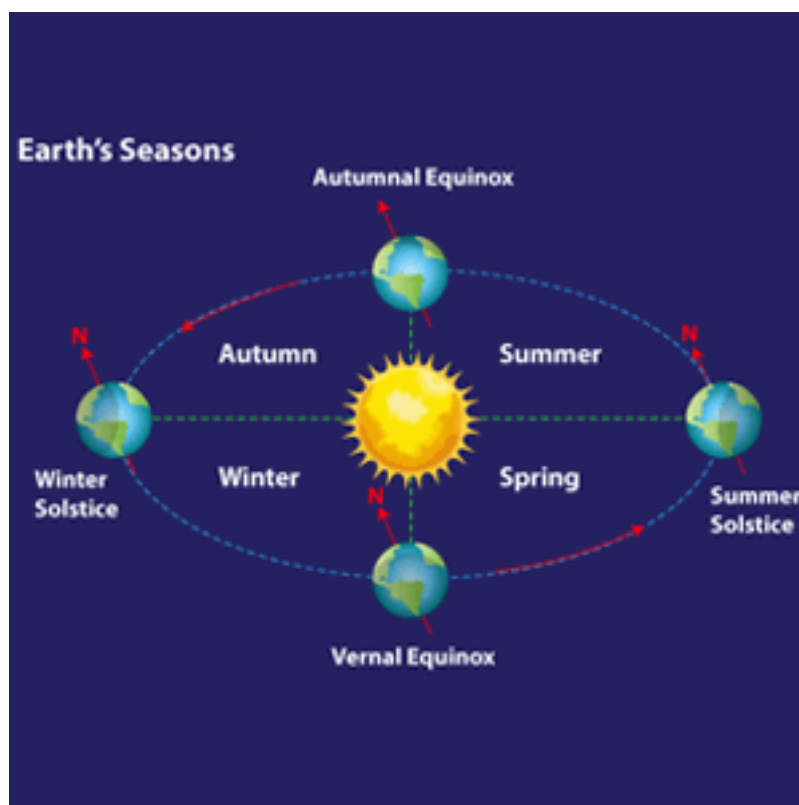


سال قبل نتیجه‌ی جستجوی من برای یافتن فرمول محاسبه‌ی زمان سال تحویل، برای ارسال ایمیل‌های خودکار تبریک آن، در سایت‌های ایرانی حاصلی نداشت. اما واژه‌ی انگلیسی Equinox سرآغازی شد برای یافتن این الگوریتم. نام علمی لحظه‌ی سال تحویل، [Vernal Equinox](#) است. Equinox به معنای نقطه‌ای است که یک فصل، به فصلی دیگر تبدیل می‌شود:



Equinox واژه‌ای است لاتین به معنای «شب‌های مساوی» و به این نکته اشاره دارد که در Equinox، طول شب و روز یکی می‌شوند. هر سال دارای دو Equinox است: vernal equinox و autumnal equinox (بهاری و پاییزی). البته باید در نظر داشت که Equinox بهاری در نیم کره‌ی شمالی بیشتر معنا پیدا می‌کند؛ زیرا در نیم کره‌ی جنوبی در همین زمان، پاییز شروع می‌شود. بنابراین می‌توان enum زیر را برای تعریف این چهار ثابت رخدادهای خورشیدی تعریف کرد:

```
public enum SunEvent
{
    /// <summary>
    /// march equinox
    /// </summary>
    VernalEquinox,

    /// <summary>
    /// june solstice
    /// </summary>
    SummerSolstice,

    /// <summary>
    /// september equinox
    /// </summary>
    AutumnalEquinox,
```

```

    /// <summary>
    /// december solstice
    /// </summary>
    WinterSolstice
}

