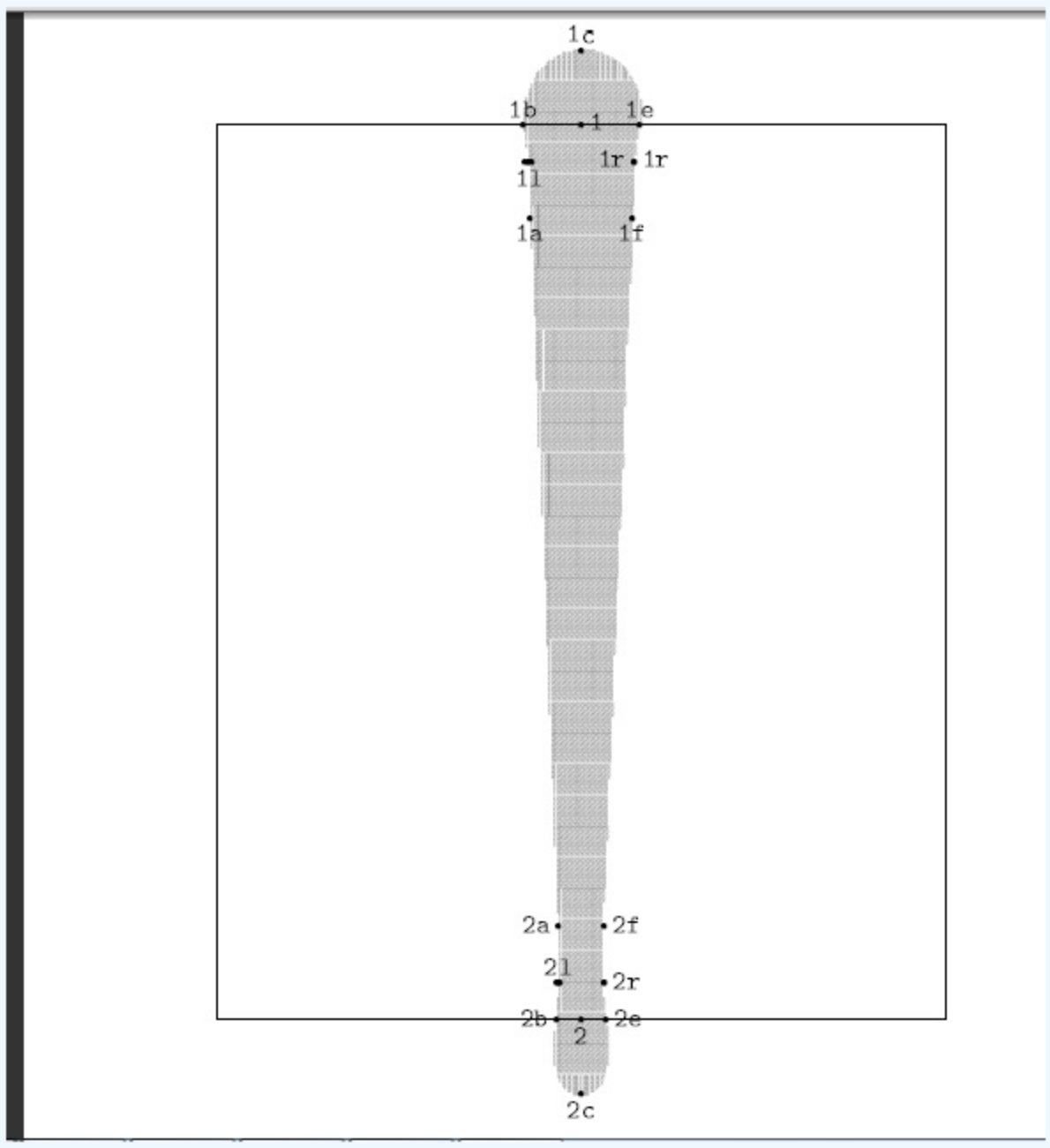
```
  z$m = 0.4[z$b, z$a];
  y$n - y$b = 0.4(y$a - y$b);
  x$n-x$a = (0.5*(breadth_out-breadth))*(0.6*(y$a-y$b)) / (y1-y2);

  z$l = 0.5[z$m, z$n];
  x$r = x$ + x$ - x$l;
labels($a,$b,$c,$e,$f,$l,$r, $m, $n)
enddef;


def serif_edge_my_suffix $ =
```

