

Date, Time & Encoding

---- by *Hatter Jiang*

Agenda

- 时间
 - 绝对时间、时间感知与表达
 - 格林尼治时间
 - 冬令时与夏令时
- 编码
 - ASCII
 - GB2313/GBK/GB18030
 - Unicode
 - UTF8/UTF16/UTF32

基于公开资料整理，仅供内部分享



时间

时间是物理学中的七个基本物理量之一，符号t。在国际单位制（SI）中，时间的基本单位是秒，符号s。

对秒的定义：铯-133 的原子基态的两个超精细能阶间跃迁对应辐射的 9,192,631,770 个周期的持续时间。

--- 1967年召开的第13届国际度量衡大会

一般程序中的绝对时间表示是和 Thu, 01 Jan 1970 00:00:00 GMT 的差值，单位是毫秒。

时区&当地时间

区时：一种按全球统一的时区系统计量的时间。每当太阳当头照的时候，就是中午12点钟。但不同地方看到太阳当头照的时间是不一样的。

例如，上海已是中午12点时，莫斯科的居民还要经过5个小时才能看到太阳当头照；而澳大利亚的悉尼人早已是下午2点钟了。

所以如果各地方都使用当地的时间标准，将会给行政管理、交通运输、以及日常生活等带来很多不便。

为了克服这个困难，天文学家就商量出一个解决的办法：将全世界经度每相隔15度划一个区域，这样一共有24个区域。在每个区域内都采用统一的时间标准，称为“区时”。而相邻区域的区时则相差1个小时。

当人们向东从一个区域到相邻的区域时，就将自己的钟表拨快1小时。走过几个区域就拨快几个小时。相反当人们向西从一个区域到相邻的区域时，就将自己的钟表拨慢1小时。走过几个区域就拨慢几个小时。

UTC、GMT

UTC - Coordinated Universal Time 协调世界时（英：Coordinated Universal Time，法：Temps Universel Coordonné），又称世界统一时间，世界标准时间，国际协调时间。英文（CUT）和法文（TUC）的缩写不同，作为妥协，简称UTC。

而协调世界时（UTC）是基于原子物理学的特性，将在海平面上实现的“原子时秒”定义为国际标准时的时间单位。

GMT - Greenwich Mean Time

格林尼治标准时（GMT）是格林尼治天文台通过天文学观测将每日太阳穿过本初子午线的瞬间定为正午时刻，并以此来制定时间，所以格林尼治标准时是“天文学时间”。

闰秒

国际原子时的准确度为每日数纳秒，而世界时的准确度为每日数毫秒。

对于这种情况，一种称为协调世界时的折衷时标于1972年面世。

为确保协调世界时与世界时相差不会超过0.9秒，在有需要的情况下会在协调世界时内加上正或负闰秒。

因此协调世界时与国际原子时之间会出现若干整数秒的差别。

位于巴黎的国际地球自转事务中央局负责决定何时加入闰秒。

时间表示的国际规范

ISO8601 - https://en.wikipedia.org/wiki/ISO_8601 年为4位数字，如公元1年为0001,公元前1年为0000,公元前2年为-0001。月为2位数字，如01。日为2位数字，如31。

RFC5322 The time-of-day MUST be in the range 00:00:00 through 23:59:60 (the number of seconds allowing for a leap second)

GB/T 7408-2005

Date:

2016-06-04

Date & Time:

2016-06-04T08:57:06+00:00

2016-06-04T08:57:06Z

20160604T085706Z

冬令时与夏令时

DST - Daylight Saving Time

阳光节约时，在我国称为夏时制，又称夏令时，是一种为节约能源而人为调整地方时间的制度。

实施冬令时&夏令时的时区：

中国的历史时间；

美国 EST/EDT, CST/CDT, PST/PDT;

.....

1582年10月发生了什么？

源代码:

```
SimpleDateFormat sdf = new SimpleDateFormat("yyy/MM/dd");
System.out.println(sdf.parse("1582/10/4"));
System.out.println(sdf.parse("1582/10/5"));

Date d0 = sdf.parse("1582/10/4");
Date d1 = new Date(d0.getTime() + TimeUnit.DAYS.toMillis(1L));

System.out.println(d0);
System.out.println(d1);
```

输出:

```
Thu Oct 04 00:00:00 CST 1582
Fri Oct 15 00:00:00 CST 1582
```

```
Thu Oct 04 00:00:00 CST 1582
Fri Oct 15 00:00:00 CST 1582
```

格里历与儒略历

现行公历（拉丁語：Calendarium Gregorianum，又稱格里曆），是由義大利醫生兼哲學家阿洛伊修斯·里利烏斯改革儒略曆制定的曆法，由教皇格列高利十三世在1582年10月4日頒行。

