NSDate和NSDateFormatter深入理解

NSDate对象描述的是手机当前正显示的时间转换为公历0时区的时间，相对于2001/1/1零时的秒数。NSDate对象在它生成的那一刻，就是一个固定的不可变的时间点(到2001/1/1零点的秒数，unix时间戳是相对于1970/1/1零时的秒数)，它参考的值是：以UTC为标准的，2001年1月1日00：00：00这一刻的时间绝对值。故NSDate和时区和文化无关，但是开发断点调试时(其实和APP当时调用[NSDate date]那一刻的手机时区有关，但是得到的date已经减去了当时手机的时区，故可以看做跟时区无关了)

什么是Unix时间戳(Unix timestamp)： Unix时间戳(Unix timestamp)，或称Unix时间(Unix time)、POSIX时间(POSIX time)，是一种时间表示方式，定义为从格林威治时间1970年01月01日00时00分00秒起至现在的总秒数。Unix时间戳不仅被使用在Unix系统、类Unix系统中，也在许多其他操作系统中被广泛采用。

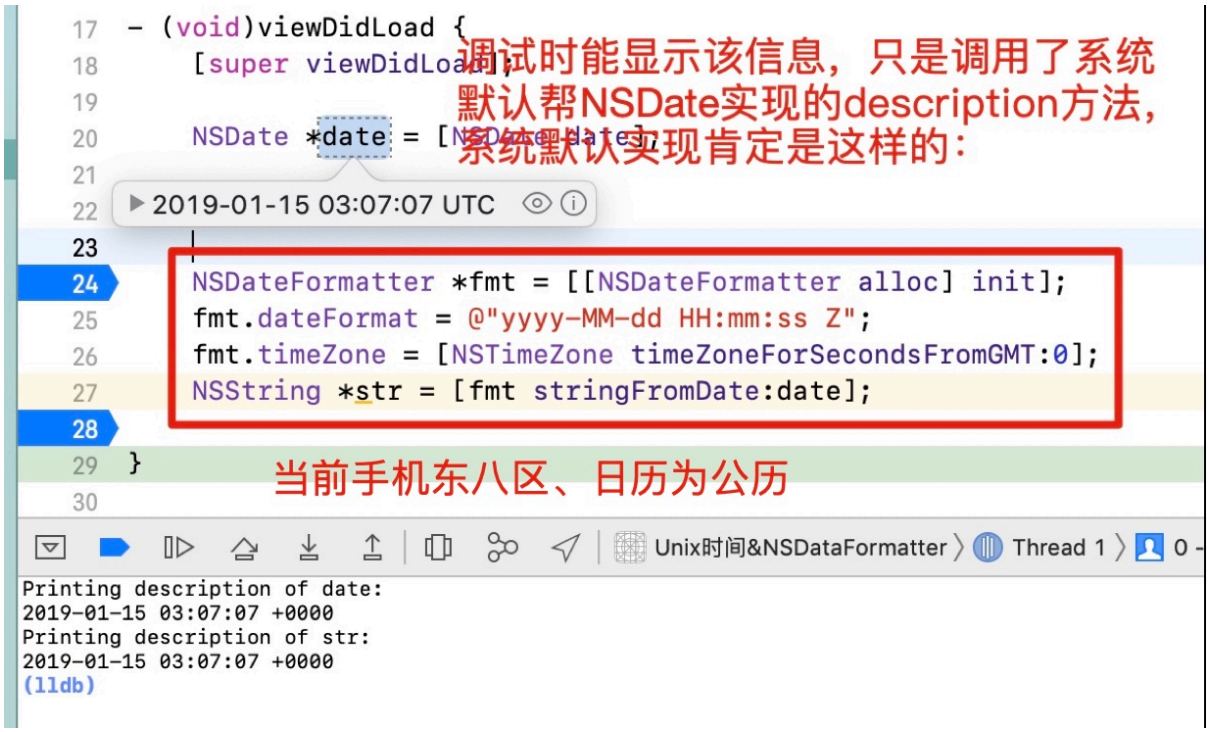
场景1

```
17  - (void)viewDidLoad {
18      [super viewDidLoad];
19
20      NSDate *date = [NSDate date];
21
22      NSDateFormatter *fmt = [[NSDateFormatter alloc] init];
23      fmt.dateFormat = @"yyyy-MM-dd HH:mm:ss Z";
24      NSString *str = [fmt stringFromDate:date];
25
26  }
27
28  /*
```

