在前面一篇文章中，介绍了属性文字的基本使用，本章节主要针对文字的段落样式展开演示说明。

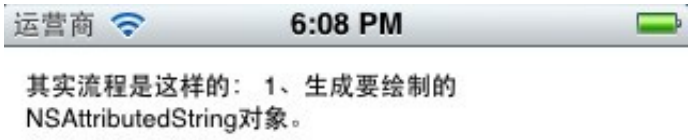
先定义一段演示文字（文字中有中，英文）。

[cpp] view plain copy

```
NSString *src = [NSString stringWithString:@"其实流程是这样的： 1、生成要绘制的NSAttributedString对象。 2、生成一个CTFramesetterRef对象，然后创建一个CGPath对象，这个Path对象用于表示可绘制区域坐标值、长宽。 3、使用上面生成的setter和path生成一个CTFrameRef对象，这个对象包含了这两个对象的信息（字体信息、坐标信息），它就可以使用CTFrameDraw方法绘制了。"];
    NSMutableAttributedString * mabstring = [[NSMutableAttributedString alloc]initWithString:src];

long slen = [mabstring length];
```

[cpp] view plain copy

```
"font-family: Arial, Helvetica, sans-serif;" > 在未设置段落样式的情况下，效果：
```



从上面的交果来看，想必大家也看到了，英文部份换行显示了。这个一般情况下不注意，但在大的段落文章中就会出现不对齐现象。

先不管上面的，下面逐个来演示一下段落属性。
段落样式定义：

[cpp] view plain copy

```
kCTParagraphStyleSpecifierAlignment = 0,              //对齐属性
 kCTParagraphStyleSpecifierFirstLineHeadIndent = 1,    //首行缩进
 kCTParagraphStyleSpecifierHeadIndent = 2,             //段头缩进
 kCTParagraphStyleSpecifierTailIndent = 3,             //段尾缩进
 kCTParagraphStyleSpecifierTabStops = 4,               //制表符模式
 kCTParagraphStyleSpecifierDefaultTabInterval = 5,     //默认tab间隔
 kCTParagraphStyleSpecifierLineBreakMode = 6,          //换行模式
 kCTParagraphStyleSpecifierLineHeightMultiple = 7,     //多行高
 kCTParagraphStyleSpecifierMaximumLineHeight = 8,      //最大行高
 kCTParagraphStyleSpecifierMinimumLineHeight = 9,      //最小行高
 kCTParagraphStyleSpecifierLineSpacing = 10,           //行距
 kCTParagraphStyleSpecifierParagraphSpacing = 11,      //段落间距  在段的末尾（Bottom）加上间隔，这个值为负数。
 kCTParagraphStyleSpecifierParagraphSpacingBefore = 12,  //段落前间距 在一个段落的前面加上间隔。TOP
 kCTParagraphStyleSpecifierBaseWritingDirection = 13,    //基本书写方向
 kCTParagraphStyleSpecifierMaximumLineSpacing = 14,     //最大行距
 kCTParagraphStyleSpecifierMinimumLineSpacing = 15,     //最小行距
```

17. kCTParagraphStyleSpecifierLineSpacingAdjustment = 16, //行距调整
18. kCTParagraphStyleSpecifierCount = 17, //

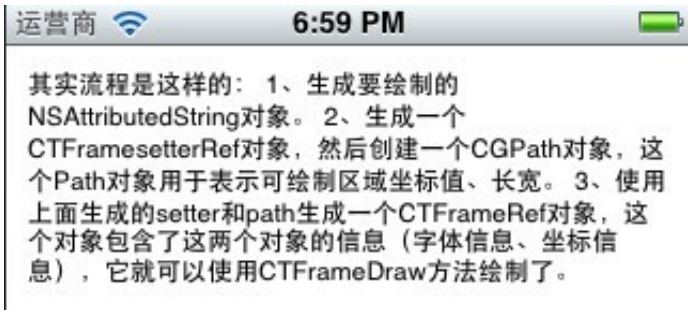
对齐属性：

```
kCTLeftTextAlignment = 0,                //左对齐kCTRightTextAlignment
 = 1,              //右对齐kCTCenterTextAlignment = 2,              /
/居中对齐kCTJustifiedTextAlignment = 3,          //文本对齐kCTNaturalTe
xtAlignment = 4              //自然文本对齐
```

