

# Las fechas

Con PowerShell, la obtención de la fecha y hora se realiza con el commandlet `Get-Date`. Si bien es cierto que un simple `Get-Date` en la consola le muestra la hora y fecha actual, esta fecha puede mostrarse en un buen número de formatos (véase Las fechas - Los formatos, de este capítulo).

Una variable que contiene una fecha es de tipo `DateTime`. Para verificarlo usted mismo, escriba el comando siguiente:

```
PS > (Get-Date).GetType()


IsPublic IsSerial Name          BaseType
-----
True     True     DateTime    System.ValueType
```


Los objetos de tipo `DateTime` ocultan muchos métodos interesantes, como el método `IsDaylightSavingTime`, que nos indica si la hora actual está ajustada a la hora de verano o a la hora de invierno.

Para conocer las posibles acciones sobre las fechas, mejor aún es hacer un `Get-Member` en el objeto devuelto por `Get-Date`:

```
PS > Get-Date | Get-Member
```

Cuando se habla de la fecha o del tiempo, es necesario definir lo que llamamos una unidad de tiempo. Y la unidad más baja que existe en el sistema se llama «tick». Un tick es una unidad de medida de tiempo. Corresponde a un latido del corazón del ordenador, es decir, a un periodo del Timer (el Timer es un componente electrónico que gestiona el tiempo). Este valor es actualmente diez millonésimas de segundo.

 Para conocer el número de ticks presentes en un segundo, escriba el comando siguiente: `PS > $((Get-Date).ticks) - $((Get-Date).Addseconds(-1).ticks)9990000`

 Aproximadamente 10 millones contando el tiempo de tratamiento del comando.

## 1. Métodos de manipulación de los objetos DateTime

Gracias al commandlet `Get-Member` aplicado a un objeto de tipo `DateTime` ha podido ver una lista considerable de métodos. La tabla siguiente recoge los métodos más comunes con una breve descripción.

Método	Descripción
Add y toda la familia de los Add-* AddDays AddHours AddMilliseconds AddMinutes AddMonths AddSeconds AddTicks AddYears	Añade o resta una o varias unidades de tiempo al objeto fecha. Añade si el valor anterior pasado como argumento es positivo, resta si el valor anterior pasado es negativo (véase Las fechas - Manipulación de las fechas, de este capítulo).
CompareTo	Compara una fecha a otra. Los valores devueltos son los siguientes:

	<p>-1: si la fecha es anterior a aquélla con la que se la compara.</p> <p>1: si es posterior.</p> <p>0: si son iguales.</p>
Equals	<p>Devuelve un indicador booleano de comparación.</p> <p><b>True:</b> si las dos fechas son idénticas.</p> <p><b>False:</b> en el caso contrario.</p>
GetDateTimeFormats	Devuelve todos los formatos disponibles para el objeto DateTime.
GetHashCode	Devuelve el código de hash de la variable.
GetType	Devuelve el tipo de la variable. En el caso de una fecha, se trata del tipo DateTime.
GetTypeCode	Devuelve el código asociado al tipo.
<p>La familia de los Get-*</p> <p>Get_Date</p> <p>Get_Day</p> <p>Get_DayOfWeek</p> <p>Get_DayOfYear</p> <p>Get_Hour</p> <p>Get_Millisecond</p> <p>Get_Minute</p> <p>Get_Month</p> <p>Get_Second</p> <p>Get_Ticks</p> <p>Get_TimeOfDay</p> <p>Get_Year</p>	<p>Los métodos Get-* devuelven el parámetro de la fecha en cuestión. Ejemplo: el método Get_DayOfWeek devuelve una variable que contiene el día de la semana correspondiente.</p>
Get_Kind	Devuelve el tipo de variable.
IsDayLightSavingTime	Devuelve un valor booleano que indica si la hora actual está ajustada a la hora de verano o a la hora de invierno.
Subtract	Sustraer una fecha del objeto.
ToBinary	Devuelve el valor de la fecha en formato binario.
ToFileTime	Devuelve el valor de objeto DateTime en curso, en hora de archivo Windows.
ToFileTimeUtc	Devuelve el valor de objeto DateTime en curso, hora de archivo Windows. (Hora Universal).
ToLocalTime	Devuelve el valor de objeto DateTime en curso, en hora local.
ToLongDateString	Devuelve una cadena de caracteres que contiene la fecha en formato largo.