当前手机东八区、日历为公历

```
Unix时间&NSDataFormatter 〉 Thread 1 〉 0 -[ViewC
Printing description of date:
2019-01-15 03:31:37 +0000
Printing description of str:
2019-01-15 11:31:37 +0800
(lldb)
```

场景2


- (void)viewDidLoad {
[super viewDidLoad];
NSDate *date = [NSDate date];
2019-01-15 03:07:07 UTC
NSDateFormatter *fmt = [[NSDateFormatter alloc] init];
fmt.dateFormat = @"yyyy-MM-dd HH:mm:ss Z";
fmt.timeZone = [NSTimeZone timeZoneForSecondsFromGMT:0];
NSString *str = [fmt stringFromDate:date];
}
调试时能显示该信息，只是调用了系统默认帮NSDate实现的description方法，系统默认实现肯定是这样的：
当前手机东八区、日历为公历
Unix时间&NSDataFormatter
Thread 1
Printing description of date:
2019-01-15 03:07:07 +0000
Printing description of str:
2019-01-15 03:07:07 +0000
(lldb)

场景3


- (void)viewDidLoad {
[super viewDidLoad];
NSDate *date = [NSDate date];
NSDateFormatter *fmt = [[NSDateFormatter alloc] init];
fmt.dateFormat = @"yyyy-MM-dd HH:mm:ss Z";
fmt.timeZone = [NSTimeZone timeZoneForSecondsFromGMT:0];
NSString *str = [fmt stringFromDate:date];
@"0031-01-15 03:17:23 +0000"
当前手机为+8时区，日本日历
Unix时间&NSDataFormatter
Thread 1
0 -[ViewController viewDidLoad]
Printing description of date:
2019-01-15 03:17:23 +0000
Printing description of str:
0031-01-15 03:17:23 +0000
(lldb)

场景4

A representation of a specific point in time, independent of any calendar or time zone.
NSDate objects encapsulate a single point in time, independent of any particular calendrical system or time zone. Date objects are immutable, representing an invariant time interval relative to an absolute reference date (00:00:00 UTC on 1 January 2001).
The NSDate class provides methods for comparing dates, calculating the time interval between two dates, and creating a new date from a time interval relative to another date. NSDate objects can be used in conjunction with NSDateFormatter objects to create localized representations of dates and times, as well as with NSCalendar objects to perform calendar arithmetic.
NSDate is toll-free bridged with its Core Foundation counterpart, CFDateRef. See Toll-Free Bridging for more information on toll-free bridging.
Important
The Swift overlay to the Foundation framework provides the Date structure, which bridges to the NSDate class. For more information about value types, see Working with Cocoa Frameworks in Using Swift with Cocoa and Objective-C (Swift 4.1).

表示特定时间点，与任何日历或时区无关。
NSDate对象封装单个时间点，独立于任何特定的日历系统或时区。日期对象是不可变的，表示相对于绝对参考日期（2001年1月1日00:00:00 UTC）的不变时间间隔。
NSDate类提供了比较日期，计算两个日期之间的时间间隔以及从相对于另一个日期的时间间隔创建新日期的方法。 NSDate对象可以与NSDateFormatter对象一起使用，以创建日期和时间的本地化表示，以及与NSCalendar对象一起执行日历算法。
NSDate与其核心基金会同行CFDateRef建立了免费电话。有关免费桥接的更多信息，请参阅免费电话桥接。
重要
Swift覆盖到Foundation框架提供了Date结构，它结合了NSDate类。有关值类型的更多信息，请参阅使用Swift与Cocoa和Objective-C一起使用Cocoa框架（Swift 4.1）。

