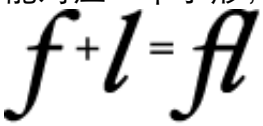
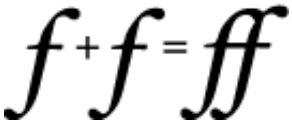
前面两篇文章介绍了文字的样式，段落样式。本文章主要介绍行模式。CTLineRef

知识了解：

1.字符（Character）和字形（Glyphs）

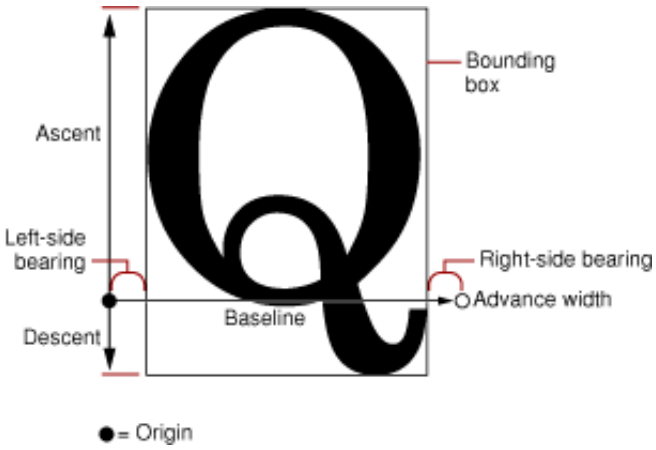
排版系统中文本显示的一个重要的过程就是字符到字形的转换，字符是信息本身的元素，而字形是字符的图形表征，字符还会有其它表征比如发音。 字符在计算机中其实就是一个编码，某个字符集中的编码，比如Unicode字符集，就囊括了大都数存在的字符。 而字形则是图形，一般都存储在字体文件中，字形也有它的编码，也就是它在字体中的索引。 一个字符可以对应多个字形（不同的字体，或者同种字体的不同样式:粗体斜体等）；多个字符也可能对应一个字形，比如字符的连写（ Ligatures）。