默认文本

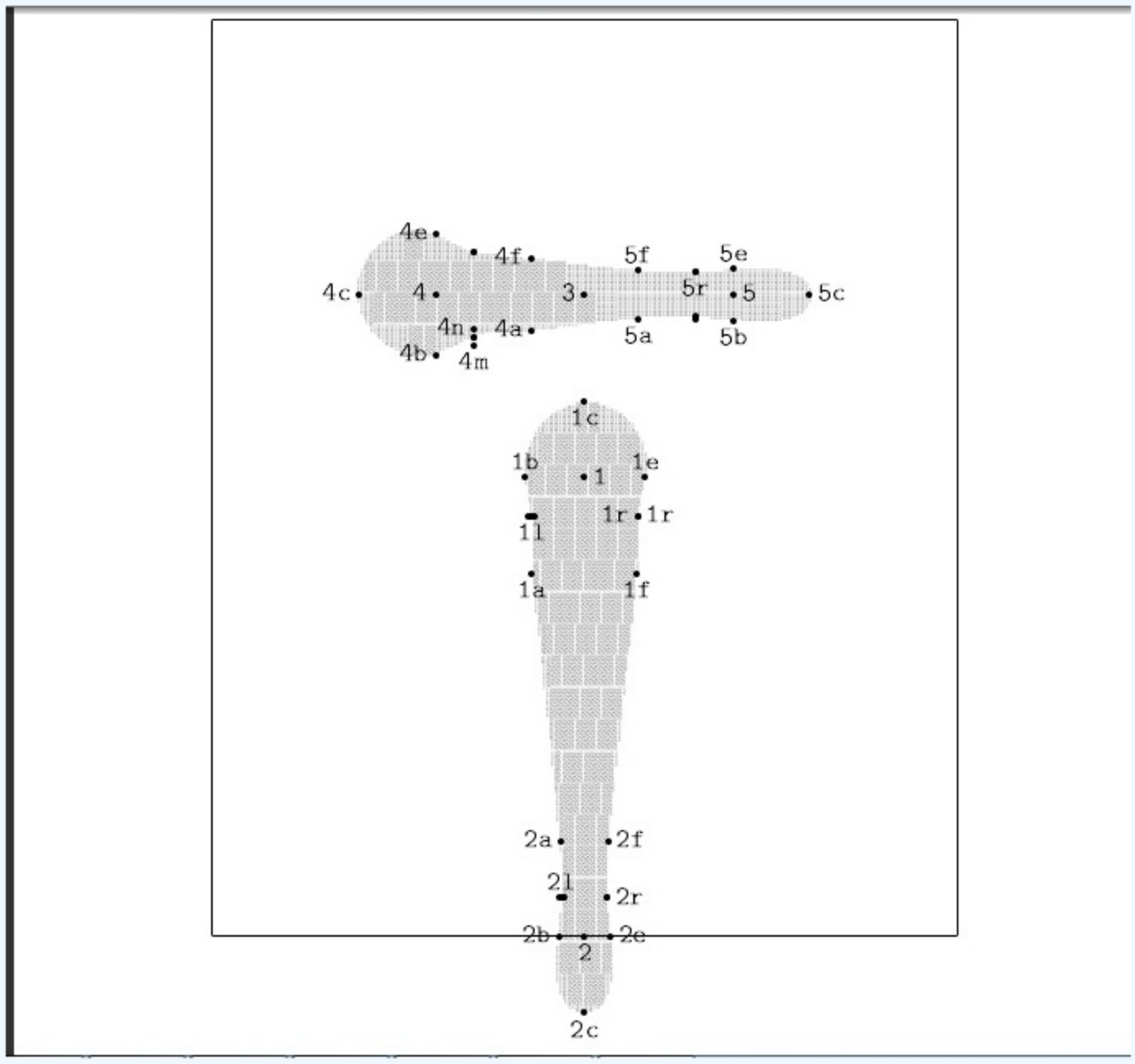
```
  z$m = 0.4[z$b, z$a];
  x$n - x$b = 0.4(x$a - x$b);
  y$n-y$a = (0.5*(breadth_out-breadth))*(0.6*(x$a-x$b)) / (x4-x5);

  z$l = 0.5[z$m, z$n];
  y$r = y$ + y$ - y$l;
labels($a,$b,$c,$e,$f,$l,$r, $m, $n)
enddef;


def serif_edge_my_suffix $ =
```

1: 1 默认文本

实际实现l和i的proof mode的图分别为



和



操作过程用到的命令一般如下：

```
mf '\mode=ljfour;mode_setup;input xx.mf'
gftopk xx.600gf xx.600pk

mf '\mode=ljfour; mag=magstep(7); mode_setup;input xx.mf'
mv xx.tfm xx2150.tfm
gftopk xx2150.2150gf xx2150.2150pk
ln -s  xx2150.2150pk xx2150.600pk

gftodvi xx.2602gf
```

```
xdvi xx.dvi -mfmode ljfour:600

latex xx.tex

(dvipdf  xx.dvi; xpdf xx.pdf)
```

为了让一个字的制作和检查方便一点，写bash脚本如下，以字母h为例：

tmp9\genH.sh:

```
#!/bin/bash

mf '\mode=ljfour; mode_setup; input liH.mf'
mv -f liH.tfm liH600.tfm
mv -f liH.600gf liH600.600gf
gftopk liH600.600gf liH600.600pk

mf '\mode=ljfour; mag=magstep(7);mode_setup; input liH.mf'
mv -f liH.tfm liH2150.tfm
mv -f liH.2150gf liH2150.2150gf
gftopk liH2150.2150gf liH2150.2150pk
rm -f liH2150.600pk
ln -s liH2150.2150pk  liH2150.600pk

latex liH_2.tex
```