```

در ادامه برای محاسبه‌ی زمان equinox از فصل 27 کتاب [Astronomical Algorithms](#) کمک گرفته شده و تمام اعداد و ارقام و جداولی را که ملاحظه می‌کنید از این کتاب استخراج شده‌اند.

```

/// <summary>
/// Based on Jean Meeus book _Astronomical Algorithms_
/// </summary>
public static class EquinoxCalculator
{
    /// <summary>
    /// Degrees to Radians conversion factor.
    /// </summary>
    public static readonly double Deg2Radian = Math.PI / 180.0;

    public static bool ApproxEquals(double d1, double d2)
    {
        const double epsilon = 2.2204460492503131E-16;
        if (d1 == d2)
            return true;
        var tolerance = ((Math.Abs(d1) + Math.Abs(d2)) + 10.0) * epsilon;
        var difference = d1 - d2;
        return (-tolerance < difference && tolerance > difference);
    }

    /// <summary>
    /// Calculates time of the Equinox and Solstice.
    /// </summary>
    /// <param name="year">Year to calculate for.</param>
    /// <param name="sunEvent">Event to calculate.</param>
    /// <returns>Date and time event occurs as a fractional Julian Day.</returns>
    public static DateTime GetSunEventUtc(this int year, SunEvent sunEvent)
    {
        double y;
        double julianEphemerisDay;

        if (year >= 1000)
        {
            y = (Math.Floor((double)year) - 2000) / 1000;

            switch (sunEvent)
            {
                case SunEvent.VernalEquinox:
                    julianEphemerisDay = 2451623.80984 + 365242.37404 * y + 0.05169 * (y * y) - 0.00411
                    * (y * y * y) - 0.00057 * (y * y * y * y);
                    break;
                case SunEvent.SummerSolstice:
                    julianEphemerisDay = 2451716.56767 + 365241.62603 * y + 0.00325 * (y * y) - 0.00888
                    * (y * y * y) - 0.00030 * (y * y * y * y);
                    break;
                case SunEvent.AutumnalEquinox:
                    julianEphemerisDay = 2451810.21715 + 365242.01767 * y + 0.11575 * (y * y) - 0.00337
                    * (y * y * y) - 0.00078 * (y * y * y * y);
                    break;
                case SunEvent.WinterSolstice:
                    julianEphemerisDay = 2451900.05952 + 365242.74049 * y + 0.06223 * (y * y) - 0.00823
                    * (y * y * y) - 0.00032 * (y * y * y * y);
                    break;
                default:
                    throw new NotSupportedException();
            }
        }
        else
        {
            y = Math.Floor((double)year) / 1000;

            switch (sunEvent)
            {
                case SunEvent.VernalEquinox:
                    julianEphemerisDay = 1721139.29189 + 365242.13740 * y + 0.06134 * (y * y) - 0.00111
                    * (y * y * y) - 0.00071 * (y * y * y * y);
                    break;
            }
        }
    }
}

```

```

        case SunEvent.SummerSolstice:
            julianEphemerisDay = 1721233.25401 + 365241.72562 * y + 0.05323 * (y * y) - 0.00907
* (y * y * y) - 0.00025 * (y * y * y * y);
            break;
        case SunEvent.AutumnalEquinox:
            julianEphemerisDay = 1721325.70455 + 365242.49558 * y + 0.11677 * (y * y) - 0.00297
* (y * y * y) - 0.00074 * (y * y * y * y);
            break;
        case SunEvent.WinterSolstice:
            julianEphemerisDay = 1721414.39987 + 365242.88257 * y + 0.00769 * (y * y) - 0.00933
* (y * y * y) - 0.00006 * (y * y * y * y);
            break;
        default:
            throw new NotSupportedException();
    }
}

var julianCenturies = (julianEphemerisDay - 2451545.0) / 36525;
var w = 35999.373 * julianCenturies - 2.47;
var lambda = 1 + 0.0334 * Math.Cos(w * Deg2Radian) + 0.0007 * Math.Cos(2 * w * Deg2Radian);
var sumOfPeriodicTerms = getSumOfPeriodicTerms(julianCenturies);
return JulianToUtcDate(julianEphemerisDay + (0.00001 * sumOfPeriodicTerms / lambda));
}

/// <summary>
/// Converts a fractional Julian Day to a .NET DateTime.
/// </summary>
/// <param name="julianDay">Fractional Julian Day to convert.</param>
/// <returns>Date and Time in .NET DateTime format.</returns>
public static DateTime JulianToUtcDate(double julianDay)
{
    double a;
    int month, year;

    var j = julianDay + 0.5;
    var z = Math.Floor(j);
    var f = j - z;

    if (z >= 2299161)
    {
        var alpha = Math.Floor((z - 1867216.25) / 36524.25);
        a = z + 1 + alpha - Math.Floor(alpha / 4);
    }
    else
        a = z;

    var b = a + 1524;
    var c = Math.Floor((b - 122.1) / 365.25);
    var d = Math.Floor(365.25 * c);
    var e = Math.Floor((b - d) / 30.6001);
    var day = b - d - Math.Floor(30.6001 * e) + f;

    if (e < 14)
        month = (int)(e - 1.0);
    else if (ApproxEquals(e, 14) || ApproxEquals(e, 15))
        month = (int)(e - 13.0);
    else
        throw new NotSupportedException("Illegal month calculated.");

    if (month > 2)
        year = (int)(c - 4716.0);
    else if (month == 1 || month == 2)
        year = (int)(c - 4715.0);
    else
        throw new NotSupportedException("Illegal year calculated.");

    var span = TimeSpan.FromDays(day);

    return new DateTime(year, month, (int)day, span.Hours, span.Minutes,
        span.Seconds, span.Milliseconds, new GregorianCalendar(), DateTimeKind.Utc);
}

/// <summary>