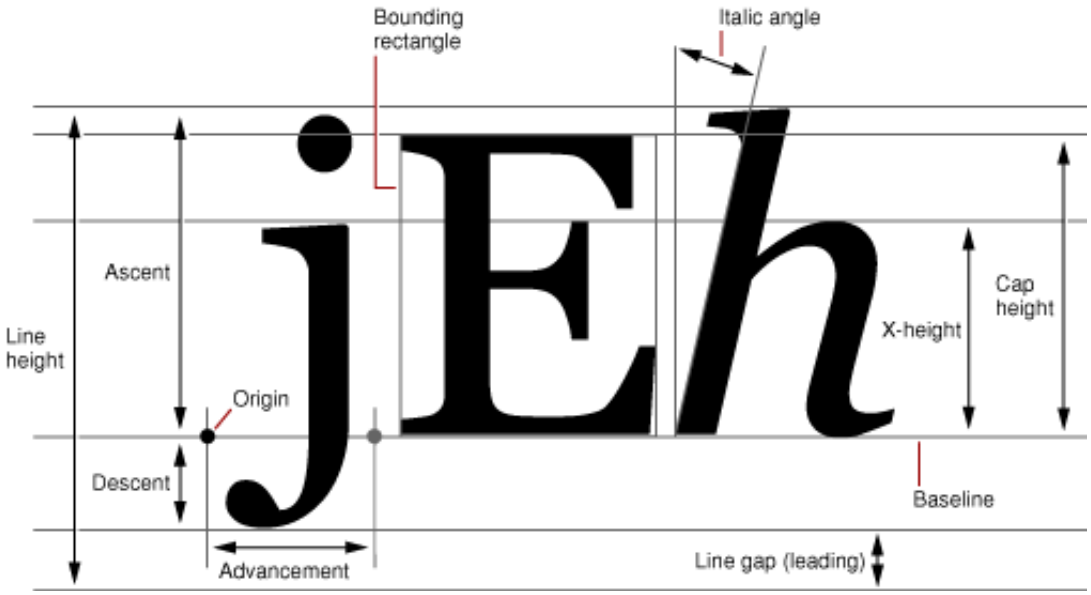




Roman Ligatures

下面就来详情看看字形的各个参数也就是所谓的字形度量Glyph Metrics





- bounding box（边界框 bbox），这是一个假想的框子，它尽可能紧密的装入字形。
- baseline（基线），一条假想的线,一行上的字形都以此线作为上下位置的参考，在这条线的左侧存在一个点叫做基线的原点，
- ascent（上行高度）从原点到字体中最高（这里的高深都是以基线为参照线的）的字形的顶部的距离，ascent是一个正值
- descent（下行高度）从原点到字体中最深的字形底部的距离，descent是一个负值（比如一个字体原点到最深的字形的底部的距离为2，那么descent就为-2）
- linegap（行距），linegap也可以称作leading（其实准确点讲应该叫做External leading）,行高lineHeight则可以通过 ascent + |descent| + linegap 来计算。

一些Metrics专业知识还可以参考Free Type的文档 Glyph metrics，其实iOS就是使用Free Type库来进行字体渲染的。

以上图片和部分概念来自苹果文档 Querying Font Metrics ， Text Layout

2.坐标系

首先不得不说 苹果编程中的坐标系花样百出，经常让开发者措手不及。 传统的Mac中的坐标系的原点在左下角，比如NSView默认的坐标系，原点就在左下角。但Mac中有些View为了其实现的便捷将原点变换到左上角，像NSTableView的坐标系坐标原点就在左上角。iOS UIKit的UIView的坐标系原点在左上角。
往底层看，Core Graphics的context使用的坐标系的原点是在左下角。而在iOS中的底层界面绘制就是通过Core Graphics进行的，那么坐标系列是如何变换的呢？ 在UIView的drawRect方法中我们可以通过UIGraphicsGetCurrentContext()来获得当前的Graphics Context。drawRect方法在被调用前，这个Graphics Context被创建和配置好，你只管使用便是。如果你细心，通过CGContextGetCTM(CGContextRef c)可以看到其返回的值

并不是CGAffineTransformIdentity，通过打印出来看到值为

```
Printing description of contextCTM:(CGAffineTransform) contextCTM = {
        a = 1        b = 0        c = 0        d = -1        tx = 0
      ty = 460}
```

这是非retina分辨率下的结果，如果是如果是retina上面的a,d,ty的值将会乘2，如果是iPhone 5，ty的值会再大些。 但是作用都是一样的就是将上下文空间坐标系进行了flip，使得原本左下角原点变到左上角，y轴正方向也变换成向下。

还是老样子，拿一个事先定义好的属性字串进行开讲。

[cpp] view plain copy

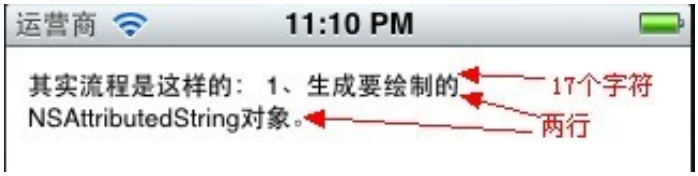
```
01. NSString *src = [NSString stringWithString:@"其实流程是这样的： 1、生成要绘制的
    NSAttributedString对象。 "];
02.
03.  NSMutableAttributedString * mabstring = [[NSMutableAttributedString
    alloc]initWithString:src];
04.
05. long slen = [mabstring length];
```

将属性字串放到frame当中。

[cpp] view plain copy

```
01. CTFramesetterRef framesetter =
    CTFramesetterCreateWithAttributedString((CFAttributedStringRef)mabstring);
02.
03. CGMutablePathRef Path = CGPathCreateMutable();
04.
05. //坐标点在左下角
06. CGPathAddRect(Path, NULL ,CGRectMake(10 , 10 ,self.bounds.size.width-20 ,
    self.bounds.size.height-20));
07.
08. CTFrameRef frame = CTFramesetterCreateFrame(framesetter, CFRangeMake(0, 0), Path,
    NULL);
```

显示效果：



得到属性字串在frame中被自动分成了多少个行。每行中有多少个CTRun

[cpp] view plain copy

```
01. //得到frame中的行数组
02.  CFArrayRef rows = CTFrameGetLines(frame);
03.
04. int rowcount = CFArrayGetCount(rows);
05.
06.  NSLog(@"rowcount = %i",rowcount);
07.
08.  CTLineRef line = CFArrayGetValueAtIndex(rows, 0);
09.
10. //从一行中得到CTRun数组
11.  CFArrayRef runs = CTLineGetGlyphRuns(line);
12. int runcount = CFArrayGetCount(runs);
13.
14.  NSLog(@"runcount = %i",runcount);
```