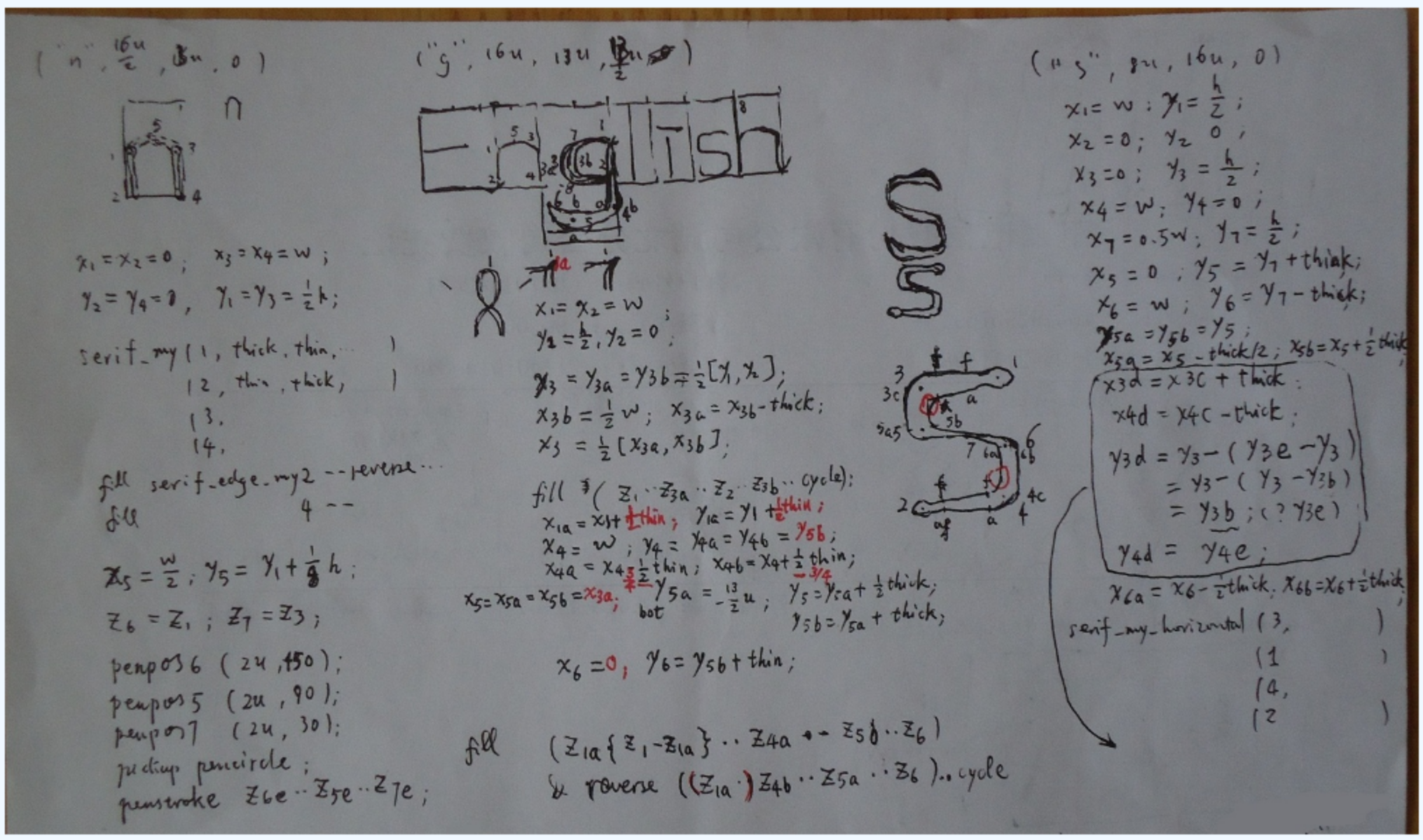
最后的liH_2.tex是检查生成的大小两个字形的tex文件，字母h的如下：

tmp9\liH_2.tex:

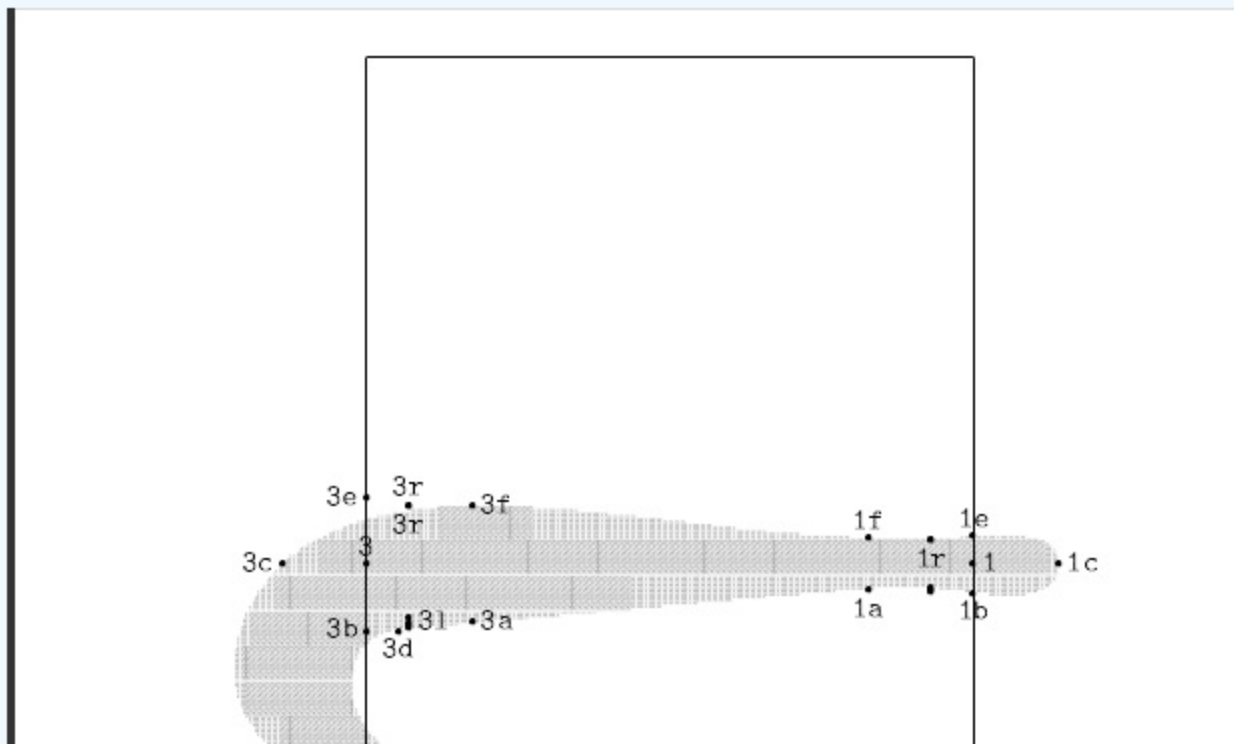
```
\documentclass{article}

\newfont{\letterliH}{liH600}
\newcommand{\otherH}{{\letterliH liH600}}
\newfont{\letterliHbig}{liH2150}
\newcommand{\otherHbig}{{\letterliHbig liH2150}}

\begin{document}

Let's try having a strange \otherH\ \otherHbig\

\end{document}
```