ToLongTimeString	Devuelve una cadena de caracteres que contiene la hora en formato largo.
ToOADate	Devuelve la fecha en formato OLE ( <i>Object Linking and Embedding</i> ) automation (número flotante). El formato OLE automation corresponde al número de días desde el 30 de diciembre de 1899 a medianoche.
ToShortDateString	Devuelve una cadena de caracteres que contiene la fecha en formato corto.
ToShortTimeString	Devuelve una cadena de caracteres que contiene la hora en formato corto.
ToString	Devuelve una cadena de caracteres que contiene la fecha y la hora en formato estándar.
ToUniversalTime	Devuelve la fecha y la hora en formato estándar.

### Ejemplo:

*Recuperación de los minutos en la hora actual.*

```
PS > Get-Date
jueves 8 octubre 2009 22:36:16

PS > (Get-Date).Get_minute()
36
```

## 2. Los formatos

Elegir un formato no es necesariamente sencillo, sobre todo cuando existen unos 60 formatos llamados «estándares».

Pues sí, para una sola fecha existen muchas distintas maneras de escribirla. Para comprobarlo, pruebe el comando siguiente:

```
PS > (Get-Date).GetDateTimeFormats() | Sort-Object -Unique
```

```
08.10.09                | 22:40
08.10.09 22 h 40        | 22:40:29
08.10.09 22.40          | 8 oct. 09
08.10.09 22:40          | 8 oct. 09 20 h 40
08.10.09 22:40:29       | 8 oct. 09 20.40
08/10/09                | 8 oct. 09 20:40:29
08/10/09 22 h 40        | 8 oct. 09 22 h 40
08/10/09 22.40          | 8 oct. 09 22.40
08/10/09 22:40          | 8 oct. 09 22:40
08/10/09 22:40:29       | 8 oct. 09 22:40:29
08/10/2009              | 8 octubre
08/10/2009 22 h 40      | 8 octubre 2009
08/10/2009 22.40        | 8 octubre 2009 20 h 40
08/10/2009 22:40        | 8 octubre 2009 20.40
08/10/2009 22:40:29     | 8 octubre 2009 20:40:29
08-10-09                | 8 octubre 2009 22 h 40
```

08-10-09 22 h 40	8 octubre 2009 22.40
08-10-09 22.40	8 octubre 2009 22:40
08-10-09 22:40	8 octubre 2009 22:40:29
08-10-09 22:40:29	jueves 8 octubre 2009
2009-10-08	jueves 8 octubre 2009 20 h 40
2009-10-08 22 h 40	jueves 8 octubre 2009 20.40
2009-10-08 22.40	jueves 8 octubre 2009 20:40:29
2009-10-08 22:40	jueves 8 octubre 2009 22 h 40
2009-10-08 22:40:29	jueves 8 octubre 2009 22.40
2009-10-08 22:40:29Z	jueves 8 octubre 2009 22:40
2009-10-08T22:40:29	jueves 8 octubre 2009 22:40:29
2009-10-08T22:40:29.6675819+02:00	octubre 2009
22 h 40	Thu, 08 Oct 2009 22:40:29 GMT

3. Los formatos estándar

Para ayudarle en la elección del formato, la tabla siguiente lista los formatos estándar aplicables a los valores `DateTime`.

Formato	Descripción
d	Formato de fecha corta.
D	Formato de fecha larga.
f	Formato de fecha larga y hora abreviada.
F	Formato de fecha larga y hora completa.
g	Formato de fecha corta y hora abreviada.
G	Formato de fecha corta y hora completa.
m,M	Formato de mes y día: "dd MMMM".
r,R	Formato de fecha y hora basada en la especificación de la RFC 1123.
s	Formato de fecha y hora escogida.
t	Formato de hora abreviada.
T	Formato de hora completa.
u	Formato de fecha y hora universal (indicador de tiempo universal: "Z").
U	Formato de fecha larga y hora completa con tiempo universal.
y,Y	Formato de año y mes.