格里曆與儒略曆一樣，格里曆也是每四年在2月底置一閏日，但格里曆特別規定，除非能被400整除，所有的世紀年（能被100整除）都不設閏日；如此，每四百年，格里曆僅有97個閏年，比儒略曆減少3個閏年。格里曆的曆年平均長度為365.2425日，接近平均回歸年的365.242199074日，即約每3300年誤差一日，也更接近春分點回歸年的365.24237日，即約每8000年誤差一日；而儒略曆的曆年為365.25日，約每128年就誤差一日。到1582年時，儒略曆的春分日（3月21日）與地球公轉到春分點的實際時間已相差10天。因此，格里曆開始實行時，將儒略曆1582年10月4日星期四的次日，為格里曆1582年10月15日星期五，即有10天被刪除，但原星期的週期保持不變。格里曆的紀年沿用儒略曆，自傳統的耶穌誕生年開始，稱為「公元」，亦稱「西元」。

北京时间

CTT (CST/CDT) vs GMT+8 (GMT+0800)

CTT
PRC
Asia/Shanghai
Asia/Chongqing
Asia/Chungking
Asia/Harbin
Asia/Kashgar
Asia/Macao
Asia/Macau
Asia/Taipei
Asia/Urumqi

CST的其他含义:

Australia/South - (Australia/Adelaide)
Cuba - (America/Havana)
CST - (America/Chicago)

5分12秒之迷

```
1900/01/01 07:59:59 - 1900/01/01 08:05:43  
1900/12/31 23:59:59 - 1900/12/31 23:54:17
```

```
Transition[Overlap at 1901-01-01T00:00+08:05:43 to +08:00]
```

???

http://bugs.java.com/bugdatabase/view_bug.do?bug_id=6609459

CTT历史冬夏令时切换纪录

```
1940/06/02 23:59:59 - 1940/06/03 01:00:00
1940/09/30 23:59:59 - 1940/09/30 23:00:00
1941/03/15 23:59:59 - 1941/03/16 01:00:00
1941/09/30 23:59:59 - 1941/09/30 23:00:00
1986/05/03 23:59:59 - 1986/05/04 01:00:00
1986/09/13 23:59:59 - 1986/09/13 23:00:00
1987/04/11 23:59:59 - 1987/04/12 01:00:00
1987/09/12 23:59:59 - 1987/09/12 23:00:00
1988/04/09 23:59:59 - 1988/04/10 01:00:00
1988/09/10 23:59:59 - 1988/09/10 23:00:00
1989/04/15 23:59:59 - 1989/04/16 01:00:00
1989/09/16 23:59:59 - 1989/09/16 23:00:00
1990/04/14 23:59:59 - 1990/04/15 01:00:00
1990/09/15 23:59:59 - 1990/09/15 23:00:00
1991/04/13 23:59:59 - 1991/04/14 01:00:00
1991/09/14 23:59:59 - 1991/09/14 23:00:00
```

Encoding

- ASCII & ISO/IEC 8859-1
- GB2312 & GBK & GB18030
- Unicode
- UTF-8 & UTF-16 & UTF-32

ASCII & ISO/IEC 8859-1(Latin1)

ASCII (American Standard Code for Information Interchange, 美国标准信息交换代码) 是基于拉丁字母的一套电脑编码系统，主要用于显示现代英语和其他西欧语言。它是现今最通用的单字节编码系统，并等同于国际标准ISO/IEC 646。

ISO/IEC 8859-1编码是单字节编码，向下兼容ASCII，其编码范围是0x00-0xFF，0x00-0x7F之间完全和ASCII一致，0x80-0x9F之间是控制字符，0xA0-0xFF之间是文字符号。

GB2312 & GBK

《信息交换用汉字编码字符集》是由中国国家标准总局1980年发布，1981年5月1日开始实施的一套国家标准，标准号是GB 2312—1980。

GBK全称《汉字内码扩展规范》（GBK即“国标”、“扩展”汉语拼音的第一个字母，英文名称：Chinese Internal Code Specification），中华人民共和国全国信息技术标准化技术委员会1995年12月1日制订，国家技术监督局标准化司、电子工业部科技与质量监督司1995年12月15日联合以技监标函1995 229号文件的形式，将它确定为技术规范指导性文件。这一版的GBK规范为1.0版。

GB18030

国家标准GB18030-2005《信息技术 中文编码字符集》是我国继GB2312-1980和GB13000.1-1993之后最重要的汉字编码标准，是我国计算机系统必须遵循的基础性标准之一。GB18030有两个版本：GB18030-2000和GB18030-2005。GB18030-2000是GBK的取代版本，它的主要特点是在GBK基础上增加了CJK统一汉字扩充A的汉字。GB18030-2005的主要特点是在GB18030-2000基础上增加了CJK统一汉字扩充B的汉字。