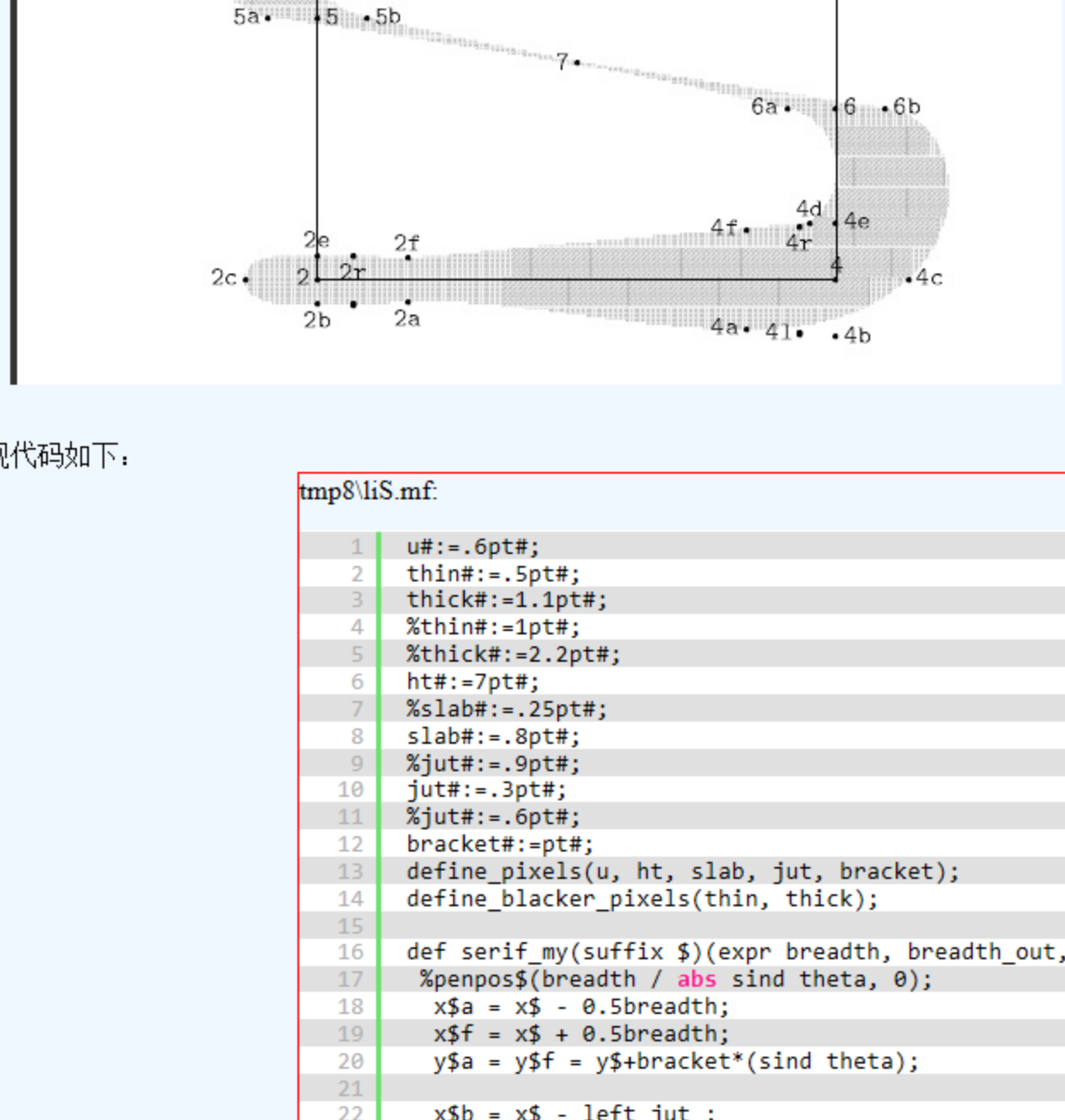
因文件中用到latex的东西，需用latex处理，仅“tex liH_2.tex”不行。

其他几个字母E、n、h、s都用到了上面衬线画法的例程。其中只有s的复杂一点，其字形勾勒点设置及编程草图如下图右边部分：



字母s实际实现的proof mode的图为





实现代码如下：

tmp8\liS.mf:

```
u#:=.6pt#;
thin#:=.5pt#;
thick#:=1.1pt#;
%thin#:=1pt#;
%thick#:=2.2pt#;
ht#:=7pt#;
%slab#:=.25pt#;
slab#:=.8pt#;
%jut#:=.9pt#;
jut#:=.3pt#;
%jut#:=.6pt#;
bracket#:=pt#;
define_pixels(u, ht, slab, jut, bracket);
define_blacker_pixels(thin, thick);

def serif_my(suffix $)(expr breadth, breadth_out, theta, left_jut, right_jut) =
 %penpos$(breadth / abs sind theta, 0);
  x$a = x$ - 0.5breadth;
  x$f = x$ + 0.5breadth;
  y$a = y$f = y$+bracket*(sind theta);

  x$b = x$ - left_jut ;
  x$e = x$ + right_jut;
  y$b = y$e = y$;

  x$c = x$;
  y$c = y$ - slab * sind theta;

  y$l = 0.4[y$b, y$a];
  y$r = 0.4[y$e, y$f];

  z$m = 0.4[z$b, z$a];
  y$n - y$b = 0.4(y$a - y$b);
  x$n-x$a = (0.5*(breadth_out-breadth))*(0.6*(y$a-y$b)) / (y1-y2);

  z$l = 0.5[z$m, z$n];
  x$r = x$ + x$ - x$l;
labels($a,$b,$c,$e,$f,$l,$r, $m, $n)
enddef;

def serif_my_horizontal(suffix $)(expr breadth, breadth_out, theta, left_jut, right_jut) =
 %penpos$(breadth / abs sind theta, 0);
  y$a = y$ - 0.5breadth;
  y$f = y$ + 0.5breadth;
  x$a = x$f = x$+bracket*(sind theta);

  y$b = y$ - left_jut ;
  y$e = y$ + right_jut;
  x$b = x$e = x$;

  y$c = y$;
  x$c = x$ - slab * sind theta;

  x$l = 0.4[x$b, x$a];
  x$r = 0.4[x$e, x$f];

  z$m = 0.4[z$b, z$a];
  x$n - x$b = 0.4(x$a - x$b);
  y$n-y$a = (0.5*(breadth_out-breadth))*(0.6*(x$a-x$b)) / (x2-x1);

  z$l = 0.5[z$m, z$n];
  y$r = y$ + y$ - y$l;
labels($a,$b,$c,$e,$f,$l,$r, $m, $n)
enddef;


def serif_edge_my suffix $ =
  (z$a{z$l-z$a}..z$l{z$b-z$l}...z$b..z$c..z$e{z$r-z$e}...z$r{z$f-z$r}..z$f)
enddef;


beginchar("S",9.6u#,16u#,0);"Letter S";

x1=w; y1=0.5h;
x2=0; y2= 0;
x3=0; y3=0.5h;
x4=w; y4= 0;
x7=0.5w; y7=0.5[y1,y2];
x5=0; y5=y7+thin;
x6=w; y6=y7-thin;
y5a=y5b=y5;
y6a=y6b=y6;
x5a=x5-0.5thick;
x5b=x5+0.5thick;
x6a=x6-0.5thick;
x6b=x6+0.5thick;

serif_my_horizontal(3, thick,thin,   90, 2.1jut, 2.1jut);
serif_my_horizontal(1, thin, thick, -90, 0.9jut, 0.9jut);
serif_my_horizontal(4, thick,thin,   -90, 2.1jut, 2.1jut);
serif_my_horizontal(2, thin, thick,   90, 0.9jut, 0.9jut);

x3d=x3c + thick;
```

```
94  x4d=x4c - thick;
95  y3d=y3b;
96  y4d=y4e;
97
98  fill serif_edge_my1 ..z3f..z3c..z5a..controls z7..z6a..z4d--reverse serif_edge_my2..z4a..z4c..z6b..
99  penlabels(1,2,3,3d, 4,4d,5,5a,5b,6,6a,6b,7);
100 endchar;
101 end
```