结果：

[cpp] view plain copy

```
01. 2013-03-20 23:07:38.835 CTextDemo[5612:207] rowcount = 2
02. 2013-03-20 23:07:38.838 CTextDemo[5612:207] runcount = 17
```

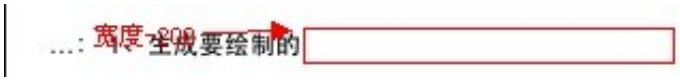
将第一行设置为使用省略号模式

[cpp] view plain copy

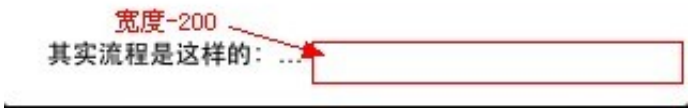
```
01. NSAttributedString *truncatedString = [[NSAttributedString alloc]initWithString:@"\u2026"];
02.  CTLineRef token = CTLineCreateWithAttributedString((__bridge
     CFAttributedStringRef)truncatedString);
03.
04.  CTLineTruncationType ltt = kCTLineTruncationStart;//kCTLineTruncationEnd;
05.  CTLineRef newline = CTLineCreateTruncatedLine(line, self.bounds.size.width-200, ltt, token);
```

```
 CGContextSetTextPosition(context,20, 20);
CTLineDraw(newline, context);
```

效果：



CTLineTruncationType 为kCTLineTrunceationEnd;



省略号在中间

> 其实流程...要绘制的

```
CFIndex CTLineGetGlyphCount( CTLineRef line );
```

获取一行中的图像个数，即有多少个CTRun。

```
CFArrayRef CTLineGetGlyphRuns( CTLineRef line );
```

获取CTRUN数组，可以通过CFArrayGetCount得到数组的个数得到的值与CTLineGetGlyphCount相同。

```
CGFloat CTLineGetOffsetForStringIndex( CTLineRef line, CFIndex charIndex, CGFloat* secondaryOffset );
```

获取一行文字中，指定charIndex字符相对x原点的偏移量，返回值与secondaryOffset同为一个值。如果charIndex超出一行的字符长度则反回最大长度结束位置的偏移量，如一行文字共有17个字符，哪么返回的是第18个字符的起始偏移，即第17个偏移+第17个字符占有的宽度=第18个起始位置的偏移。因此想求一行字符所占的像素长度时，就可以使用此函数，将charIndex设置为大于字符长度即可。

[cpp] view plain copy

```
01. //获取整段文字中charIndex位置的字符相对line的原点的x值
02.     CGFloat offset;
03.     CGFloat retoffset = CTLineGetOffsetForStringIndex(line,1,&offset);
04.     NSLog(@"return offset = %f",retoffset);
05.     NSLog(@"output offset = %f",offset);
```

效果：

[cpp] view plain copy

```
01. 2013-03-21 13:37:22.330 CTextDemo[6851:207] return offset = 12.000000
02. 2013-03-21 13:37:22.331 CTextDemo[6851:207] output offset = 12.000000
```


```
double CTLineGetPenOffsetForFlush( CTLineRef line, CGFloat flushFactor, double flushWidth );
```

获取相对于Flush的偏移量。即[flushwidth - line(字符占的像素)]*flushFactor/100;这是我个人推的公式，发现精确度上还存在偏差。

当flushFactor取值为0,0.5,1时分别显示的效果为左对齐，居中对齐，右对齐。

演示代码:

[cpp]
view plain
copy

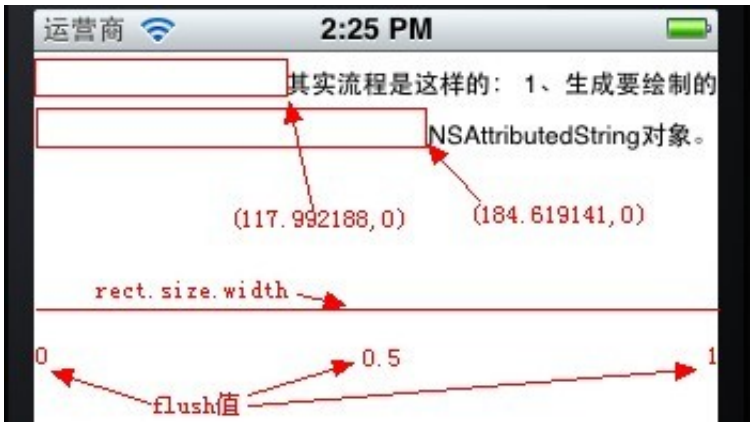
```
01.  - (void)drawRect:(CGRect)rect

02.  {
```

```
03.       NSString *src = [NSString stringWithString:@"其实流程是这样的:  1、生成要绘制的NSAttributedString对象。 "];

04.

05.     NSMutableAttributedString * mabstring = [[NSMutableAttributedString alloc]initWithString:src];

06.

07. long slen = [mabstring length];

08.

09.

10.     CTFramesetterRef framesetter = CTFramesetterCreateWithAttributedString((CFAttributedStringRef)mabstring);

11.

12.     CGMutablePathRef Path = CGPathCreateMutable();

13.

14. //坐标点在左下角

15.     CGPathAddRect(Path, NULL ,CGRectMake(10 , 10 ,self.bounds.size.width-20 , self.bounds.size.height-20));

16.

17.     CTFrameRef frame = CTFramesetterCreateFrame(framesetter, CFRangeMake(0, 0), Path, NULL);

18.

19.

20. //得到frame中的行数组
```

```
21.     CFArrayRef rows = CTFrameGetLines(frame);
22.
23. if (rows) {
24. const CFIndex numberOfLines = CFArrayGetCount(rows);
25. const CGFloat fontLineHeight = [UIFont systemFontOfSize:20].lineHeigh
    t;
26.         CGFloat textOffset = 0;
27.
28.         CGContextRef ctx = UIGraphicsGetCurrentContext();
29.         CGContextSaveGState(ctx);
30.         CGContextTranslateCTM(ctx, rect.origin.x, rect.origin.y+[UIFo
    nt systemFontOfSize:20].ascender);
31.         CGContextSetTextMatrix(ctx, CGAffineTransformMakeScale(1,-1))
    ;
32.
33. for (CFIndex lineNumber=0; lineNumber
34.             CTLineRef line = CFArrayGetValueAtIndex(rows, lineNumber)
    ;
35. float flush;
36. switch (2) {
37. case UITextAlignmentCenter: flush = 0.5;    break; //1
38. case UITextAlignmentRight:  flush = 1;      break; //2
```

```
39. case UITextAlignmentLeft:  //0
40. default:                    flush = 0;      break;
41.             }
42.
43.             CGFloat penOffset = CTLineGetPenOffsetForFlush(line, flus
    h, rect.size.width);
44.             NSLog(@"penOffset = %f",penOffset);
45.             CGContextSetTextPosition(ctx, penOffset, textOffset);//在
    偏移量x,y上打印
46.             CTLineDraw(line, ctx);//draw 行文字
47.             textOffset += fontLineHeight;
48.         }
49.
50.         CGContextRestoreGState(ctx);
51.
52.     }
53. }
```

效果：




```
CFIndex CTLineGetStringIndexForPosition( CTLineRef line, CGPoint posi
tion );
```

获取一行中光标点击处（position）的字符索引，这个值只能为0或最大字符长度。

```
CFRange CTLineGetStringRange( CTLineRef line );
```

获取一行字符占的范围(包括换行符一起计算)，返回一行位置的起始位置(location)和长度(length)。

location不是每行都从0开始的，而是该行的前N行字符和。

```
double CTLineGetTrailingWhitespaceWidth( CTLineRef line );
```

获取一行未尾字符后空格的像素长度。如果："abc "后面有两个空格，返回的就是这两个空格占有的像素长度。

[cpp]
view plain
copy

```
01.  double wspace = CTLineGetTrailingWhitespaceWidth(line);

02.  NSLog(@"whitespacewidth = %f",wspace);
```

double CTLineGetTypographicBounds(CTLineRef line, CGFloat* ascent, CGFloat* descent, CGFloat* leading);

获取一行中上行高(ascent)，下行高(descent)，行距(leading),整行高为(ascent+|descent|+leading) 返回值为整行字符串长度占有的像素宽度。