```

```

/// These values are from Table 27.C
/// </summary>
private static double getSumOfPeriodicTerms(double julianCenturies)
{
    return 485 * Math.Cos(Deg2Radian * 324.96 + Deg2Radian * (1934.136 * julianCenturies))
        + 203 * Math.Cos(Deg2Radian * 337.23 + Deg2Radian * (32964.467 * julianCenturies))
        + 199 * Math.Cos(Deg2Radian * 342.08 + Deg2Radian * (20.186 * julianCenturies))
        + 182 * Math.Cos(Deg2Radian * 27.85 + Deg2Radian * (445267.112 * julianCenturies))
        + 156 * Math.Cos(Deg2Radian * 73.14 + Deg2Radian * (45036.886 * julianCenturies))
        + 136 * Math.Cos(Deg2Radian * 171.52 + Deg2Radian * (22518.443 * julianCenturies))
        + 77 * Math.Cos(Deg2Radian * 222.54 + Deg2Radian * (65928.934 * julianCenturies))
        + 74 * Math.Cos(Deg2Radian * 296.72 + Deg2Radian * (3034.906 * julianCenturies))
        + 70 * Math.Cos(Deg2Radian * 243.58 + Deg2Radian * (9037.513 * julianCenturies))
        + 58 * Math.Cos(Deg2Radian * 119.81 + Deg2Radian * (33718.147 * julianCenturies))
        + 52 * Math.Cos(Deg2Radian * 297.17 + Deg2Radian * (150.678 * julianCenturies))
        + 50 * Math.Cos(Deg2Radian * 21.02 + Deg2Radian * (2281.226 * julianCenturies))
        + 45 * Math.Cos(Deg2Radian * 247.54 + Deg2Radian * (29929.562 * julianCenturies))
        + 44 * Math.Cos(Deg2Radian * 325.15 + Deg2Radian * (31555.956 * julianCenturies))
        + 29 * Math.Cos(Deg2Radian * 60.93 + Deg2Radian * (4443.417 * julianCenturies))
        + 28 * Math.Cos(Deg2Radian * 155.12 + Deg2Radian * (67555.328 * julianCenturies))
        + 17 * Math.Cos(Deg2Radian * 288.79 + Deg2Radian * (4562.452 * julianCenturies))
        + 16 * Math.Cos(Deg2Radian * 198.04 + Deg2Radian * (62894.029 * julianCenturies))
        + 14 * Math.Cos(Deg2Radian * 199.76 + Deg2Radian * (31436.921 * julianCenturies))
        + 12 * Math.Cos(Deg2Radian * 95.39 + Deg2Radian * (14577.848 * julianCenturies))
        + 12 * Math.Cos(Deg2Radian * 287.11 + Deg2Radian * (31931.756 * julianCenturies))
        + 12 * Math.Cos(Deg2Radian * 320.81 + Deg2Radian * (34777.259 * julianCenturies))
        + 9 * Math.Cos(Deg2Radian * 227.73 + Deg2Radian * (1222.114 * julianCenturies))
        + 8 * Math.Cos(Deg2Radian * 15.45 + Deg2Radian * (16859.074 * julianCenturies));
}
}

```

خروجی‌های زمانی ستاره شناسی، عموماً بر اساس فرمت Julian Date است که آغاز آن 4713BCE January 1, 12 hours GMT است. به همین جهت در انتهای این مباحث، تبدیل Julian Date به DateTime دات نت را نیز ملاحظه می‌کنید. همچنین باید دقت داشت که خروجی نهایی بر اساس UTC است و برای زمان ایران، باید 3.5 ساعت به آن اضافه شود.