He aquí algunos ejemplos de la aplicación de los diversos formatos.

Ejemplos:

Si desea devolver una fecha en formato estándar tal como se define en la RFC 1123, el comando será el siguiente:

```
PS > Get-Date -Format r
Sun, 20 Sep 2009 12:48:53 GMT
```

*Si desea devolver una fecha en formato fecha corta y hora completa tal como se define en la RFC 1123, el comando será el siguiente:*

```
PS > Get-Date -Format G
20/09/2009 12:49:04
```

*Si desea devolver una fecha en formato fecha larga y hora completa tal como se define en la RFC 1123, el comando será el siguiente:*

```
PS > Get-Date -Format F
domingo 20 septiembre 2009 12:49:30
```

## 4. Los formatos personalizados


Naturalmente la visualización de la fecha no se limita a los formatos estándar. También son posibles los formatos personalizados.

Y para ello, debe utilizar el parámetro `-Format` asociado a especificadores de formato. La diferencia con los formatos estándar enunciados anteriormente, reside en el hecho de que los formatos personalizados son elementos combinables en una cadena de caracteres, por ejemplo. Mientras que los formatos estándar sólo tienen sentido si no están combinados.

He aquí la lista no exhaustiva de los formatos personalizados:

Formato	Descripción
d	Representación del día por un número comprendido entre: 1-31.
dd	Representación del día por un número comprendido entre: 01-31. La diferencia con el formato «d» es la inserción de un cero no significativo para los números que van de 1 a 9.
ddd	Representación del día en formato de nombre abreviado. Ejemplo: Lun., Mar., Mie., etc.
dddd	Representación del día en formato de nombre completo.
f	Representación de la cifra más significativa de la fracción de segundo.
ff	Representación de las dos cifras más significativas de la fracción de segundo.
fff	Representación de las tres cifras más significativas de la fracción de segundo.
ffff	Representación de las cuatro cifras más significativas de la fracción de segundo.
h	Representación de la hora por su número. Número comprendidos entre: 1-12.
hh	Representación de la hora por un número con inclusión de un cero no significativo para los números de 1 a 9. Números comprendidos entre: 01-12.
H	Representación de la hora por un número. Números comprendidos entre: 0-23.
HH	Representación de la hora por un número con inclusión de un cero no significativo para los números de 0 a 9. Números comprendidos entre: 00-23.

m	Representación de los minutos por un número. Números comprendidos entre: 0-59.
mm	Representación de los minutos por un número con inclusión de un cero no significativo para los números de 0 a 9. Números comprendidos entre: 00-59.
M	Representación del mes por un número. Números comprendidos entre: 1-12.
MM	Representación del mes por un número con inclusión de un cero no significativo para los números de 1 a 9. Números comprendidos entre: 01-12.
MMM	Representación del mes en formato de su nombre abreviado.
MMMM	Representación del mes en formato de su nombre completo.
y	Representación del año en formato de un número de dos cifras, o más. Si el año contiene más de dos cifras, sólo las dos cifras de menor peso aparecen en el resultado y si contiene menos, únicamente la o las cifras (sin cero significativo) aparecen.
yy	Como el caso anterior, pero con la diferencia que si el año contiene menos de dos cifras, el número se rellenará con ceros no significativos para alcanzar las dos cifras.
yyy	Representación del año en formato de un número de tres cifras. Si el año contiene más de tres cifras, sólo las tres cifras de menor peso aparecen en el resultado. Si el año contiene menos de tres cifras, el número se rellenará con ceros no significativas para alcanzar tres cifras.
yyyy	Representación del año en formato de un número de cuatro cifras. Si el año contiene más de cuatro cifras, sólo las cuatro cifras de menor peso aparecen en el resultado. Si el año contiene menos de cuatro cifras, el número se rellenará con ceros no significativas para alcanzar cuatro cifras.