Unicode & UCS

Unicode 1.0

1991年，不包含CJK统一汉字集的Unicode 1.0发布。随后，CJK统一汉字集的制定于1993年完成，发布了ISO 10646-1:1993，即Unicode 1.1。

从Unicode 2.0开始，Unicode采用了与ISO 10646-1相同的字库和字码；ISO也承诺，ISO 10646将不会替超出U+10FFFF的UCS-4编码赋值，以使得两者保持一致。两个项目仍都独立存在，并独立地公布各自的标准。但统一碼聯盟和ISO/IEC JTC1/SC2都同意保持两者标准的码表兼容，并紧密地共同调整任何未来的扩展。在发布的時候，Unicode一般都會採用有關字碼最常見的字型，但ISO 10646一般都盡可能採用Century字型。

<http://www.daiziyi.com/post/52.html>

Unicode字符平面

目前的Unicode字元分為17組編排，每組稱為平面（Plane），而每平面擁有65536（即 2^{16} ）個代碼點。然而目前只用了少數平面。

平面 始末字元值 中文名稱 英文名稱 0號平面 U+0000 - U+FFFF 基本多文種平面
Basic Multilingual Plane，簡稱BMP 1號平面 U+10000 - U+1FFFF 多文種補充平面
Supplementary Multilingual Plane，簡稱SMP 2號平面 U+20000 - U+2FFFF 表意文字補充平面 Supplementary Ideographic Plane，簡稱SIP 3號平面 U+30000 - U+3FFFF 表意文字第三平面（未正式使用[1]） Tertiary Ideographic Plane，簡稱TIP 4號平面 至 13號平面 U+40000 - U+DFFFF （尚未使用）
14號平面 U+E0000 - U+EFFFF 特別用途補充平面 Supplementary Special-purpose Plane，簡稱SSP 15號平面 U+F0000 - U+FFFFFF 保留作為私人使用區（A區）[2]
Private Use Area-A，簡稱PUA-A 16號平面 U+100000 - U+10FFFF 保留作為私人使用區（B區）[2] Private Use Area-B，簡稱PUA-B

特殊注意字符

U+2028

U+2029

o o O O

o U+03BF

o U+006F

O U+039F

O U+004F

<http://masatokinugawa.lo.cm/2013/09/u2028u2029.domxss.html>

<http://www.unicode.org/reports/tr36/>

BOM

UTF-8: EF BB BF

UTF-16 (大端序) : FE FF

UTF-16 (小端序) : FF FE

UTF-32 (大端序) : 00 00 FE FF

UTF-32 (小端序) : FF FE 00 00

于是IETF推出了 UTF-8 和 UTF-16 两种解决方案 (UTF-32用的太少,忽略)

实际上,IETF比较希望UTF-8成为事实标准(RFC2279),而UTF-16,也就是卖ISO和Unicode个面子,实现一下而已(RFC2781)

组合字符1

字符ò (U+00F2) 与组合ò (U+006F U+0030) , 字符ó (U+00F3) 与组合ó (U+006F U+0031) , 前者为Unicode中的基本字符, 后者则是一个基字符和一个组合字符合并而来。

ò == o + '

ó == o + '

l + J == ÿ

组合字符2

泰语组合字符 : _____(๒).....

判断UTF16字符是否为非基本平面字符

```
function isHighSurrogate(ch) {
    return (ch.charCodeAt(0) & 0xFC00) === 0xD800;
}

function isLowSurrogate(ch) {
    return (ch.charCodeAt(0) & 0xFC00) === 0xDC00;
}
```

UTF16 -> UTF8

```
var _countBits = function(_c) {
    var cnt = 0;
    while(_c > 0) { cnt++; _c = _c >>> 1; }
    return cnt;
};
function UnicodeToUtf8Bytes2(code) {
    if ((code == null) || (code < 0) || (code > (Math.pow(2, 31) -1))) {
        return ["?".charCodeAt(0)];
    }
    if (code < 0x80) { return [code]; }
    var arr = [];
    while ((code >>> 6) > 0) {
        arr.push(0x80 | (code & 0x3F));
        code = code >>> 6;
    }
    if ((arr.length + 2 + (_countBits(code))) > 8) {
        arr.push(0x80 | code);
        code = 0;
    }
    var pre = 0x80;
    for (var i = 0; i < arr.length; i++) {
        pre |= (0x80 >>> (i + 1));
    }
    arr.push(pre | code);
    return arr.reverse();
}
```

Read more ...

1. https://en.wikipedia.org/wiki/ISO_8601 - *ISO 8601*
2. http://blog.swanspace.org/ramble_unicode/

The end.