خروجی این الگوریتم را برای سال‌های 2014 تا 2022 به صورت ذیل مشاهده می‌کنید:

```

2014 -> 1392/12/29 20:28:08
2015 -> 1394/01/01 02:16:29
2016 -> 1395/01/01 08:01:21
2017 -> 1395/12/30 14:00:00
2018 -> 1396/12/29 19:46:10
2019 -> 1398/01/01 01:29:29
2020 -> 1399/01/01 07:21:03
2021 -> 1399/12/30 13:08:41
2022 -> 1400/12/29 19:04:37

```

برای نمونه زمان محاسبه شده‌ی 02:16:29 01/01/1394 با زمان رسمی اعلام شده‌ی ساعت 2 و 15 دقیقه و 10 ثانیه روز شنبه 1 فروردین 1394 و یا برای سال 93 زمان محاسبه شده‌ی 20:28:08 29/12/1392 با زمان رسمی ساعت ۲۰ و ۲۷ دقیقه و ۷ ثانیه پنجشنبه ۲۹ اسفند ۱۳۹۲، تقریباً برابری می‌کند.

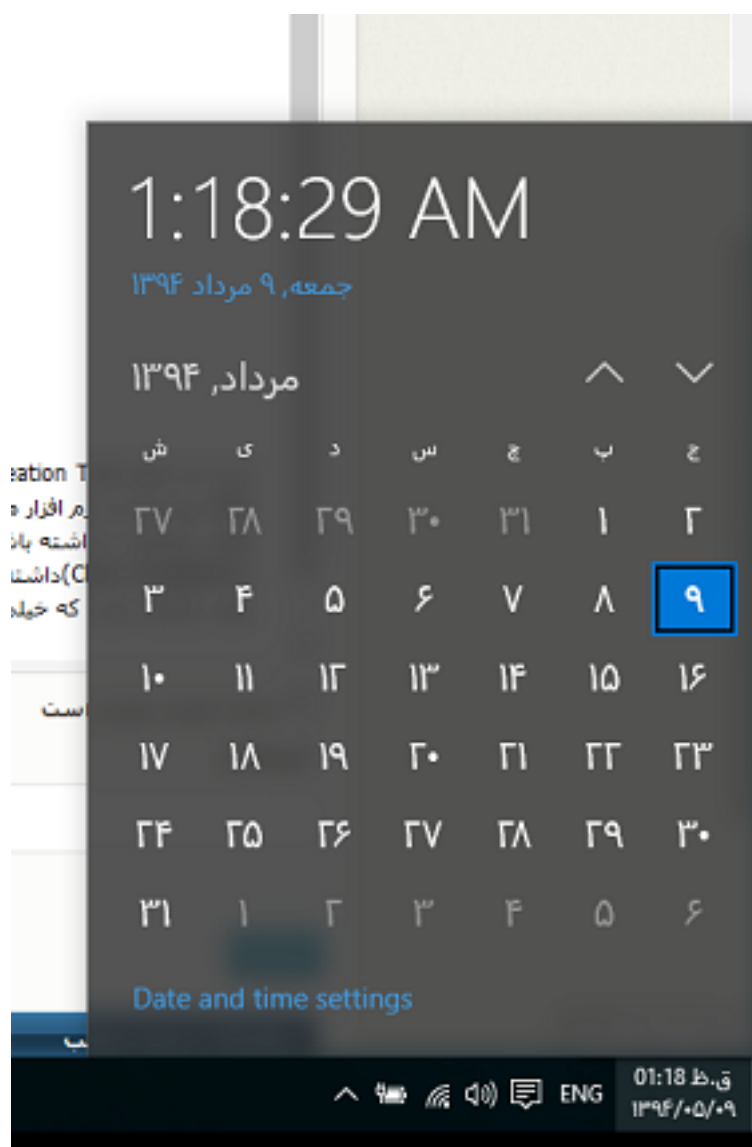
کدهای کامل این پروژه را از اینجا می‌توانید دریافت کنید