段落默认样式为

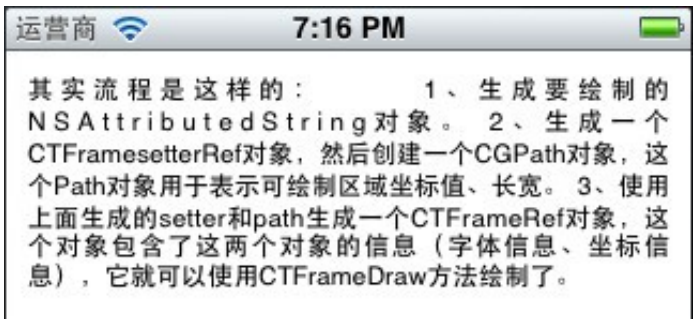
```
kCTNaturalTextAlignment
```

效果：



居中：

文本对齐Justified效果



对齐方式设置代码：

[cpp]
view plain
copy

```
01.  CTTextAlignment alignment = kCTJustifiedTextAlignment;

02.      CTParagraphStyleSetting alignmentStyle;

03.      alignmentStyle.spec=kCTParagraphStyleSpecifierAlignment;//指定为对齐属性

04.      alignmentStyle.valueSize=sizeof(alignment);
```

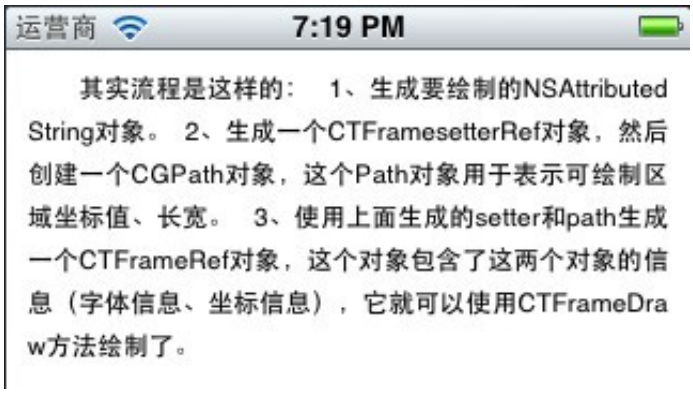
```
05.      alignmentStyle.value=&alignment;
```

首行缩进代码:

```
[cpp]
view plain
copy
```

```
01.  //首行缩进

02.      CGFloat fristlineindent = 24.0f;

03.      CTParagraphStyleSetting fristline;

04.      fristline.spec = kCTParagraphStyleSpecifierFirstLineHeadIndent;

05.      fristline.value = &fristlineindent;

06.      fristline.valueSize = sizeof(float);
```

效果:

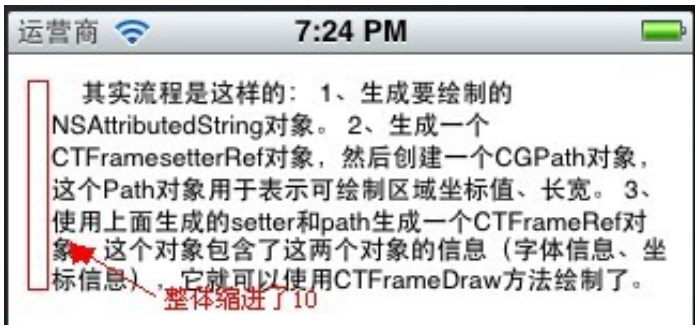


段头缩进代码:

[cpp]
view plain
copy

```
01.  //段缩进

02.  CGFloat headindent = 10.0f;

03.  CTParagraphStyleSetting head;

04.  head.spec = kCTParagraphStyleSpecifierHeadIndent;

05.  head.value = &headindent;

06.  head.valueSize = sizeof(float);
```

效果：



段尾缩进代码：

```
[cpp]
view plain
copy
```

```
01. //段尾缩进
02. CGFloat tailindent = 50.0f;
03. CTParagraphStyleSetting tail;
04. tail.spec = kCTParagraphStyleSpecifierTailIndent;
05. tail.value = &tailindent;
06. tail.valueSize = sizeof(float);
```

效果:

制表符（tab）代码：

```
[cpp]
view plain
copy
```

```
01.  //tab

02.      CTTextAlignment tabalignment = kCTJustifiedTextAlignment;

03.      CTTextTabRef texttab = CTTextTabCreate(tabalignment, 24, NULL);

04.      CTParagraphStyleSetting tab;

05.      tab.spec = kCTParagraphStyleSpecifierTabStops;

06.      tab.value = &texttab;

07.      tab.valueSize = sizeof(CTTextTabRef);
```

效果（未看出哪有变化感觉行距大了点）：

换行模式：

```
kCTLineBreakByWordWrapping = 0,         //出现在单词边界时起作用，如果该单词不在能在一行里显示时，整体换行。此为段的默认值。kCTLineBreakByCharWrapping = 1,         //当一行中最后一个位置的大小不能容纳一个字符时，才进行换行。kCTLineBr
```

eakByClipping = 2,　　//超出画布边缘部份将被截除。kCTLineBreakByTruncatingHead = 3,　　//截除前面部份，只保留后面一行的数据。前部份以...代替。kCTLineBreakByTruncatingTail = 4,　　//截除后面部份，只保留前面一行的数据，后部份以...代替。kCTLineBreakByTruncatingMiddle = 5　　//在一行中显示段文字的前面和后面文字，中间文字使用...代替。

换行模式代码:

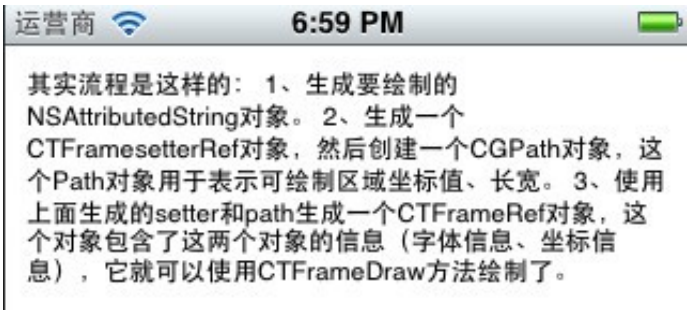
[cpp]
view plain
copy

```
01. //换行模式

02.     CTParagraphStyleSetting lineBreakMode;
```

```
03.     CTLineBreakMode lineBreak = kCTLineBreakByWordWrapping;//kCTLineBr
    eakByCharWrapping;//换行模式

04.     lineBreakMode.spec = kCTParagraphStyleSpecifierLineBreakMode;

05.     lineBreakMode.value = &lineBreak;

06.     lineBreakMode.valueSize = sizeof(CTLineBreakMode);
```

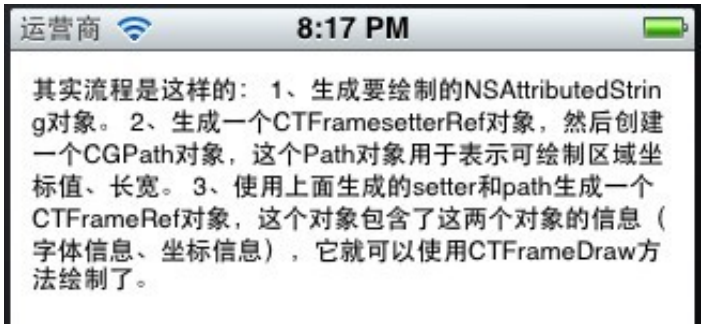
kCTLineBreakByWordWrapping

效果:



kCTLineBreakByCharWrapping

效果:



kCTLineBreakByClipping

效果：

kCTLineBreakByTruncatingHead

效果：

kCTLineBreakByTruncatingTail

效果：

kCTLineBreakByTruncatingMiddle

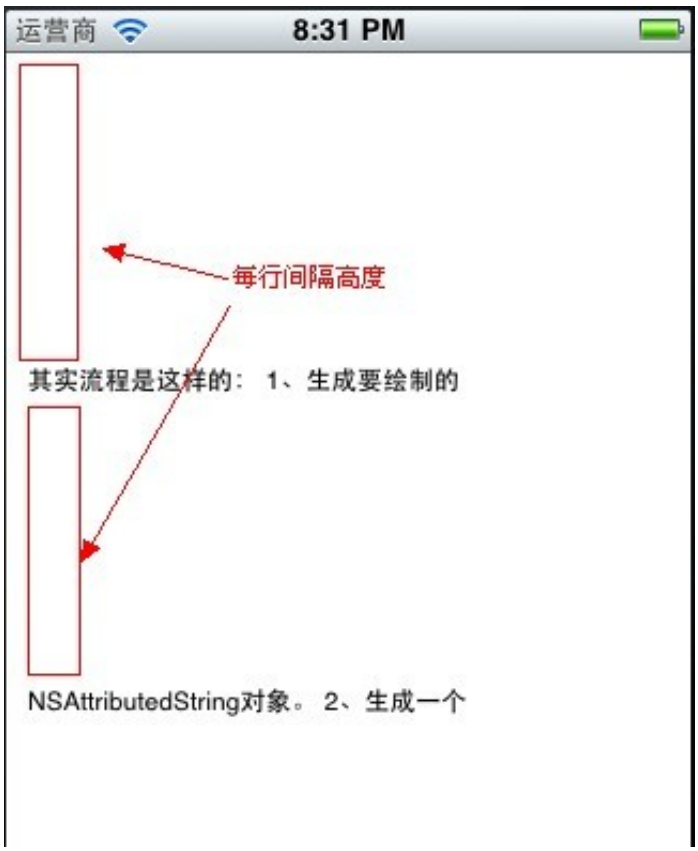
效果：

多行高设置代码：

[cpp]
view plain
copy

```
01. //多行高

02. CGFloat MutiHeight = 10.0f;

03. CTParagraphStyleSetting Muti;

04. Muti.spec = kCTParagraphStyleSpecifierLineHeightMultiple;

05. Muti.value = &MutiHeight;

06. Muti.valueSize = sizeof(float);
```

效果：

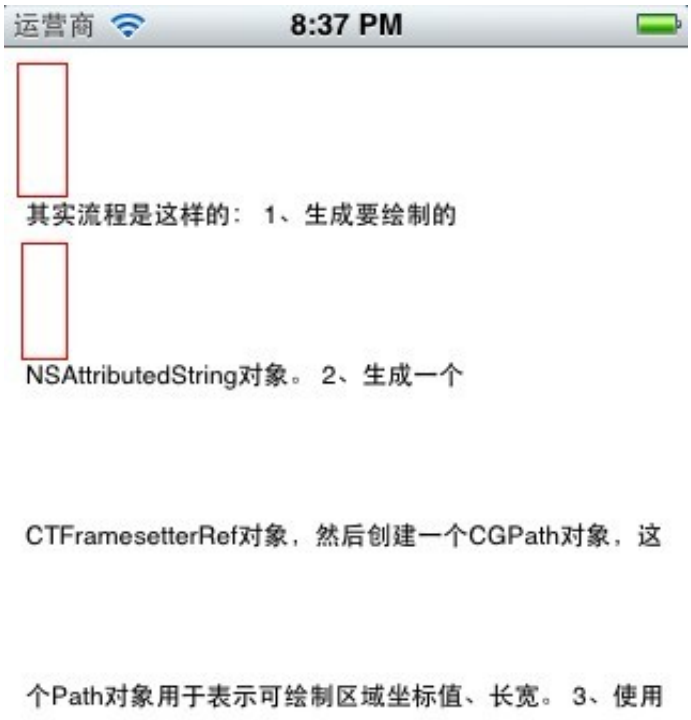

运营商
8:31 PM
每行间隔高度
其实流程是这样的： 1、生成要绘制的
NSAttributedString对象。 2、生成一个

最大行高代码：

[cpp]
view plain
copy

```
01.  //最大行高

02.      CGFloat MaxHeight = 5.0f;

03.      CTParagraphStyleSetting Max;

04.      Max.spec = kCTParagraphStyleSpecifierLineHeightMultiple;

05.      Max.value = &MaxHeight;

06.      Max.valueSize = sizeof(float);
```

效果:



行距代码：

```
[cpp]
view plain
copy
```

```
01. //行距
02. CGFloat _linespace = 5.0f;
03. CTParagraphStyleSetting lineSpaceSetting;
04. lineSpaceSetting.spec = kCTParagraphStyleSpecifierLineSpacing;
05. lineSpaceSetting.value = &_linespace;
06. lineSpaceSetting.valueSize = sizeof(float);
```