 Para obtener una lista completa de esos especificadores de formato, acuda al sitio MSDN de Microsoft: [http://msdn.microsoft.com/es-es/library/8kb3ddd4\(v=VS.80\).aspx](http://msdn.microsoft.com/es-es/library/8kb3ddd4(v=VS.80).aspx)

#### Ejemplo:

En el primer ejemplo, deseamos simplemente mostrar la fecha en el formato siguiente:

```
<Nombre del Día><Número del día> <Mes> <Año> ---- <Hora>:<Minuto>:<Segundo>
```

```
PS > Get-Date -Format 'dddd dd MMMM yyyy ---- HH:mm:ss '
domingo 20 septiembre 2009 ---- 12:50:16
```

#### Ejemplo:

Supongamos que necesita generar informes cuyo nombre de archivo corresponda a la fecha en que se ha generado.

Para ello nada más fácil que...

```
PS > New-Item -Type file -Name "Informe_$((Get-Date) -Format 'dd-MM-yyyy')).txt"
```

#### Resultado:

```
PS > New-Item -Type File -Name "Informe_$((Get-Date) -Format 'dd-MM-yyyy')).txt"
```

Directorio: C:\Temp

Mode	LastWriteTime	Length	Name
----	-----	-----	----
-a---	20/09/2009 12:51	0	Informe_20-09-2009.txt

Existe un último modo de visualización. Éste último se llama visualización en modo «Unix».

Como puede imaginar, este modo recupera en formato Unix las propiedades del objeto `DateTime` que usted especifique. He aquí lo esencial de estos especificadores:

Formato	Descripción
%m	Mes del año (01-12).
%d	Día del mes (01-31).
%y	Año, únicamente las dos últimas cifras (00-99).
%Y	Año con cuatro cifras
%D	Vista en formato mm/dd/yy.
%H	Horas (00-23).
%M	Minutos (00-59).
%S	Segundos (00-59).
%T	Hora en formato HH:MM:SS.
%J	Día del año (1-366).
%w	Día de la semana (0-6) con Sábado = 0.
%a	Abreviación del día (lun., mar., etc.).
%h	Abreviación del mes (Feb., Jul., etc.).
%r	Hora en formato HH:MM:SS con HH (0-12).
%n	Nueva línea.
%t	Tabulación.

#### Ejemplo:

*Visualización en formato Unix de la fecha actual.*

```
PS > Get-Date -Uformat 'Estamos a %a %d %Y, y son las %T'
```

```
Estamos a sáb 29 2010, y son las 17:20:36
```

## 5. Manipulación de las fechas

### a. Crear una fecha

Existen varias maneras de crear una fecha en PowerShell. La más corriente es utilizar el comando `Get-Date`. Utilizado sin parámetro, este comando devuelve la fecha y la hora. Si queremos crear una variable `DateTime` con una fecha de nuestra elección, debemos especificarla gracias a los parámetros: `-Year`, `-Month`, `-Day`, `-Hour`, `-Minute`, `-Second`.

Ejemplo:

Si queremos establecer una variable que contenga la fecha del 10 de febrero de 2009, el comando será el siguiente:

```
PS > $Date = Get-Date -Year 2009 -Month 2 -Day 10
```

Observe que cualquier parámetro que no se ha precisado toma el valor correspondiente de la fecha del día.

### b. Modificar una fecha

En la ejecución de un script o para una aplicación de terceros podemos querer modificar una fecha determinada. Para ello, será necesario utilizar la familia de métodos `Add*`.

Los métodos `Add` permiten añadir un entero relativo de días con `AddDays`, de horas con `AddHours`, de milisegundos con `AddMilliseconds`, de meses con `AddMonth`, de segundos con `AddSeconds`, de años con `AddYears` y de ticks con `AddTicks`.



Los enteros relativos son el conjunto de enteros (0,1,2,3..) positivos y negativos (0,-1,-2,-3,..).