Biāoshì tèdìng shíjiān diǎn, yǔ rènhé rìlì huò shíqū wúguān.
NSDate duìxiàng fēngzhuāng dāngè shíjiān diǎn, dúlì yú rènhé tèdìng de rìlì xìtǒng huò shíqū. Rìqí duìxiàng shì bùkě biàn de, biāoshì xiāngduì yú juéduì cānkǎo rìqí (2001 nián 1

展开

当前手机时间为 2019-01-01 11:32:08 +0800

```
po [NSDate date]
2019-01-01 03:32:08 +0000
```

[NSDate date]计算逻辑：

1.获取手机当前正显示的年月日时分秒(用户手动修改为错误的日期，也会拿到手机当前正显示的这个错误时间，总之就是以手机正显示的时间为准)
2.将手机正显示的年月日时分秒，转换为公历(可能会改变年月日，不会变时分秒)
3.获取手机当前时区(东八区-8，东九区-9)； 将转换为公历后的时间，加上或者减去当前时区与0时区的差异小时数（年月日，时分秒都可能会改变；比如正在跨年时）
4.最终得到的为公历0时区的当前时间。最后将这个公历0时区的时间转换为相对于2001/1/1零时的秒数(unix时间戳是相对于1970/1/1零时的秒数)。

这样[NSDate date]获取到的时间就与当时手机设置的日历和时区无关了.
但如果用户手动修改了手机时间，比如现在明明是：公历2019/1/1 00:00:00 +0800，用户非要手动修改为：公历2018/12/31 00:00:00 +0800，那么[NSDate date]获取到的也是：公历2018/12/31 00:00:00 +0800，已经是公历，不用转换，再减去东八区-8，最终得到：公历2018/12/30 16:00:00 +0000
这样对APP没有任何影响，只会导致APP和服务器真实时间对应不上。

[NSDate date]:
将手机当前正显示的时间转换为公历0时区的时间，相对于2001/1/1零时的秒数

理解了NSDate之后，很容易理解NSDateFormatter，这个类只是一个时间和字符串转换工具，所有的NSDate产生的那一刻就是一个固定不变的秒数了，至于需要将这个秒数转换成哪种日历、哪个时区、哪种格式的字符串给用户看，这完全取决于你给NSDateFormatter设置的日历、时区、格式。故设置NSDateFormatter的日历、时区、格式，对NSDate(自对象生成就是固定不变的一个秒数)没有任何影响，只会影响由NSDate转换后的字符串的值。

```
//date转str：NSDateFormatter的日历和时区，决定转换后的字符串的显示格式
//str转date：NSDateFormatter的日历和时区，明确转换前字符串的来源信息，以便系统能够根据你提供的字符串的时区和日历，准确的转换为NSDate(公历、0时区)
```

详见github demo: Unix时间&NSDataFormatter

NSDateFormatter常见写法:

```
    NSDateFormatter *formatter = [[NSDateFormatter alloc] init]; //如不进一步设置，则默认赋值手机当前的日历和时区

    formatter.calendar = [[NSCalendar alloc] initWithCalendarIdentifier:NSCalendarIdentifierGregorian]; //强制所有人都显示公历,防止用户切换手机日历为佛历

    [formatter setTimeZone:[NSTimeZone timeZoneWithAbbreviation:@"UTC"]];
    [formatter setTimeZone:[NSTimeZone timeZoneWithAbbreviation:@"GMT"]];
    formatter.timeZone = [NSTimeZone timeZoneForSecondsFromGMT:0]; //0时区

    [formatter setDateFormat:@"YYYY-MM-dd HH:mm:ss"]; //以周计算年，会导致2018.12.31被错计算为2019.12.31
    [formatter setDateFormat:@"yyyy-MM-dd HH:mm:ss Z"];
```