效果:

段前间距设置代码（段与段之间）：

```
[cpp]
view plain
copy
```

```
01.  //段前间隔

02.  CGFloat paragraphspace = 5.0f;

03.  CTParagraphStyleSetting paragraph;

04.  paragraph.spec = kCTParagraphStyleSpecifierLineSpacing;

05.  paragraph.value = ¶graphspace;

06.  paragraph.valueSize = sizeof(float);
```

效果:


```
kCTWritingDirectionNatural = -1,            //普通书写方向，一般习惯是从左到右kCTWritingDirectionLeftToRight = 0,         //从左到右写kCTWritingDirectionRightToLeft = 1          //从右到左写
```

基本书写方向代码:

[cpp]
view plain
copy

```
01.  //书写方向

02.  CTWritingDirection wd = kCTWritingDirectionRightToLeft;

03.  CTParagraphStyleSetting writedic;
```

```
04.  writedic.spec = kCTParagraphStyleSpecifierBaseWritingDirection;

05.  writedic.value = &wd;

06.  writedic.valueSize = sizeof(CTWritingDirection);
```

效果：

这个跟字体右对齐效果上类似。

好了，段落的API样式介绍到这里，里面还有很多配合设置时的效果。读者自行演示了。

下面附上draw 代码：

[cpp]
view plain
copy

```
01. -(void)ParagraphStyle

02. {

03.     NSString *src = [NSString stringWithString:@"其实流程是这样的： 1、生成要绘制的NSAttributedString对象。 2、生成一个CTFramesetterRef对象，然后创建一个CGPath对象，这个Path对象用于表示可绘制区域坐标值、长宽。 3、使用上面生成的setter和path生成一个CTFrameRef对象，这个对象包含了这两个对象的信息（字体信息、坐标信息），它就可以使用CTFrameDraw方法绘制了。"];

04.

05. //修改windows回车换行为mac的回车换行

06. //src = [src stringByReplacingOccurrencesOfString:@"\r\n" withString:@"\n"];

07.

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

```
09.
10. long slen = [mabstring length];
11.
12.
13. //创建文本对齐方式
14.     CTTextAlignment alignment = kCTRightTextAlignment;//kCTNaturalTex
    tAlignment;
15.     CTParagraphStyleSetting alignmentStyle;
16.     alignmentStyle.spec=kCTParagraphStyleSpecifierAlignment;//指定为对
    齐属性
17.     alignmentStyle.valueSize=sizeof(alignment);
18.     alignmentStyle.value=&alignment;
19.
20. //首行缩进
21.     CGFloat fristlineindent = 24.0f;
22.     CTParagraphStyleSetting fristline;
23.     fristline.spec = kCTParagraphStyleSpecifierFirstLineHeadIndent;
24.     fristline.value = &fristlineindent;
25.     fristline.valueSize = sizeof(float);
26.
27. //段缩进
```

```
28.      CGFloat headindent = 10.0f;
29.      CTParagraphStyleSetting head;
30.      head.spec = kCTParagraphStyleSpecifierHeadIndent;
31.      head.value = &headindent;
32.      head.valueSize = sizeof(float);
33.
34.  //段尾缩进
35.      CGFloat tailindent = 50.0f;
36.      CTParagraphStyleSetting tail;
37.      tail.spec = kCTParagraphStyleSpecifierTailIndent;
38.      tail.value = &tailindent;
39.      tail.valueSize = sizeof(float);
40.
41.  //tab
42.      CTTextAlignment tabalignment = kCTJustifiedTextAlignment;
43.      CTTextTabRef texttab = CTTextTabCreate(tabalignment, 24, NULL);
44.      CTParagraphStyleSetting tab;
45.      tab.spec = kCTParagraphStyleSpecifierTabStops;
46.      tab.value = &texttab;
47.      tab.valueSize = sizeof(CTTextTabRef);
```