Por ejemplo, para saber qué día de la semana será el mismo día que hoy pero dentro de un año, basta con añadir un año a la fecha actual y recuperar el día de la semana correspondiente.

```
PS > $date = Get-Date
PS > $date.AddYears(1).DayOfWeek

Friday
```

De la misma manera, es fácil de recuperar el día de su nacimiento.

Ejemplo:

```
PS > $date = [DateTime]'02/09/1974'
PS > $date.DayOfWeek

Saturday
```



Cuando se especifique una fecha como en el ejemplo anterior, el formato esperado es el formato anglosajón, es decir: mes/día/año.

### c. Comparar fechas

Existen varios tipos de comparación de fechas, la comparación más sencilla se efectúa con el método `CompareTo`. Aplicado a la variable de tipo `DateTime`, este método permite una comparación rápida y devuelve los valores siguientes:

Valor de retorno	Descripción
------------------	-------------



<b>-1</b>	Si la fecha es anterior a aquélla con la que se la compara.
<b>1</b>	Si es posterior.
<b>0</b>	Si son iguales.

Ejemplo:

Comparación de la fecha de dos archivos.

Para ello, basta con recuperar una a una las fechas de creación y compararlas con el método **CompareTo**.

```
PS > $Fecha_archivo_1 = (Get-item Archivo_1.txt).Get_CreationTime()
PS > $Fecha_archivo_2 = (Get-item Archivo_2.txt).Get_CreationTime()
PS > $Fecha_archivo_1.CompareTo($Fecha_archivo_2)
-1
```

El segundo método consiste en calcular el tiempo transcurrido entre dos fechas para de este modo poder compararlas. Esta operación es posible gracias al commandlet **New-TimeSpan**. Para obtener mayor información sobre él teclee: **help New-TimeSpan**.

Ejemplo:

Cálculo del tiempo transcurrido desde su nacimiento.

Para determinar el número de segundos transcurridos desde su nacimiento.

Es necesario, en primer lugar, calcular el tiempo transcurrido gracias al comando **New-TimeSpan**. Luego, en segundo lugar, transformaremos el valor recibido por el comando **New-TimeSpan** en segundos.

```
PS > New-TimeSpan $(Get-Date -Year 1985 -Month 10 -Day 6 `
-Hour 8 -Minute 30) $(Get-Date)

Days           : 9001
Hours          : 9
Minutes        : 4
Seconds        : 0
Milliseconds    : 0
Ticks          : 7777190400000000
TotalDays      : 9001,37777777778
TotalHours     : 216033,066666667
TotalMinutes   : 12961984
TotalSeconds   : 777719040
TotalMilliseconds : 777719040000
```

Resultado en segundos:

```
PS > (New-TimeSpan $(Get-Date -Year 1985 -Month 10 -Day 6 `
-Hour 8 -Minute 30) $(Get-Date)).TotalSeconds

777719100
```



El comando **New-TimeSpan** devuelve un valor de tipo **Timespan**. Para observar todos los métodos aplicables al tipo **Timespan**, teclee el comando siguiente: **New-Timespan | Get-Member**

## 6. Aplicaciones de todo tipo

### a. Manipulaciones en torno a las fechas

En esta parte, proporcionamos algunos ejemplos de scripts para demostrar el alcance de las distintas aplicaciones posibles en torno a las fechas.

#### Ejemplo:

*Visualización del mes en curso en modo gráfico.*

Aunque no hayamos abordado todavía las clases gráficas de Framework .NET con PowerShell (véase el capítulo .NET - Crear interfaces gráficas - Windows Forms), esta función le ofrece una breve reseña de las posibilidades gráficas ofrecidas.

Observe que para instanciar los objetos de la clase `Windows.Forms.Form` es necesario previamente cargar el assembly correspondiente. No haremos más hincapié en este momento ya que veremos todo esto ampliamente en el capítulo .NET.