[cpp]
view plain
copy

```
01.  CGFloat asc,des,lead;

02.  double lineHeight = CTLineGetTypographicBounds(line, &asc, &des, &lea
     d);

03.      NSLog(@"ascent = %f,descent = %f,leading = %f,lineheight = %f",as
     c,des,lead,lineHeight);
```

```
CGRect CTLineGetImageBounds( CTLineRef line, CGContextRef context );
```

获取一行文字的范围，什么意思，就是指把这一行文字点有的像素距阵作为一个image图片，来得到整个矩形区域。

演示代码:

[cpp]
view plain
copy

```
01. -(void)drawBounds

02. {
```

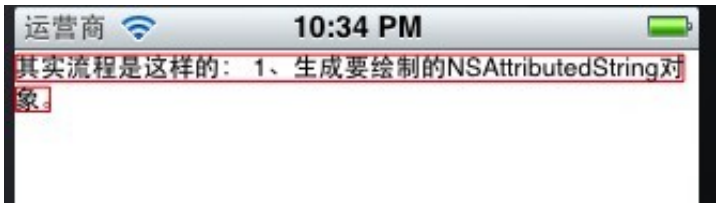
```
03.     NSString *src = [NSString stringWithString:@"其实流程是这样的： 1、生成要绘制的NSAttributedString对象。 "];

04.

05.     NSAttributedString * string = [[NSAttributedString alloc]initWithString:src];

06.

07.     CGContextRef ctx = UIGraphicsGetCurrentContext();

08.

09.     CGContextSetTextMatrix(ctx , CGAffineTransformIdentity);

10.

11. //CGContextSaveGState(ctx);

12.

13. //x，y轴方向移动

14.     CGContextTranslateCTM(ctx , 0 ,self.bounds.size.height);

15.

16. //缩放x，y轴方向缩放，－1.0为反向1.0倍,坐标系转换,沿x轴翻转180度

17.     CGContextScaleCTM(ctx, 1.0 ,-1.0);

18.

19. // layout master

20.     CTFramesetterRef framesetter = CTFramesetterCreateWithAttributedString(
```

```
21.           (CFAttributedStringRef)string);
22.     CGMutablePathRef Path = CGPathCreateMutable();
23.
24. //坐标点在左下角
25.     CGPathAddRect(Path, NULL ,CGRectMake(0 , 0 ,self.bounds.size.widt
    h , self.bounds.size.height));
26.
27.     CTFrameRef frame = CTFramesetterCreateFrame(framesetter, CFRangeM
    ake(0, 0), Path, NULL);
28.
29.     CFArrayRef Lines = CTFrameGetLines(frame);
30.
31. int linecount = CFArrayGetCount(Lines);
32.
33.     CGPoint origins[linecount];
34.     CTFrameGetLineOrigins(frame,
35.                           CFRangeMake(0, 0), origins);
36.     NSInteger lineIndex = 0;
37.
38. for (id oneLine in (NSArray *)Lines)
39.     {
```

```
40.         CGRect lineBounds = CTLineGetImageBounds((CTLineRef)oneLine,
    ctx);

41.

42.         lineBounds.origin.x += origins[lineIndex].x;

43.         lineBounds.origin.y += origins[lineIndex].y;

44.

45.         lineIndex++;

46. //画长方形

47.

48. //设置颜色，仅填充4条边

49.         CGContextSetStrokeColorWithColor(ctx, [[UIColor redColor] CGC
    olor]);

50. //设置线宽为1

51.         CGContextSetLineWidth(ctx, 1.0);

52. //设置长方形4个顶点

53.         CGPoint poins[] = {CGPointMake(lineBounds.origin.x, lineBound
    s.origin.y),CGPointMake(lineBounds.origin.x+lineBounds.size.width, li
    neBounds.origin.y),CGPointMake(lineBounds.origin.x+lineBounds.size.wi
    dth, lineBounds.origin.y+lineBounds.size.height),CGPointMake(lineBoun
    ds.origin.x, lineBounds.origin.y+lineBounds.size.height)};

54.         CGContextAddLines(ctx,poins,4);

55.         CGContextClosePath(ctx);

56.         CGContextStrokePath(ctx);

57.
```

```
58.      }
59.
60.
61.
62.      CTFrameDraw(frame,ctx);
63.
64.      CGPathRelease(Path);
65.      CFRelease(framesetter);
66.  }
```

效果图：



通这个RECT我们可以对文字增加点击事件或其它触发动作等。

OK，CTLine 介绍完毕。