代码可见其中实际并未用到serif_my宏，这里只是从别处拷贝过来，保留未删而已。serif_my_horizontal宏基本保留未动， 只是有一处x坐标相减的两点要改，未能完全代码重用，相比上面字母i使用的serif_my_horizontal宏改动如下：

.\client_ftp\page_making\LiShu_ENGLISH_english\tmp5\lowerI.mf

.7/4/29 19:24:14 2,050 字节 其它一切 ANSI UNIX

```
def serif_my_horizontal(suffix $)(expr breadth, breadth_out, theta, l
 %penpos$(breadth / abs sind theta, 0);
  y$a = y$ - 0.5breadth;
  y$f = y$ + 0.5breadth;
  x$a = x$f = x$+bracket*(sind theta);

  y$b = y$ - left_jut ;
  y$e = y$ + right_jut;
  x$b = x$e = x$;

  y$c = y$;
  x$c = x$ - slab * sind theta;

  x$l = 0.4[x$b, x$a];
  x$r = 0.4[x$e, x$f];

  z$m = 0.4[z$b, z$a];
  x$n - x$b = 0.4(x$a - x$b);
  y$n-y$a = (0.5*(breadth_out-breadth))*(0.6*(x$a-x$b)) / (x4-x5);

  z$l = 0.5[z$m, z$n];
  y$r = y$ + y$ - y$l;
labels($a,$b,$c,$e,$f,$l,$r, $m, $n)
enddef;
```

19 默认文本

D:\...\client_ftp\page_making\LiShu_ENGLISH_english\tmp8\liS.mf

2017/4/29 22:14:06 2,337 字节 其它一切 ANSI UNIX

```
def serif_my_horizontal(suffix $)(expr breadth, breadth_out, theta, l
 %penpos$(breadth / abs sind theta, 0);
  y$a = y$ - 0.5breadth;
  y$f = y$ + 0.5breadth;
  x$a = x$f = x$+bracket*(sind theta);

  y$b = y$ - left_jut ;
  y$e = y$ + right_jut;
  x$b = x$e = x$;

  y$c = y$;
  x$c = x$ - slab * sind theta;

  x$l = 0.4[x$b, x$a];
  x$r = 0.4[x$e, x$f];

  z$m = 0.4[z$b, z$a];
  x$n - x$b = 0.4(x$a - x$b);
  y$n-y$a = (0.5*(breadth_out-breadth))*(0.6*(x$a-x$b)) / (x2-x1);

  z$l = 0.5[z$m, z$n];
  y$r = y$ + y$ - y$l;
labels($a,$b,$c,$e,$f,$l,$r, $m, $n)
enddef;
```

45: 19 默认文本

字母E中也用到serif_my_horizontal宏，相比上面字母中的该名称的宏也是进行类似的微小修改，如下所示将相减的x1、x2 调换一下：

.\client_ftp\page_making\LiShu_ENGLISH_english\tmp42\liE2.mf

.7/4/28 9:41:00 2,751 字节 其它一切 ANSI UNIX

```
def serif_my_horizontal(suffix $)(expr breadth, breadth_out, theta, l
 %penpos$(breadth / abs sind theta, 0);
  y$a = y$ - 0.5breadth;
  y$f = y$ + 0.5breadth;
  x$a = x$f = x$+bracket*(sind theta);

  y$b = y$ - left_jut ;
  y$e = y$ + right_jut;
  x$b = x$e = x$;

  y$c = y$;
  x$c = x$ - slab * sind theta;

  x$l = 0.4[x$b, x$a];
  x$r = 0.4[x$e, x$f];

  z$m = 0.4[z$b, z$a];
  x$n - x$b = 0.4(x$a - x$b);
  y$n-y$a = (0.5*(breadth_out-breadth))*(0.6*(x$a-x$b)) / (x1-x2);

  z$l = 0.5[z$m, z$n];
  y$r = y$ + y$ - y$l;
labels($a,$b,$c,$e,$f,$l,$r, $m, $n)
enddef;
```

默认文本

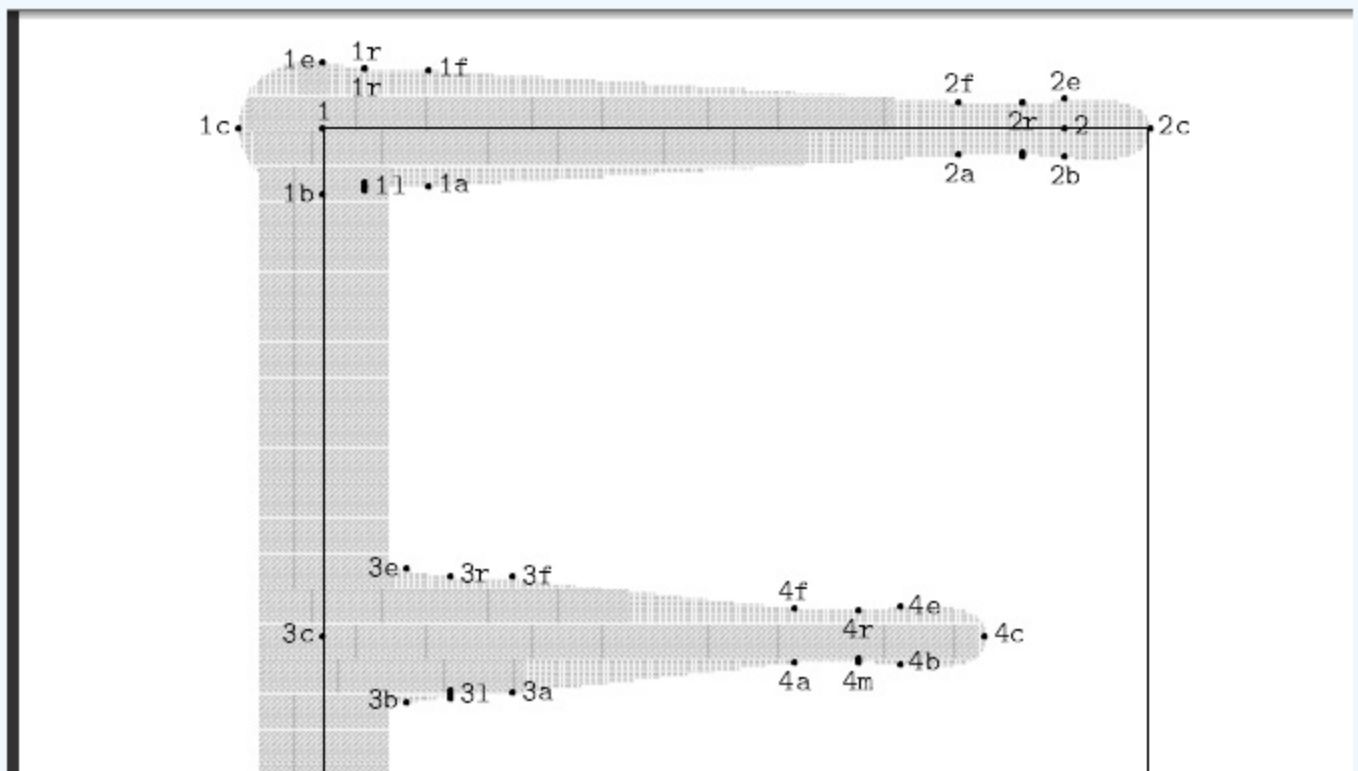
D:\...\client_ftp\page_making\LiShu_ENGLISH_english\tmp8\liS.mf