```
48.
49. //换行模式
50.     CTParagraphStyleSetting lineBreakMode;
51.     CTLineBreakMode lineBreak = kCTLineBreakByTruncatingMiddle;//kCTLineBreakByWordWrapping;//换行模式
52.     lineBreakMode.spec = kCTParagraphStyleSpecifierLineBreakMode;
53.     lineBreakMode.value = &lineBreak;
54.     lineBreakMode.valueSize = sizeof(CTLineBreakMode);
55.
56. //多行高
57.     CGFloat MutiHeight = 10.0f;
58.     CTParagraphStyleSetting Muti;
59.     Muti.spec = kCTParagraphStyleSpecifierLineHeightMultiple;
60.     Muti.value = &MutiHeight;
61.     Muti.valueSize = sizeof(float);
62.
63. //最大行高
64.     CGFloat MaxHeight = 5.0f;
65.     CTParagraphStyleSetting Max;
66.     Max.spec = kCTParagraphStyleSpecifierLineHeightMultiple;
```

```
67.      Max.value = &MaxHeight;
68.      Max.valueSize = sizeof(float);
69.
70.  //行距
71.      CGFloat _linespace = 5.0f;
72.      CTParagraphStyleSetting lineSpaceSetting;
73.      lineSpaceSetting.spec = kCTParagraphStyleSpecifierLineSpacing;
74.      lineSpaceSetting.value = &_linespace;
75.      lineSpaceSetting.valueSize = sizeof(float);
76.
77.  //段前间隔
78.      CGFloat paragraphspace = 5.0f;
79.      CTParagraphStyleSetting paragraph;
80.      paragraph.spec = kCTParagraphStyleSpecifierLineSpacing;
81.      paragraph.value = ¶graphspace;
82.      paragraph.valueSize = sizeof(float);
83.
84.  //书写方向
85.      CTWritingDirection wd = kCTWritingDirectionRightToLeft;
86.      CTParagraphStyleSetting writedic;
```

```
87.      writedic.spec = kCTParagraphStyleSpecifierBaseWritingDirection;
88.      writedic.value = &wd;
89.      writedic.valueSize = sizeof(CTWritingDirection);
90.
91.  //组合设置
92.      CTParagraphStyleSetting settings[] = {
93.          alignmentStyle
94.          fristline,
95.          head,
96.          tail,
97.          tab,
98.          lineBreakMode,
99.          Muti,
100.         Max,
101.         lineSpaceSetting,
102.         writedic
103.         indentSetting
104.
105.     };
106.
```

```
107. //通过设置项产生段落样式对象
108.     CTParagraphStyleRef style = CTParagraphStyleCreate(settings, 11);
109.
110. // build attributes
111.     NSMutableDictionary *attributes = [NSMutableDictionary dictionary
     WithObject:(id)style forKey:(id)kCTParagraphStyleAttributeName ];
112.
113. // set attributes to attributed string
114.     [mabstring addAttributes:attributes range:NSMakeRange(0, slen)];
115.
116.
117.     CTFramesetterRef framesetter = CTFramesetterCreateWithAttributedS
     tring((CFAttributedStringRef)mabstring);
118.
119.     CGMutablePathRef Path = CGPathCreateMutable();
120.
121. //坐标点在左下角
122.     CGPathAddRect(Path, NULL ,CGRectMake(10 , 10 ,self.bounds.size.wi
     dth-20 , self.bounds.size.height-20));
123.
124.     CTFrameRef frame = CTFramesetterCreateFrame(framesetter, CFRangeM
     ake(0, 0), Path, NULL);
```

```
125.
126.
127.
128.  //获取当前(View)上下文以便于之后的绘画，这个是一个离屏。
129.      CGContextRef context = UIGraphicsGetCurrentContext();
130.
131.      CGContextSetTextMatrix(context , CGAffineTransformIdentity);
132.
133.  //压栈，压入图形状态栈中.每个图形上下文维护一个图形状态栈，并不是所有的当前绘画环境
      的图形状态的元素都被保存。图形状态中不考虑当前路径，所以不保存
134.  //保存现在得上下文图形状态。不管后续对context上绘制什么都不会影响真正得屏幕。
135.      CGContextSaveGState(context);
136.
137.  //x，y轴方向移动
138.      CGContextTranslateCTM(context , 0 ,self.bounds.size.height);
139.
140.  //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
141.      CGContextScaleCTM(context, 1.0 ,-1.0);
142.
143.      CTFrameDraw(frame,context);
```

```
144.
145.     CGPathRelease(Path);
146.     CFRelease(framesetter);
147. }
```

```
- (void) drawRect:(CGRect)rect
{
   [self ParagraphStyle];
}
```