NSDateFormatter 'YYYY' 和 'yyyy' 的区别

2018/12/31是周一，如果写作YYYY则2018/12/31因为位于2019年的第一周内，故会被当做2019/12/31

过程4： 于是开始了度娘，发现**YYYY**是表示：当天所在的周属于的年份，一周从周日开始，周六结束，只要本周跨年，那么这周就算入下一年。

（@"yyyy-MM-dd hh:mm:ss"）
hh与HH的区别:分别表示12小时制,24小时制

G:　公元时代，例如AD公元
yy:　年的后2位
yyyy:　完整年
MM:　月，显示为1-12,带前置0
MMM:　月，显示为英文月份简写,如 Jan
MMMM:　月，显示为英文月份全称，如 Janualy
dd:　日，2位数表示，如02

d:　日，1-2位显示，如2，无前置0
EEE:　简写星期几，如Sun
EEEE:　全写星期几，如Sunday
aa:　上下午，AM/PM
H:　时，24小时制，0-23
HH:　时，24小时制，带前置0
h:　时，12小时制，无前置0
hh:　时，12小时制，带前置0
m:　分，1-2位
mm:　分，2位，带前置0
s:　秒，1-2位
ss:　秒，2位，带前置0
S:　毫秒
Z:　GMT（时区）

作者：icefishlily
来源：CSDN
原文：https://blog.csdn.net/icefishlily/article/details/79635716
版权声明：本文为博主原创文章，转载请附上博文链接！

NSDateFormatter 'YYYY' 和 'yyyy' 的区别

2014年12月29日 23:52:44 Hi_Aaron 阅读数：7061 标签：NSDateFormatter YYYY和yyyy的区别 ios 更多
个人分类：iOS

今天刷微博的时候看到这个：

V2EX › 分享发现

"YYYY-MM-dd"

glasslion · 6 小时 7 分钟前 · 583 次点击

今天有大量Twitter用户被强制登出， 并无法再次登录。
有人抓包后发现， Twitter 的 Server 认为今天是 2015 年 12月 29日。

https://twitter.com/_Ninji/status/549365454322802688

很有可能是因为某个无证程序员没搞明 Year yyyy）和 Week year(YYYY) 的区别， 一年里的绝大部分日子也无法重现这个bug， 于是乎就悲剧了。

立马就想到了**BK**项目今天**feedback**回来的**bug**，运行了大半年好好的都没有啥问题，今天突然间黑屏了，**debug**后才发现，**NSDateFormatter**的时候时间变成**2015**年了，擦，今天还是**2014**年啊亲！所以导致**create preform**的时候出现了问题。

Google了之后才发现是在格式化的时候由**YYYY**导致引起的，改回**yyyy**就好了。具体区别：

14

Also when using a date format string using the correct format is important.

@"YYYY" is week-based calendar year.

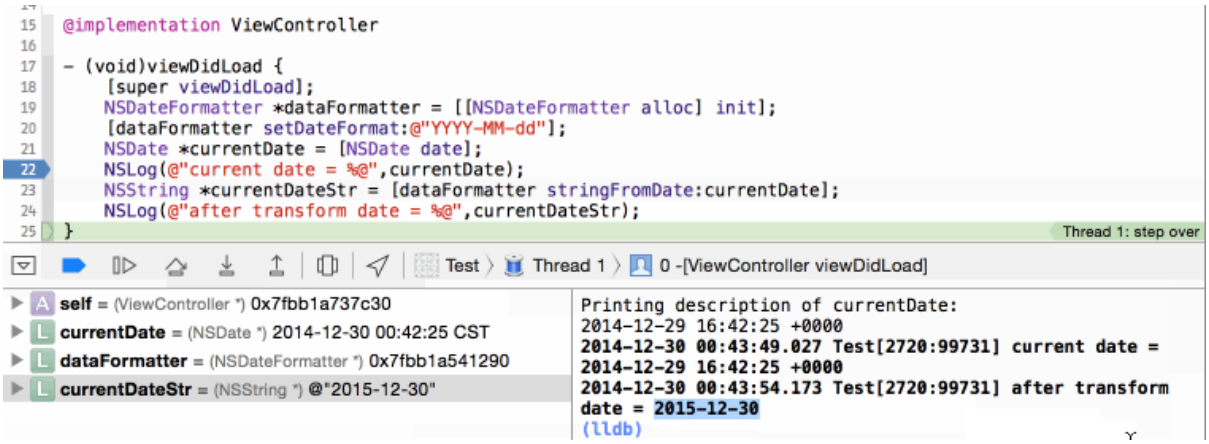
@"yyyy" is ordinary calendar year.

You can go through the whole blog, its a good to give it a look

http://realmacsoftware.com/blog/working-with-date-and-time

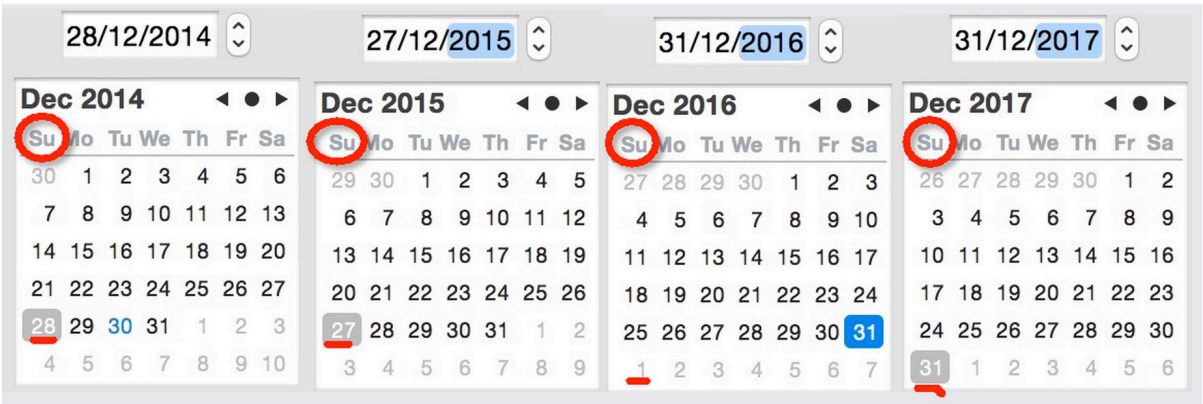
做了个简单的测试，不但认证了上面的那个问，还发现了另外一个问题：

先看一下测试：



用[NSDate date]获取当前日期是正确的：2014-12-30 00:42:25 CST （看左边debug框），但打印出来的却是2014-12-29 16:42:25 +0000 足足相差了16 个小时（看右边的debug框），打印出来的好像是美国的时间。经过@“YYYY-MM-dd”格式化当前时间后发现的时间是2015-12-30，相差了一年，这一点确实认证了@"YYYY-MM-dd"是以周计算的。

如下图，2014年的时候是在周日28号的时候出现转换后多一年的情况28，29，30，31都会；2015年是在周日27号之后出现转换多一年的情况；2016年刚好周日是2017年的1号，也就是说以周转换计算的最后一天刚好是下一年的第一天，所以不会出现转换错误的现象；2017年的周日31号又出现了转换多一年的情况。。。。以此便可看出端倪来了，具体还是自己去想吧，**只是有一点要特别注意：以后转换日期格式的时候记得还是用@“yyyy-MM-dd”这种格式吧，避免出现这种情况衍生的bug。**



参考链接：

http://stackoverflow.com/questions/15133549/difference-between-yyyy-and-yyyy-in-nsdateformatter

http://realmacsoftware.com/blog/working-with-date-and-time