Veamos el script:

```
#Calendario.ps1

# Carga del assembly Gráfico
[System.Reflection.Assembly]::LoadWithPartialName('System.windows.forms')

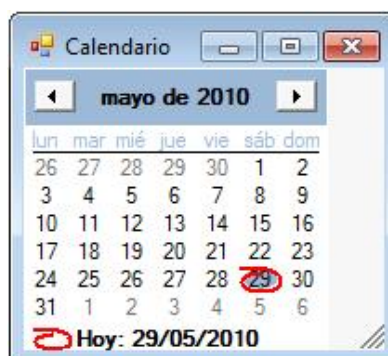
# Creación del objeto Form
$form = new-object Windows.Forms.Form
$form.Text = 'Calendario'
$form.Size = new-object Drawing.Size(210,190)

# Creación del objeto calendario
$calendario = new-object System.Windows.Forms.MonthCalendar

# Añadir el calendario al formulario
$form.Controls.Add($calendario)

# Visualización del formulario
$form.Add_Shown({$form.Activate()})
[void]$form.ShowDialog()
```


#### Resultado:



## b. Active Directory

Si usted da un vistazo en Active Directory y busca la fecha del último «logon» de un usuario, ¡¡¡ no vaya a caerse de la silla cuando vea una cifra alucinante en 64 bits !!!

En realidad, esta cifra corresponde al número de intervalos de 10 millonésimas transcurridos entre el 1º de enero de 1601 a 0h00 y la fecha en cuestión.

 **Un poco de historia:** el calendario gregoriano, que fue instaurado en 1582 por el Papa Gregorio XIII estipula que un año se compone de 365 días, salvo cuando es bisiesto, es decir, divisible por 4 y salvo los años seculares (divisibles por 100), que no son bisiestos salvo si son divisibles por 400. Ahora bien, en lo que nos concierne, el último múltiplo de 400 antes de la era informática es el año 1600. Es, pues esta fecha la que se utiliza como punto de partida para simplificar el algoritmo de determinación de la fecha.

Evidentemente, es conveniente convertir este número en una fecha «humanamente» comprensible. Es aquí donde interviene el método `AddTicks` del objeto `DateTime`. En efecto, puesto que un tick corresponde a un intervalo de diez millonésimas de segundo, no tendremos más que añadir tantos ticks como indique el valor del `lastlogon` a fecha del 1º de enero de 1601 a 0h00, para obtener una fecha representativa.

### Ejemplo:

La primera etapa consistirá evidentemente en recuperar los usuarios presentes en Active Directory:

```
PS > $ldapQuery = '(&(objectCategory=user))'
PS > $de = New-Object System.DirectoryServices.DirectoryEntry
PS > $ads = New-Object System.DirectoryServices.DirectorySearcher `
    -argumentlist $de,$ldapQuery
PS > $complist = $ads.FindAll()
```

Vemos entonces que la variable `$complist` contiene todos los usuarios. Queda ahora mostrar el nombre de usuario así como la fecha de su última conexión.

```
PS > ForEach ($i in $complist) {
    $LastLogon = $i.Properties['lastlogon']
    $LastLogon = [int64]::parse($LastLogon)

    # convierte la representación de LastLogon en forma de un entero de 64 bits

    #Creación de una fecha: 1/1/1601
    $fecha = (Get-Date -Year 1601 -Month 1 -Day 1 -Hour 0 `
        -Minute 0 -Second 0)

    #Añade los ticks a la fecha de origen
    $fecha_ultima_conexion = $fecha.AddTicks($LastLogon)
    Write-Host si.properties['name'] ": $fecha_ultima_conexion"
}
```

## c. Los archivos

Aquí tiene algo más espectacular y hasta ahora imposible de realizar con el explorador.

Gracias a PowerShell, ahora puede cambiar la fecha del último acceso a los archivos, la fecha de la última modificación e

incluso la fecha de creación, lo que puede provocar situaciones bastante inesperadas. Y ésta es la vía.

### **Etapas 1 - Creación de un archivo**

```
PS > New-Item -Name prueba.txt -Type File

Directorio: C:\Temp

Mode                LastWriteTime         Length Name
----                -
-a---            20/09/2009      12:59             0 prueba.txt
```

### **Etapas 2 - Verificación de la fecha de creación de este archivo**

```
PS > (Get-Item prueba.txt).Get_CreationTime()
domingo 20 septiembre 2009 13:00:58
```

### **Etapas 3 - Atribución de la nueva fecha de creación y del último acceso**

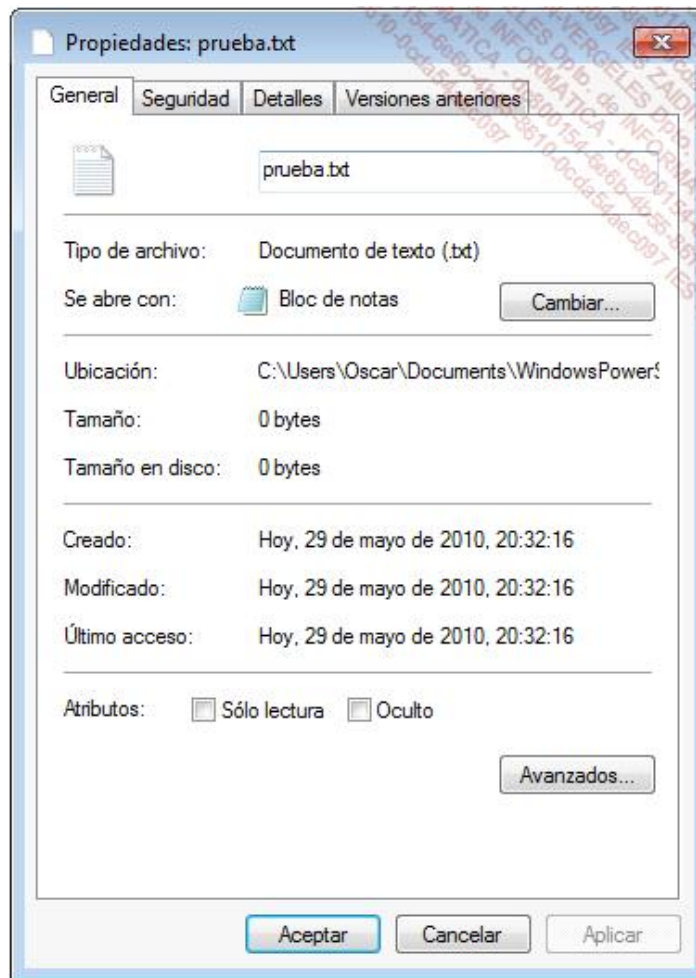
Para ello crearemos dos variables: `$fecha_ultimo_acceso` que equivale a la fecha del 13 de julio de 1998 y `$fecha_creacion` que aumentaremos hasta el 10 de julio del 2020.

```
PS > $fecha_ultimo_acceso = (Get-Date -Year 1998 -Month 7 `
                             -Day 12 -Hour 0 -Minute 0 -Second 0)
PS > $fecha_creacion = (Get-Date -Year 2012 -Month 1 `
                         -Day 1 -Hour 0 -Minute 0 -Second 0)
PS > $Archivo = Get-Item prueba.txt
PS > $Archivo.Set_CreationTime($fecha_creacion)
PS > $Archivo.Set_LastAccessTime($fecha_ultimo_acceso)
```



La creación de un objeto correspondiente a un archivo es una etapa intermedia que podrá ser sustituida por la notación siguiente: `(Get-Item prueba.txt).Set_CreationTime($fecha_creacion)`

Podrá ver ahora las propiedades de su archivo pulsando el botón secundario del ratón en el archivo - **Propiedades**.



*Propiedades del archivo*

O bien, tecleando el comando siguiente:

```
PS > Get-Item prueba.txt | Format-Table CreationTime,LastWriteTime,
LastAccessTime
```

CreationTime	LastWriteTime	LastAccessTime
-----	-----	-----
01/01/2012 00:00:00	20/09/2009 13:03:29	12/07/1998 00:00:00