[Equinox.zip](#)

امروز بعد از چندین سال، شاید بعد از 5 سال، ویندوز 7 نسخه Home Premium را به Windows 10 Home ارتقاء دادم. واقعا این روزها دیگر ویندوز 7 در انجام کارها یاری نمی‌کرد و بصورت مداوم خطای صفحه‌ی آبی را نمایش می‌داد. ولی در حین گشت و گذار وب بودم که بصورت اتفاقی به [این لینک](#) برخوردم. بعد از مطالعه‌ی لینک و مراجعه به [لینک اصلی](#) متوجه شدم مایکروسافت این امکان را به کاربران نسخه اصلی ویندوزهای قبلی داده است که بتوانند بصورت رایگان ویندوزشان را به ویندوزی معادل، بشرح جدول زیر ارتقا دهند و خوشبختانه بدون کمترین زحمت و مشقتی توانستم یک نسخه‌ی پاک و به روز را با IP ایرانی از سایت مایکروسافت بشرح زیر دریافت کنم و تجربه‌ی جدیدی داشته باشم:

از نسخه	به نسخه
Windows 7 Starter	Windows 10 Home
Windows 7 Home Basic	
Windows 7 Home Premium	
Windows 7 Professional	Windows 10 Pro
Windows 7 Ultimate	
Windows 8.1	Windows 10 Home
Windows 8.1 pro	Windows 10 Pro
Windows pro For Student	
Windows Phone 8.1	Windows 10 Mobile