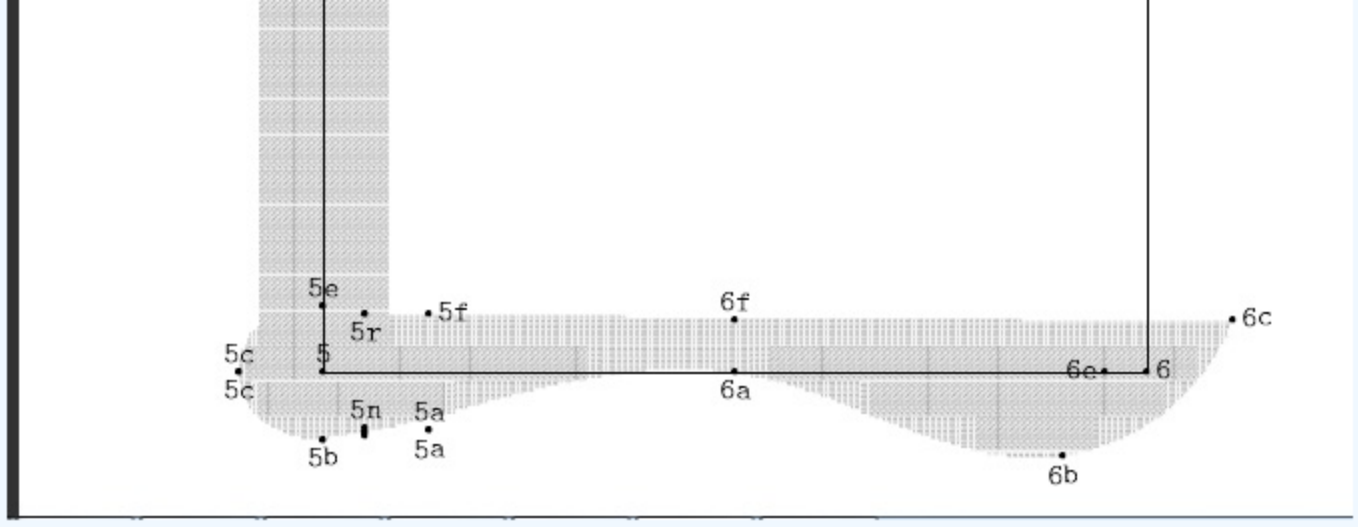
2017/4/29 22:14:06 2,337 字节 其它一切 ANSI UNIX

```
def serif_my_horizontal(suffix $)(expr breadth, breadth_out, theta, l
 %penpos$(breadth / abs sind theta, 0);
  y$a = y$ - 0.5breadth;
  y$f = y$ + 0.5breadth;
  x$a = x$f = x$+bracket*(sind theta);

  y$b = y$ - left_jut ;
  y$e = y$ + right_jut;
  x$b = x$e = x$;

  y$c = y$;
  x$c = x$ - slab * sind theta;

  x$l = 0.4[x$b, x$a];
  x$r = 0.4[x$e, x$f];

  z$m = 0.4[z$b, z$a];
  x$n - x$b = 0.4(x$a - x$b);
  y$n-y$a = (0.5*(breadth_out-breadth))*(0.6*(x$a-x$b)) / (x2-x1);

  z$l = 0.5[z$m, z$n];
  y$r = y$ + y$ - y$l;
labels($a,$b,$c,$e,$f,$l,$r, $m, $n)
enddef;
```

1: 1 默认文本

字母E实际实现的proof mode的图为





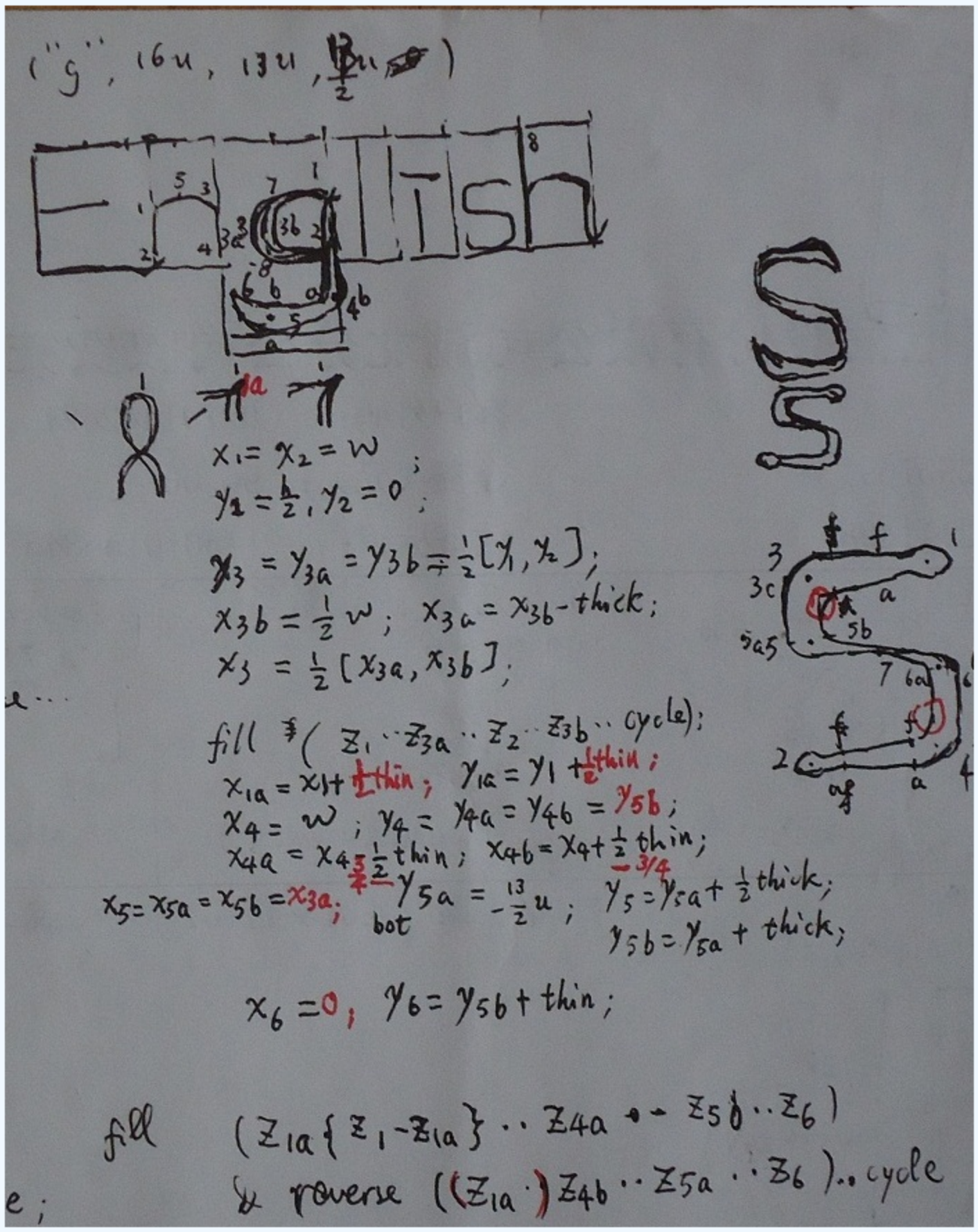
实现代码如下：

tmp42\liE2.mf:

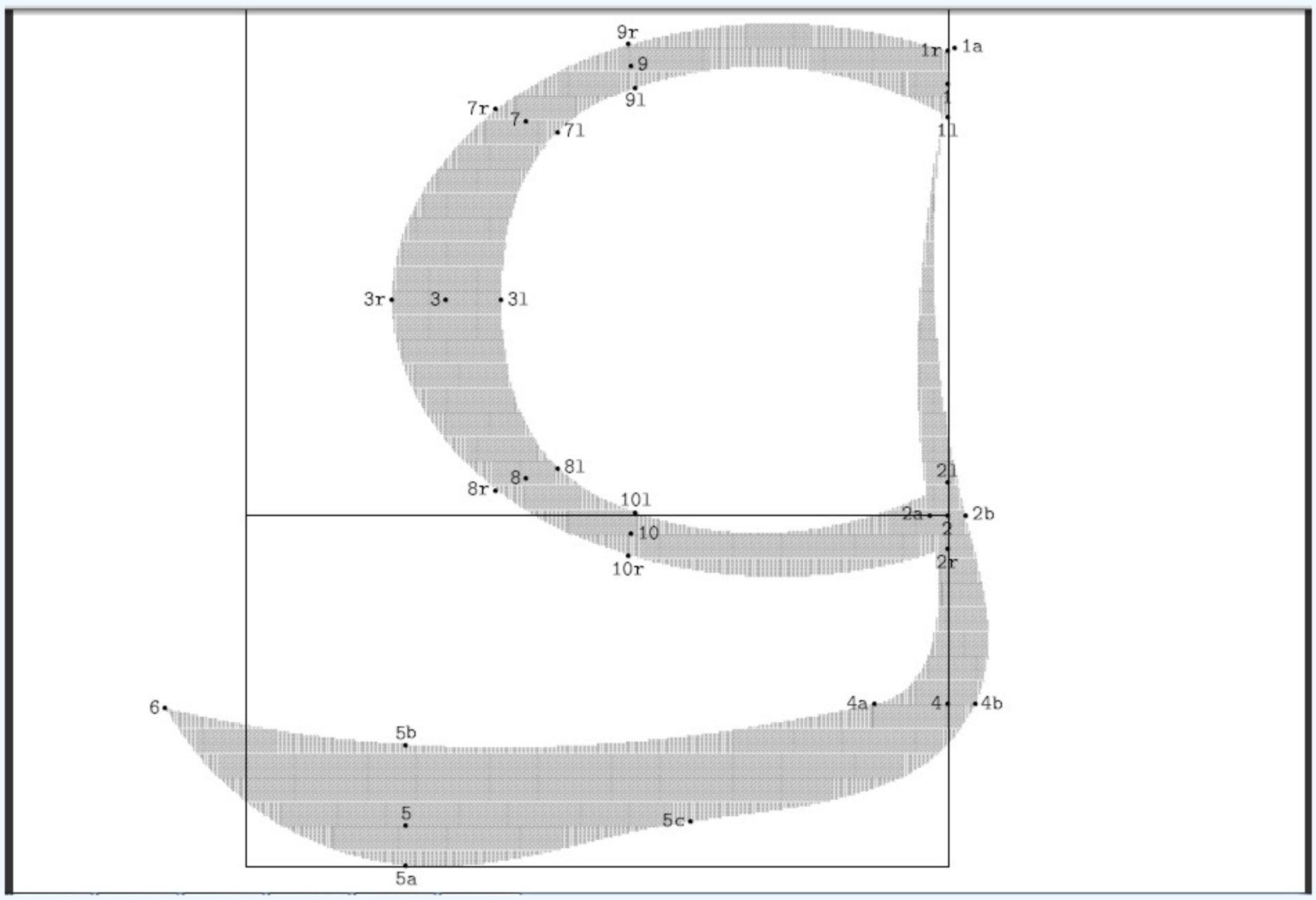
```
u#:=.6pt#;
thin#:=.5pt#;
thick#:=1.1pt#;
%thin#:=1pt#;
%thick#:=2.2pt#;
ht#:=7pt#;
%slab#:=.25pt#;
slab#:=.8pt#;
%jut#:=.9pt#;
jut#:=.3pt#;
%jut#:=.6pt#;
bracket#:=pt#;
define_pixels(u, ht, slab, jut, bracket);
define_blacker_pixels(thin, thick);

def serif_my_horizontal(suffix $)(expr breadth, breadth_out, theta, left_jut, right_jut) =
 %penpos$(breadth / abs sind theta, 0);
  y$a = y$ - 0.5breadth;
  y$f = y$ + 0.5breadth;
  x$a = x$f = x$+bracket*(sind theta);

  y$b = y$ - left_jut ;
  y$e = y$ + right_jut;
  x$b = x$e = x$;

  y$c = y$;
  x$c = x$ - slab * sind theta;

  x$l = 0.4[x$b, x$a];
  x$r = 0.4[x$e, x$f];

  z$m = 0.4[z$b, z$a];
  x$n - x$b = 0.4(x$a - x$b);
  y$n-y$a = (0.5*(breadth_out-breadth))*(0.6*(x$a-x$b)) / (x1-x2);

  z$l = 0.5[z$m, z$n];
  y$r = y$ + y$ - y$l;
labels($a,$b,$c,$e,$f,$l,$r, $m, $n)
enddef;


def serif_edge_my suffix $ =
  (z$a{z$l-z$a}..z$l{z$b-z$l}...z$b..z$c..z$e{z$r-z$e}...z$r{z$f-z$r}..z$f)
enddef;

beginchar("E",13u#,16u#,0);"Letter E";

x1=0; x2=.9w;
y1=h;y2=h;

x3=0+slab; x4=.7w;
y3=h/2;y4=h/2;

x5=0;
y5=0;

z7 = z1;
z8 = z5;

serif_my_horizontal(1, thick,thin,  90, 2.1jut, 2.1jut);
serif_my_horizontal(2, thin, thick, -90, 0.9jut, 0.9jut);
fill serif_edge_my2 -- reverse serif_edge_my1 -- cycle;

serif_my_horizontal(3, thick,thin,  90, 2.1jut, 2.1jut);
serif_my_horizontal(4, thin, thick, -90, 0.9jut, 0.9jut);
fill serif_edge_my4 -- reverse serif_edge_my3 -- cycle;

x6 = w;
y6 = 0;
x6a = x6f = (1/2)[x5,x6];
y6a = y5; y6f = y6a + thin;
x6b = x6-slab;  y6b = y5b - thin/3; %y6b = y6  - thick/2;
x6c = x6 + slab; y6c = y6 + thin;
x6e = x6 - 0.5*slab; y6e = y6;
serif_my_horizontal(5, thick,thin,  90, 2.1jut, 2.1jut);
fill reverse ( z5l{z5b-z5l}...z5b..z5c..z5e{z5r-z5e}...z5r{z5f-z5r})..z6a..z6b..z6c -- cycle;


penpos7(2u, 0);
penpos8(2u, 0);
pickup pencircle;
penstroke z7e--z8e;

penlabels(1,2,5,5a,5b,5c,5d,5e,6, 6a, 6b, 6c, 6e, 6f);
endchar;
end
```

可见在点1、2、3、4、5处都同上面一样使用了serif_my_horizontal例程。只有点6处的勾勒点是单独各个设置的， 这里就是E中最能显示隶书风格笔画的地方了。这里没有使用什么一定之规，只是不断调整设置试验然后检查，看 效果大致可以就停止改动了。

此外最能显示隶书风格笔画的地方就是字母g了。起始设计草图如下图中：



字母g实际实现的proof mode的图为



字母g的实现代码中没有用到以上衬线的例程。用的是类似前面画beta例子的笔法（penstoke），以及仍然是轮廓线填充的方法。 代码如下：

tmp7\liG.mf:

```
u#:=.6pt#;
thin#:=.5pt#;
thick#:=1.1pt#;
%thin#:=1pt#;
%thick#:=2.2pt#;
```

```
ht#:=7pt#;
%slab#:=.25pt#;
slab#:=.8pt#;
%jut#:=.9pt#;
jut#:=.3pt#;
%jut#:=.6pt#;
bracket#:=pt#;
define_pixels(u, ht, slab, jut, bracket);
define_blacker_pixels(thin, thick);


beginchar("G",16u#,16u#,8u#);"Letter G";

x1=x2=w;
y1=0.5*h + thick; y2 = 0;

y3=0.5[y1,y2];
x3=0.4w-thick;

x7=0.4w-0.5thin;y7=y1-thin;
x8=x7;  y8=y2+thin;

x9 = 0.25[x7,x1]; x10 = x9;
y9=y1+.5thin;
y10=y2-.5thin;

penpos1(1.5u, 90);
penpos9(1u, 100);
penpos7(1.6u, 160);
penpos3(2.5u,  180);
penpos8(1.6u, 200);
penpos10(1u, 260);
penpos2(1.5u, 270);
pickup pencircle;
penstroke z1e..z9e..z7e..z3e..z8e..z10e..z2e;



x1a = x1 + .2thin;    y1a = y1 + thin;
x4 = w;  x4a = x4 - 2thin;  x4b = x4 + .75thin;
bot y5a = -8u; y5 = y5a + .5thick;  y5b = y5 + thick;
y4 = y4a = y4b = y5b + .5thick;
x5 = x5a = x5b = x3  - .5thick;
z5c = 0.5[z5a, z4b] + (0, -thin);
x6 = 0-thick; y6 = y5b + thin;

x2a = x2 - .5thin; x2b = x2 + .5thin; y2a = y2b = y2;

fill (z1a{z1-z1a}..z2a..z4a{z5-z4}..z5b..z6) & reverse (z1a..z2b..z4b..z5c..z5a..z6) ..cycle;

penlabels(1,2,3, 4,5, 6, 7, 8, 9, 10, 1a, 2a,2b, 4a,4b, 5a,5b,5c);
endchar;
end
```

轮廓线勾勒点（含控制点）4、5、6的设置仍然是不断修改试验而成。

至此“English”的几个字母的字体设计基本完成，效果图就如本站首页中所见。

剩下的就是些背景图的制作和叠加了。没有用到象Photoshop或GIMP等强大的软件，也没有用在线工具软件， 用Windows画图、ACDSee、Office（PowerPoint和Word）就做成了。PowerPoint实现了背景图的透明化，ACDSee 实现了文字的倾斜、晕影特效，从The TEXbook和The METAFONTbook等书上取得原字形风格文字，中文书法 背景图来自某书法比赛的图书中作者的优秀作品，这里要表示感谢。Word制作一些常见字体的文字。叠加和拼接 等等诸多处理都用Windows画图完成。

至此首页的主体就告完成，要感谢METAFONT和TEX的强大功能，以上使用的latex和mf程序均为RedHat 9的安装版本。
源文件打包在此：LiShu_ENGLISH_english.tgz