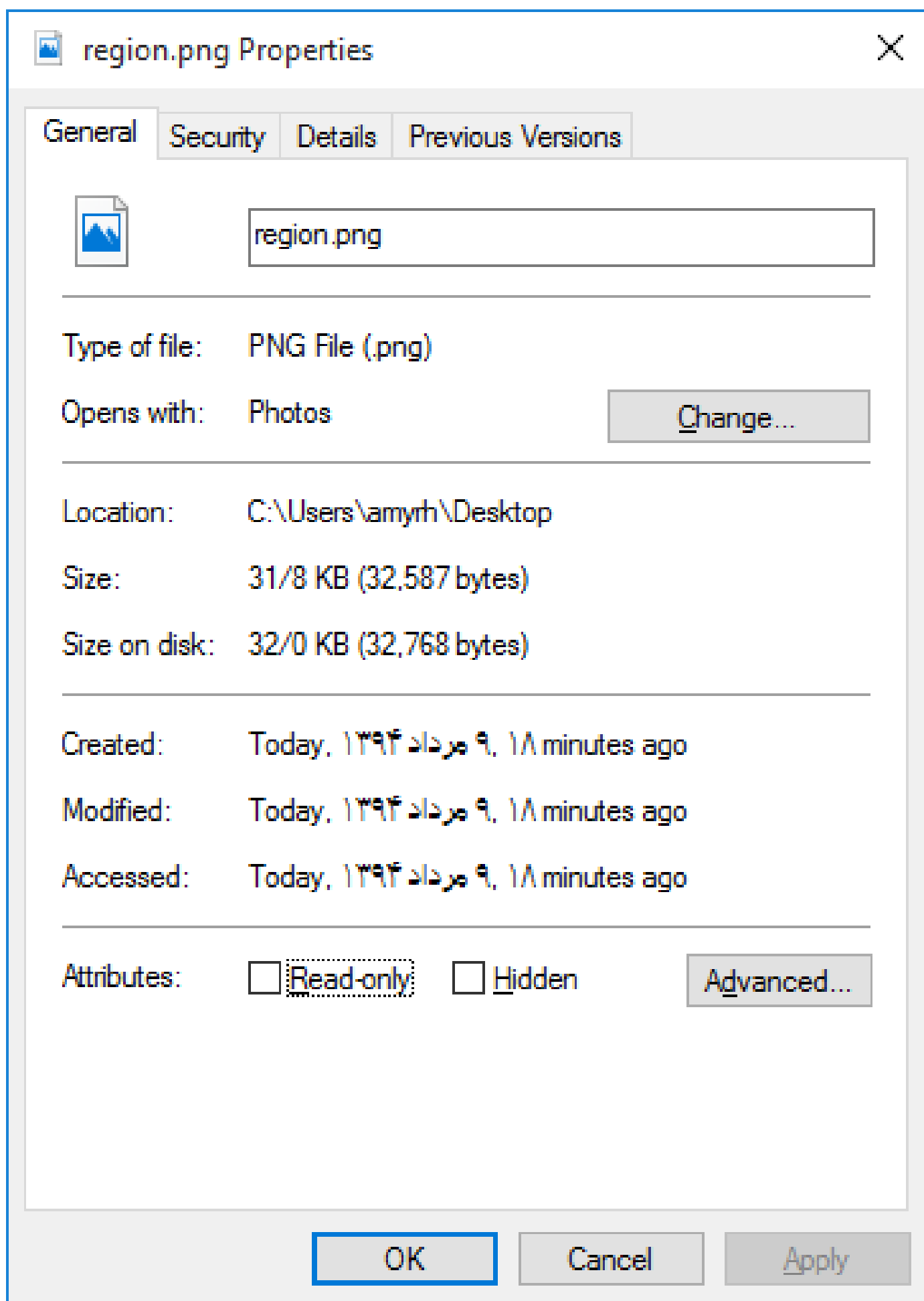
ابتدا باید فایل Media Creation Tool نسخه‌ی 64بیتی را دانلود کنید. بوسیله‌ی این نرم افزار می‌توانید نسخه‌ی ISO یا نسخه‌ی برخط و آنلاین را دریافت کنید. بعد از دریافت فایل ISO، بوسیله‌ی یه نرم افزار مانند [Rufus](#) فایل ISO را می‌تونید به یک فلش Bootable تبدیل کنید؛ یا اینکه بر روی DVD رایت کنید. در صورتیکه قصد ارتقاء نسخه‌ی اصلی ویندوز فعلی خودتان را داشته باشید، نصاب Media Creation Tool از شما شماره‌ی سریال نرم افزار را درخواست نمی‌کند. در غیر اینصورت اگر قصد داشته باشید یک نصب از ابتدا (Clean Installation) را داشته باشید، باید شماره سریال معتبر محصول قبلی را جهت فعالسازی وارد نمایید. روال و فرآیند نصب که خیلی سهل و آسان است و نیازی به توضیح ندارد. ولی یک امکان عالی که به نسخه‌ی جدید ویندوز اضافه شده‌است، پشتیبانی از تقویم فارسی هست. همانطور که مایکروسافت وعده‌ی آن را داده بود:



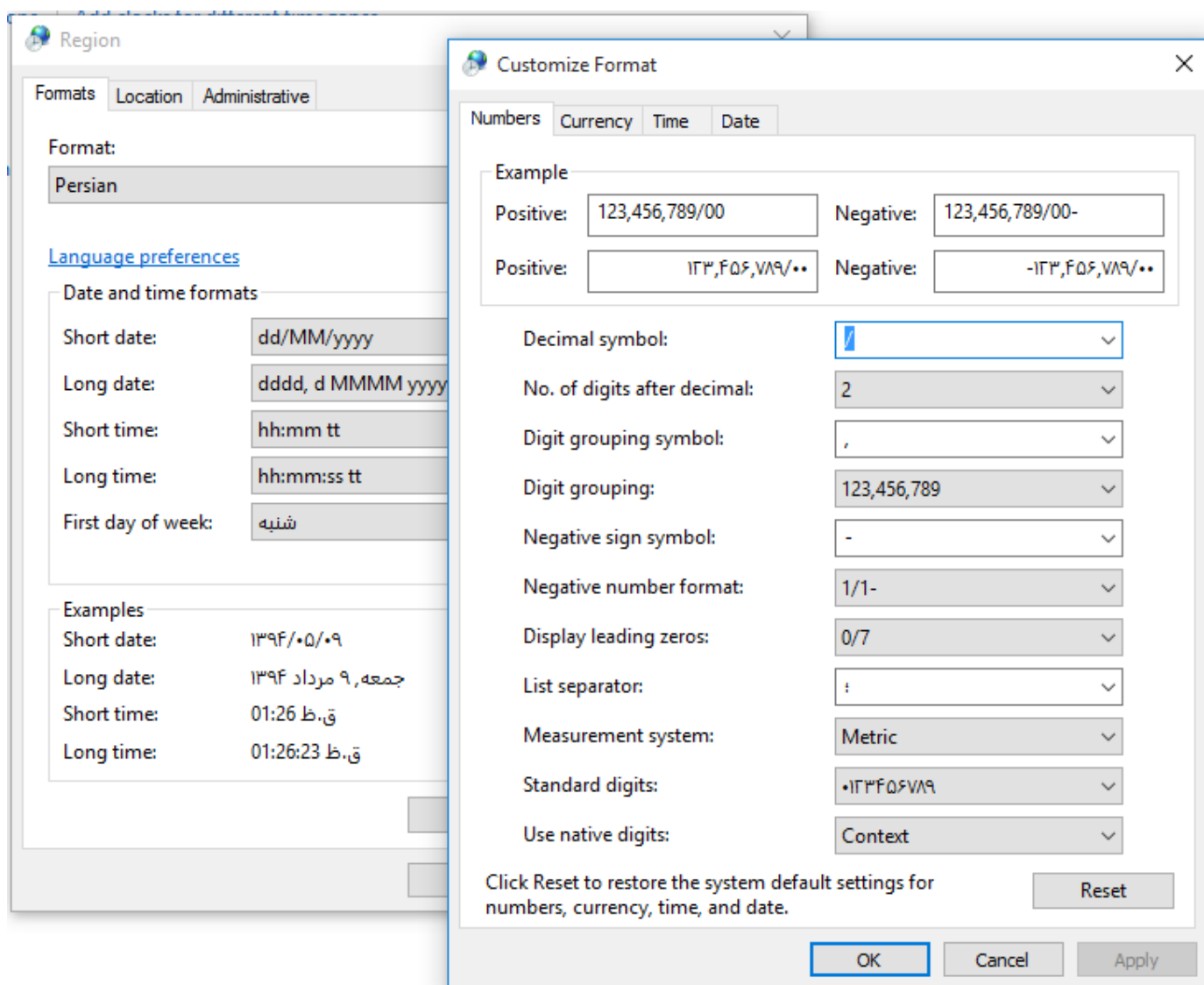
این مورد یکی از مهم‌ترین تغییرات لااقل برای ما ایرانی‌ها است. بعبارت دیگر در هر جا که تاریخ میلادی وجود داشته باشد، به تاریخ شمسی تبدیل خواهد شد. به عنوان مثال امکان مرتب سازی بر اساس تاریخ شمسی بی نقص امکان پذیر است:

This PC > Portable Data (G:) > Software		
<input type="checkbox"/> Name	Date modified	Type
Gerdoo	۱۳۹۰/۰۵/۰۱ ب.ظ ۰۹:۱۲	File folder
7 Theme	۱۳۹۰/۰۹/۱۱ ب.ظ ۰۸:۴۵	File folder
OALD8	۱۳۹۱/۰۵/۲۶ ب ۰۷:۳۴...	File folder
HJSplit v3.0	۱۳۹۱/۰۵/۲۶ ب ۰۷:۳۴...	File folder
Plotter	۱۳۹۱/۰۵/۲۶ ب ۰۷:۳۴...	File folder
RSS Readers	۱۳۹۲/۰۲/۰۶ ب ۰۷:۰۹...	File folder
Gerdoo Winter 2013	۱۳۹۲/۰۶/۱۵ ب ۰۷:۱۶...	File folder
KING Parand 2013	۱۳۹۲/۰۸/۳۰ ق ۰۹:۲۳...	File folder

و یا بعنوان مثال دیگر تاریخ خصیصه‌ها به فرمت تاریخ شمسی نمایش داده می‌شود.



و مانند سایر تقویم‌ها امکان سفارش نمودن آن وجود دارد.



نظرات خوانندگان

نویسنده: محسن خان
تاریخ: ۱۳۹۴/۰۵/۱۱ ۹:۳۱

تقویم خود ویندوز هم هست:

