Consola de Windows y PowerShell

Gianny A. Uffo

## Índice.

[Consola de Windows.](#_sscqdtt5ja93)

[PowerShell](#_ngdqc53bsnkl)

[Variables](#_fschd5wuci35)

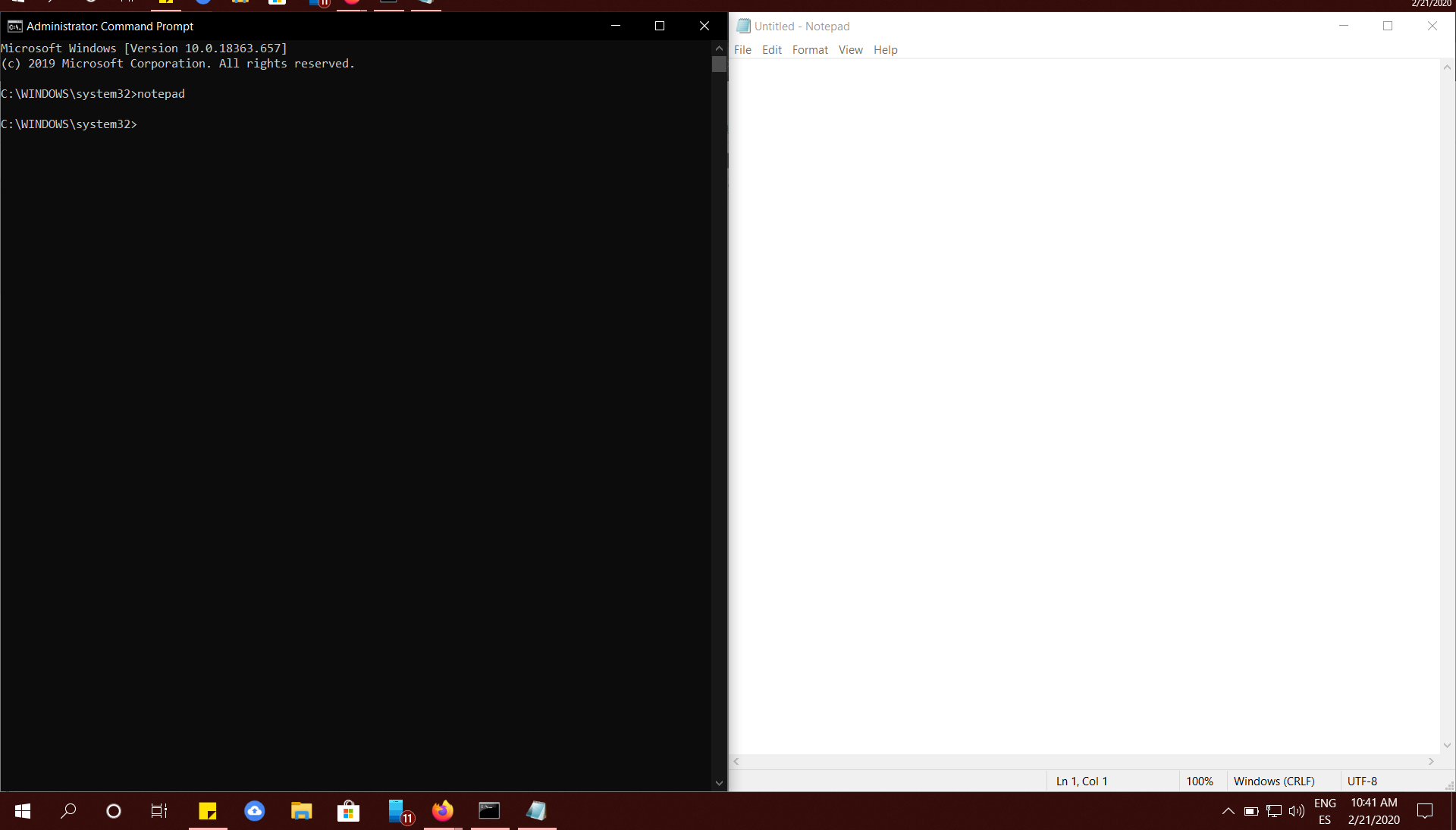
[Archivos y carpetas](#_e2zdumvutyq9)

[Rutas absolutas y relativas.](#_6g7wxsrkfzgg)

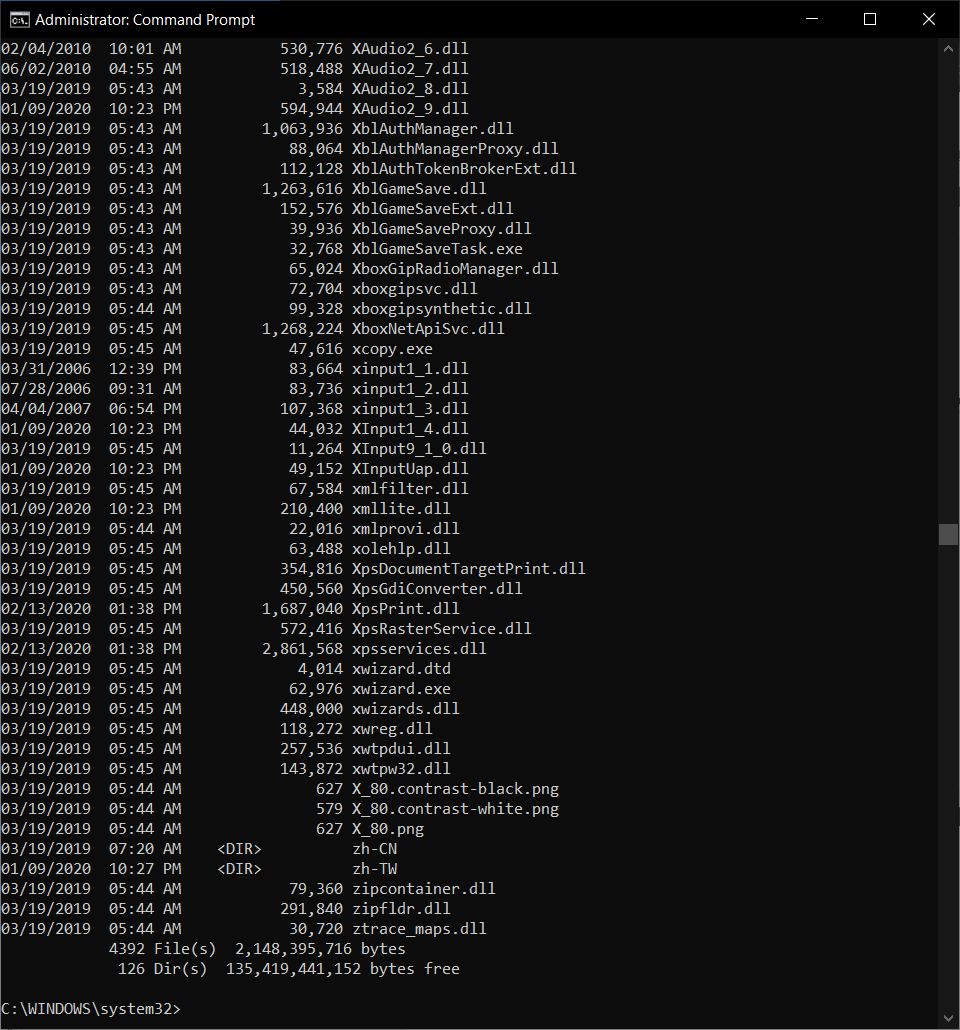
[Bibliografía.](#_3698ppm0mkz3)

# 

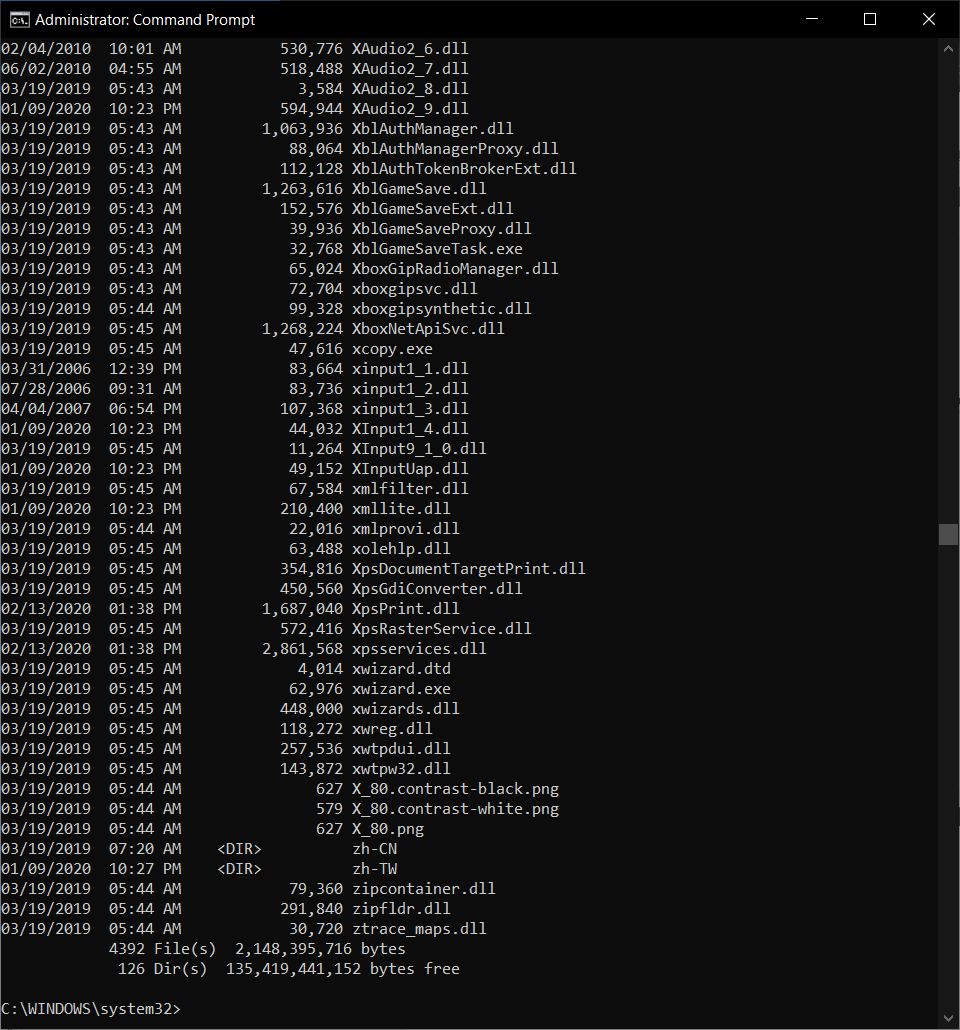
# Consola de Windows.



Ejecución de Notepad en la cmd.

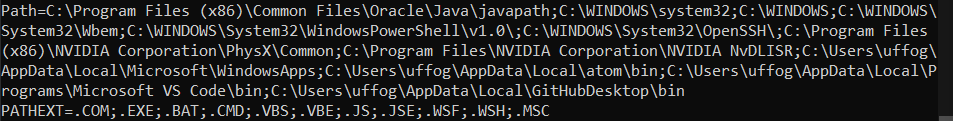


Ejecución del comando dir en la cmd.



1- Ejecución de “set” para mostrar las variables de entorno.

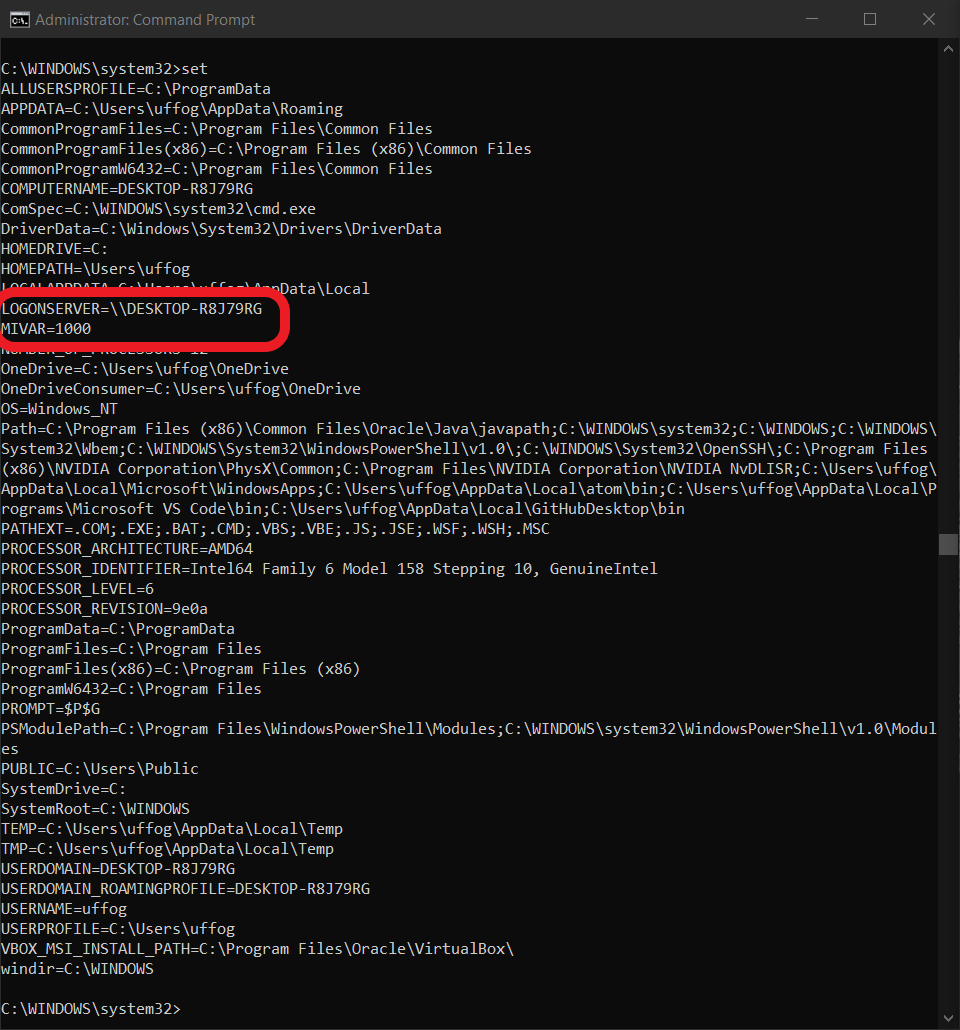
2- Variables de entorno: las variables de entorno son cadenas que contienen información acerca del entorno para el sistema y el usuario que ha iniciado sesión en ese momento. Algunos programas usan la información para determinar dónde se colocan los archivos (como los archivos temporales).

3- **Variable de entorno PATH**: El PATH de un sistema operativo es una variable de entorno que guarda las rutas donde el sistema operativo puede encontrar los ejecutables (archivos que inician la ejecución de un programa) para el intérprete de comandos del propio sistema operativo. Una variable de entorno, como cualquier variable guarda un dato. La variable PATH guarda distintas rutas a directorios, donde se encuentran los ejecutables. La variable PATH es ampliable en Windows, MacOS y Linux. Es decir, podemos especificarle al sistema operativo dónde buscar los programas ejecutables. Ej: *C:\Windows\System32\;C:\Windows\.*

**Variable de entorno TMP:** Cada vez que utilizamos programas en nuestro ordenador para realizar diversas tareas, simultáneamente y sin que podamos verlo a primera vista, se van creando archivos que se denominan “temporales”. Se crean con el fin de contener información transitoria durante el tiempo que nos encontramos trabajando con el software . Después de finalizar el proceso, se debería borrar este archivo. **Los archivos temporales son utilizados con el objetivo de recuperar datos que puedan llegar a perderse en el caso de que el programa o el PC se llegaran a detener de manera anormal y repentina.** 

Ej:*C:\DOCUME~1\{usuario}\LOCALS~1\Temp.*

**Variable de entorno HOMEPATH:** se le llama a la carpeta dentro del disco en el que está instalado el SO que se le asigna al usuario local o del usuario almacenado en servidores actual, como es el caso de Windows. Ej: *\Documents and Settings\{usuario}.*

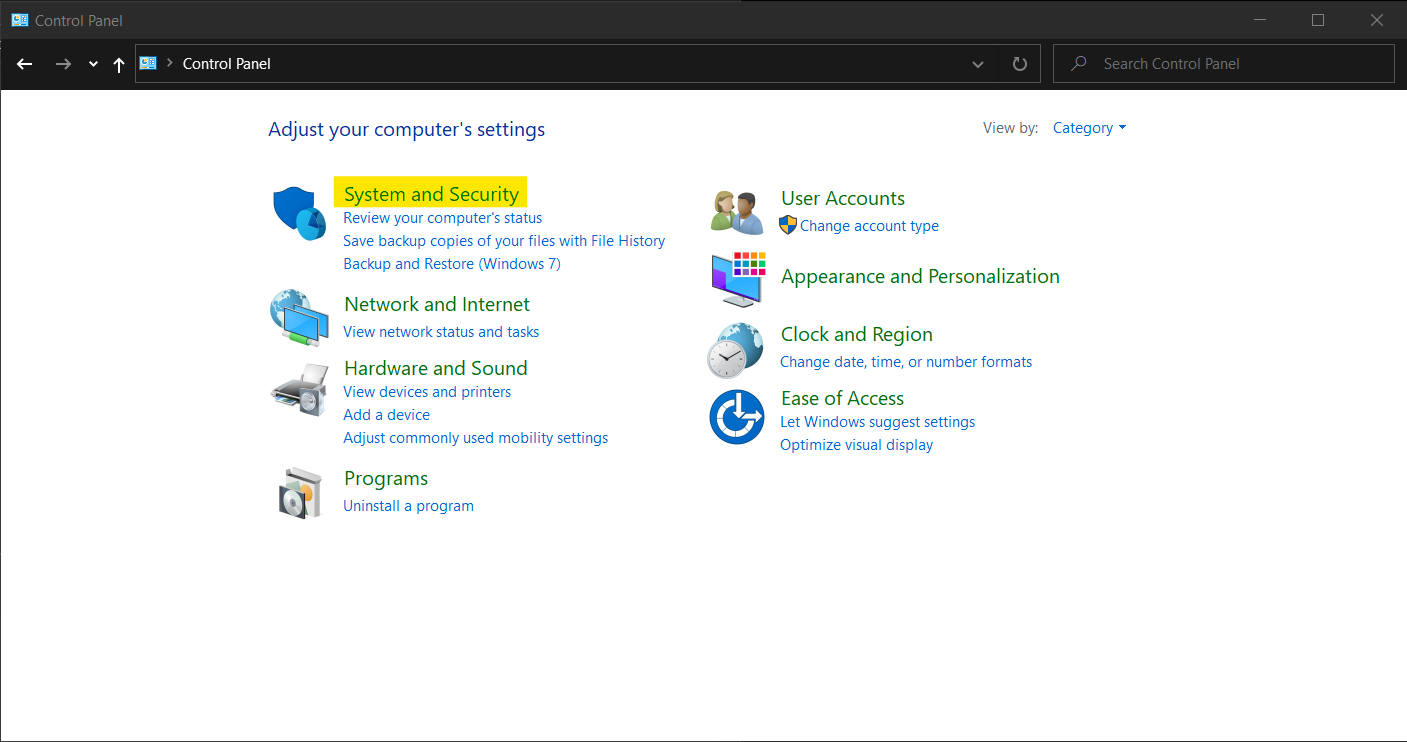


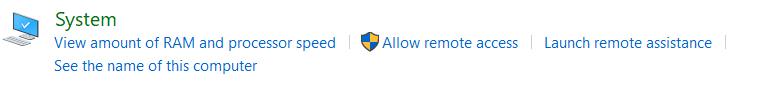
*Info extraída de* [*Soporte técnico de Office.*](https://support.office.com/es-es/article/agregar-citas-en-un-documento-de-word-ab9322bb-a8d3-47f4-80c8-63c06779f127)

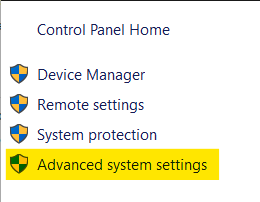
4- Variable propia con “set”.



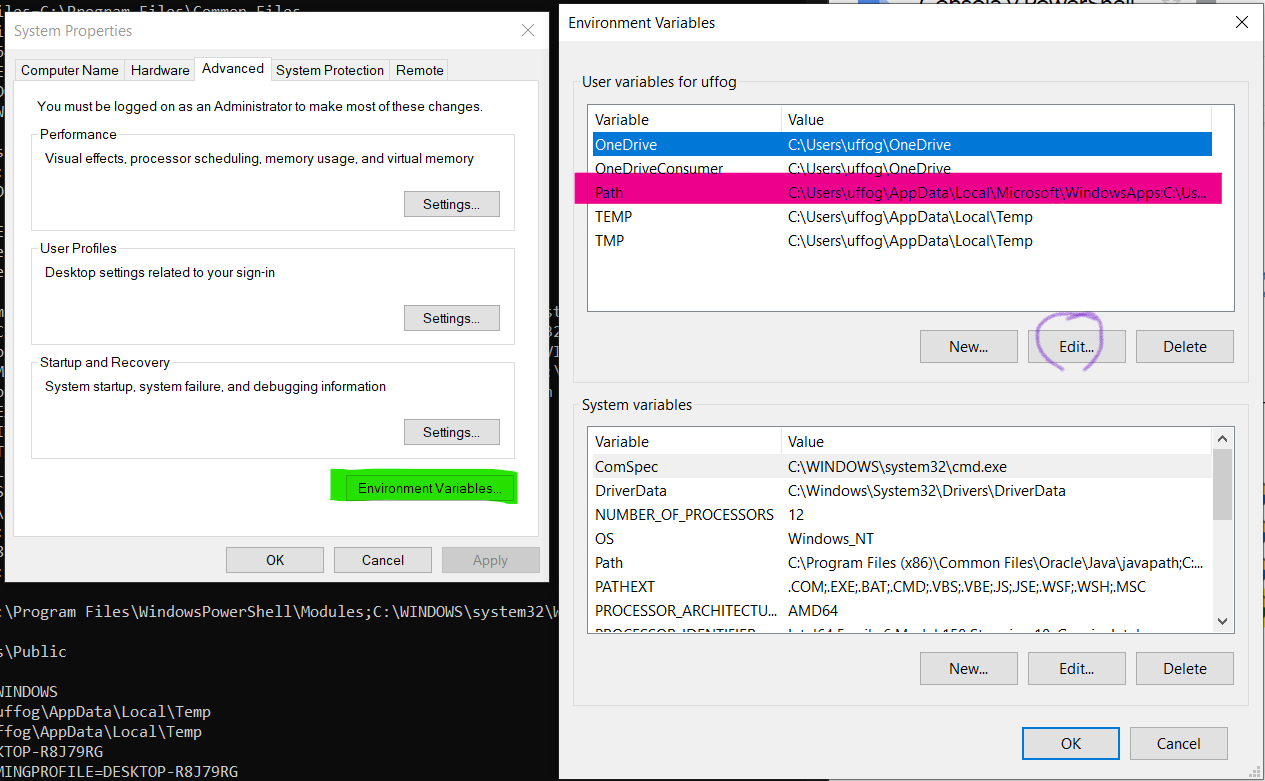
5- Control Panel\System and Security\System\Advanced system settings

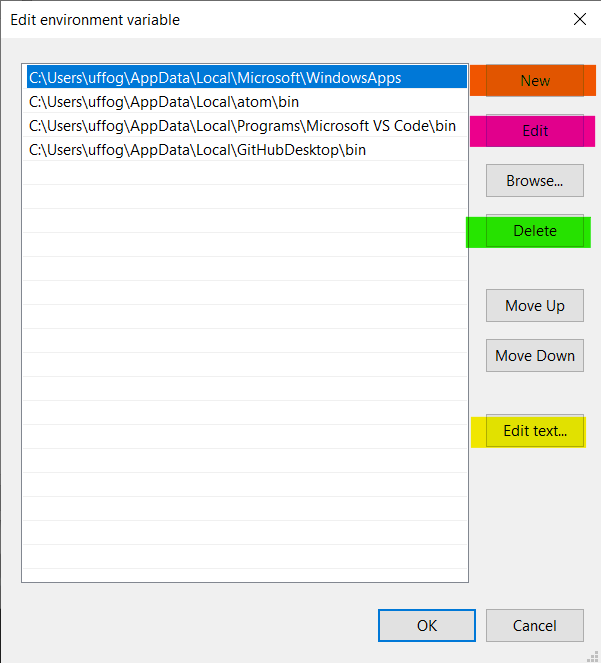






\Environment Variables…\Path\Edit



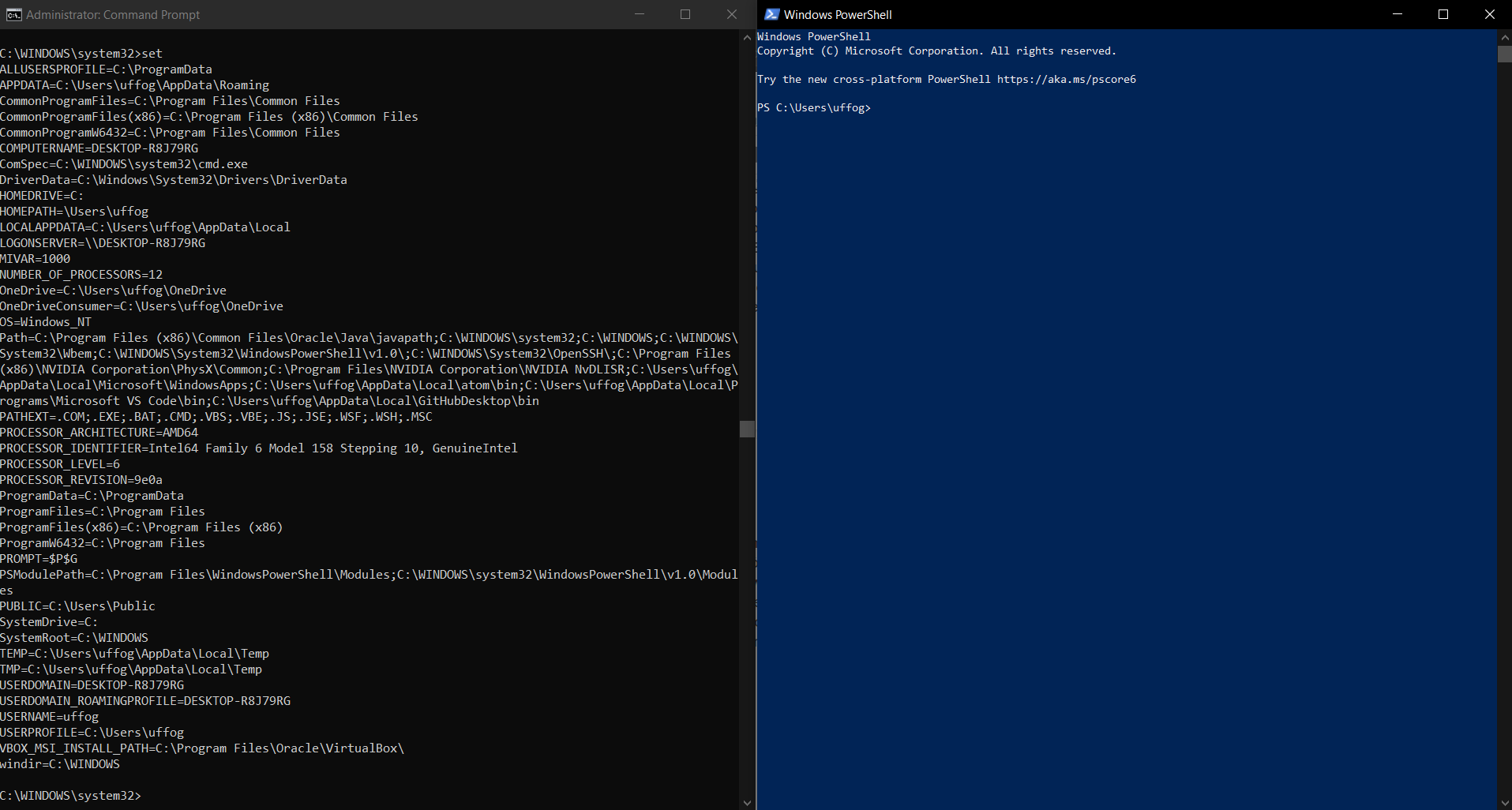
Se nos abrirá una ventana como esta en la que podremos realizar distintas acciones como crear una ruta nueva, editarla…

# PowerShell

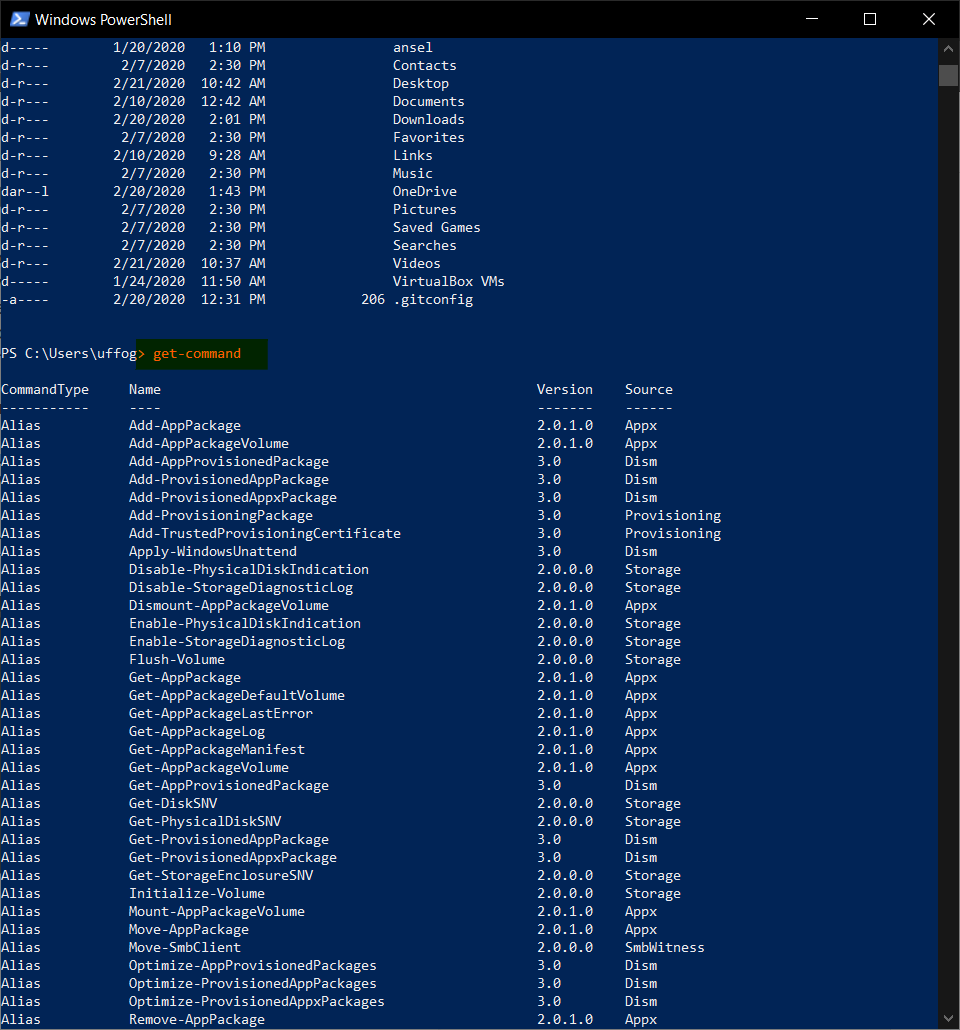
1. General.
   1. Powershell es una **herramienta orientada a administradores de sistemas** que permite automatizar tareas en un servidor (o equipo) y realizar determinadas tareas con un mayor control sobre el sistema. A parte de poder controlar el sistema operativo Windows, Powershell también permite controlar determinadas aplicaciones como Microsoft Exchange, SQL Server o IIS.

El lenguaje de la consola es muy completo. El objetivo de Powershell es llegar a igualarse a la programación PERL de UNIX para poder realizar tareas administrativas de una forma muy rápida y sencilla para los administradores de sistemas. Powershell pone a disposición del usuario gran cantidad de elementos con los que interactuar como variables, operadores matemáticos, vectores, booleanos, comparadores, comentarios y más elementos utilizados en la programación de scripts.

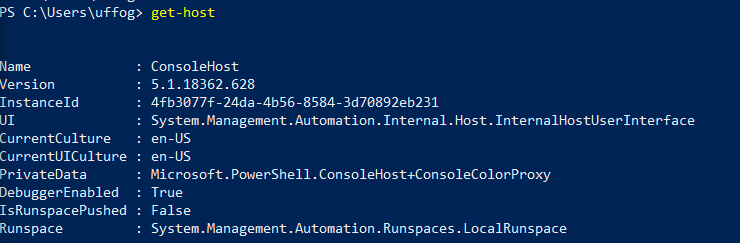
* 1. Se encuentra disponible desde Windows XP hasta Windows 8.1 y Windows 10, incluyendo sus correspondientes versiones de servidor, aunque es a partir de Windows 7 cuando la aplicación comenzó a venir instalado por defecto en los sistemas. Para los sistemas antiguos hay que instalarlo manualmente.
  2. Son completamente diferentes, a pesar de que el comando "dir" funciona de la misma manera en ambas interfaces. PowerShell usa cmdlets, que son objetos de programación autónomos que exponen las opciones de administración subyacentes dentro de Windows.

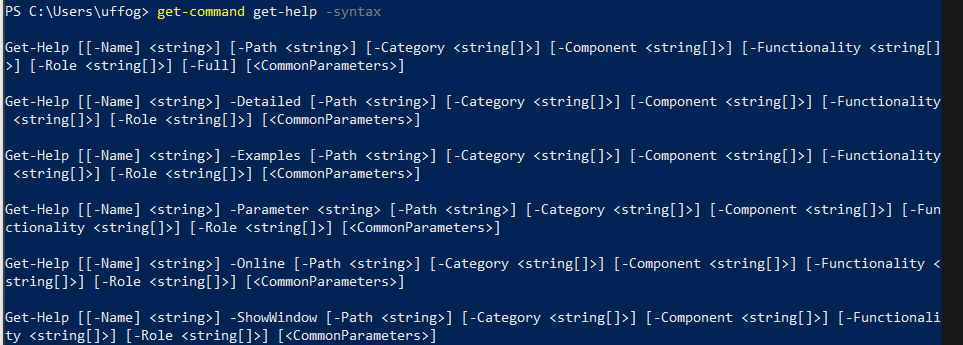
PowerShell usa tuberías para encadenar cmdlets y compartir datos de entrada / salida de la misma manera que otros shells, como bash en linux. Las tuberías permiten a los usuarios crear scripts complejos que pasan parámetros y datos de un cmdlet a otro. Los usuarios pueden crear scripts reutilizables para automatizar o realizar cambios masivos con datos variables, por ejemplo, una lista de servidores.

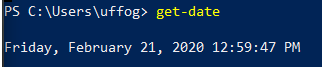
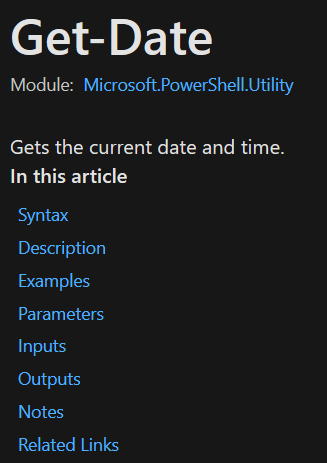
* 1. Consola de PowerShell a la izquierda.



* 1. **get-command** es el comando para mostrar todos los comandos disponibles en PowerShell.
  2. get-host nos muestra las características de PowerShell.

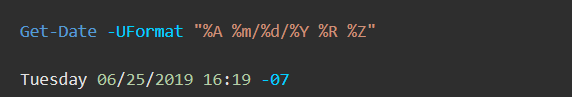


* 1. Con get-command + comando a conocer la sintaxis + -syntax podremos ver cómo poder utilizar el comando que queremos, con los tipos de datos y los parámetros que tendremos que utilizar. En este caso obtenemos la sintaxis del comando que nos sirve como guía y ayuda en PowerShell. 

1. Fechas.
   1. Con **get-date** tendremos una salida de pantalla con el día, la fecha y la hora exacta de la ejecución.
   2. Con la [ayuda online](https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/get-help?view=powershell-7) que Microsoft nos ofrece acerca del comando podemos ver su sintaxis, descripción, ejemplos y varias funcionalidades y características acerca del mismo.
   3. **Ut**ilizando .NET Format (explicado dos puntos más adelante) se puede configurar la fecha según las características que queramos sacar por consola. En este caso queremos tener el formato 20-02-2020 y el resultado obtenido es el siguiente:

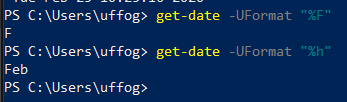


* 1. **Utilizando UFormat:**

Se utilizan los parámetros UFormat para especificar varios formatos.

*Sintaxis a seguir con UFormat.*

|  |  |  |
| --- | --- | --- |
| Format specifier Meaning Example | Format specifier Meaning Example | Format specifier Meaning Example |
| %A | Nombre completo día de la semana. | Monday |
| %a | Nombre del día de la semana abreviado. | Mon |
| %B | Nombre completo mes | January |
| %b | Nombre abreviado del mes | Jan |
| %C | Siglo | 20 of 2019 |
| %c | Fecha y tiempo: abreviado | Thu Jun 27 08:44:18 2019 |
| %D | Fecha mm/dd/yy | 06/27/19 |
| %d | Día del mes: 2 dígitos. | 02 |
| %e | Día del mes precedido de un espacio | <space>2 |
| %F | Fecha en el formato: YYYY-mm-dd, igual a %Y-%m-%d (the ISO 8601 formato de fecha) | 2020-02-25 |
| %G | Igual que ‘Y’: Año en formato de 4 dígitos. | 2020 |
| %H | Hora en formato 24h | 17 |
| %h | Nombre abreviado del mes | Feb |
| %I | Hora en formato 12h | 10 |
| %j | Día del año | 56 |
| %m | Número de mes | 02 |
| %p | AM or PM |  |
| %M | Minutos | 47 |



**Utilizando .NET format:**

****

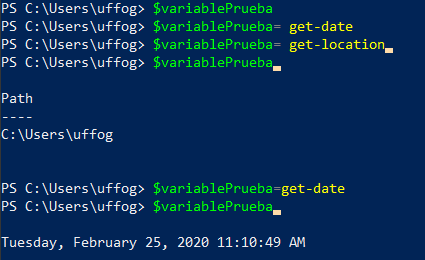
*Sintaxis a seguir con Format.*

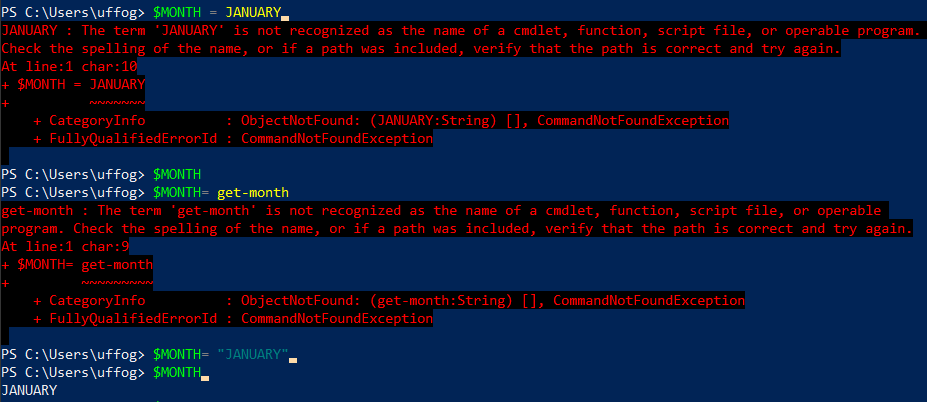
|  |  |
| --- | --- |
| Especificador | Definición |
| %A | Nombre completo día de la semana |
| %m | Número de mes |
| %d | Día del mes en dos dígitos |
| %Y | Año en formato de 4 dígitos |
| %R | Tiempo en formato 24h sin segundos |
| %Z | Zona Horaria fuera de UTC |

La condición para que funcione es escribir siempre get-date -format “” y aplicar entre comillas dobles el formato según los parámetros de la tabla que queramos.



Variables

2. Por defecto, cuando trabajamos con powershell no es necesario declarar las variables antes de utilizarlas. Al usar una variables para almacenar un datos, se declara automáticamente.
   1. Pueden crearse variables y asignarles un valor en el mismo paso. PowerShell solo crea la variable si no existe. De lo contrario, asigna el valor especificado a la variable existente. Una vez creada la variable con $, se nombra de forma inmediata y seguida al $ y se le asigna valores con =. La variable devolverá los valores que almacene por última vez.
   2. Al crear la variable MONTH no sabía cómo añadirles valores tipo String a la misma. Probé varias maneras hasta percatarme de las comillas dobles. Para vaciar la variable de algún modo, se pueden utilizar comillas dobles y un espacio en blanco entre ellas.



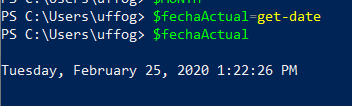
Al cerrar PowerShell y volver a abrirlo la variable creada anteriormente pierde los valores que tenía establecidos. Al ejecutar el comando, se crea la variable vacía.



En la siguiente tabla recojo algunas [variables predefinidas](https://mytcpip.com/2017/11/15/capsula-powershell-uso-de-variables-y-constantes-en-powershell/) en PowerShell que pueden ser de mucha utilidad a la hora de adentrarse más en el tema.

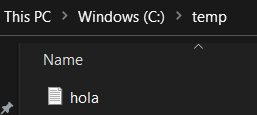
|  |  |
| --- | --- |
| $$ | Contains the last token in the last line received by the session. |
| $? | Contains the execution status of the last operation. Equivalent to %errorlevel% in the CMD shell. See also $LastExitCode below.  It contains TRUE if the last operation succeeded and FALSE if it failed. ReadOnly, AllScope |
| $^ | Contains the first token in the last line received by the session. |
| $\_ | Contains the current object in the pipeline object. You can use this variable in commands that perform an action on every object or on selected objects in a pipeline. |
| $Allnodes | This variable is available inside of a DSC configuration document when configuration data has been passed into it by using the -ConfigurationData parameter. |
| $Args | Contains an array of the undeclared [p](https://ss64.com/ps/syntax-args.html)arameters and/or parameter values that are passed to a function, script, or script block. When you create a function, you can declare the parameters by using the param keyword or by adding a comma-separated list of parameters in parentheses after the function name. |
| $ConsoleFileName | Contains the path of the console file (.psc1) that was most recently used in the session. This variable is populated when you start PowerShell with the PSConsoleFile parameter or when you use the Export-Console cmdlet to export snap-in names to a console file.  When you use the Export-Console cmdlet without parameters, it automatically updates the console file that was most recently used in the session. You can use this automatic variable to determine which file will be updated.ReadOnly, AllScope |
| $Error | Contains an array of error objects that represent the most recent errors. Constant  The most recent error is the first error object in the array ($Error[0]). |
| $Event | Contains a PSEventArgs object that represents the event that is being processed. This variable is populated only within the Action block of an event registration command, such as Register-ObjectEvent. The value of this variable is the same object that the Get-Event cmdlet returns. Therefore, you can use the properties of the $Event variable, such as $Event.TimeGenerated , in an Action script block. |
| $EventArgs | Contains an object that represents the first event argument that derives from EventArgs of the event that is being processed. This variable is populated only within the Action block of an event registration command. The value of this variable can also be found in the SourceEventArgs property of the PSEventArgs (System.Management.Automation.PSEventArgs) object that Get-Event returns. $EventSubscriber |
| $EventSubscriber | Contains a PSEventSubscriber object that represents the event subscriber of the event that is being processed. This variable is populated only within the Action block of an event registration command. The value of this variable is the same object that the Get-EventSubscriber cmdlet returns. |
| $ExecutionContext | Contains an EngineIntrinsics object that represents the execution context of the Windows PowerShell host. You can use this variable to find the execution objects that are available to cmdlets. Constant, AllScope |
| $False | Contains FALSE. You can use this variable to represent FALSE in commands and scripts instead of using the string «false». The string can be interpreted as TRUE if it is converted to a non-empty string or to a non-zero integer. Constant, AllScope |
| $ForEach | Contains the enumerator of a ForEach-Object loop. You can use the properties and methods of enumerators on the value of the $ForEach variable. This variable exists only while the For loop is running. It is deleted when the loop is completed. |
| $Home | Contains the full path of the user’s home directory. ReadOnly, AllScope This variable is the equivalent of the %HomeDrive%%HomePath% environment variables, typically C:\Users\<user> |
| $Host | Contains an object that represents the current host application for Windows PowerShell. You can use this variable to represent the current host in commands or to display or change the properties of the host, such as $Host.version or $Host.CurrentCulture, or $host.ui.rawui.setbackgroundcolor(«Red»). Constant, AllScope |
| $Input | An enumerator that contains the input that is passed to a function. The $Input variable is case-sensitive and is available only in functions and in script blocks. (Script blocks are essentially unnamed functions.) In the Process block of a function, the $Input variable contains the object that is currently in the pipeline. When the Process block is completed, the value of $Input is NULL. If the function does not have a Process block, the value of $Input is available to the End block, and it contains all the input to the function. |
| $LastExitCode | Contains the exit code of the last Windows-based program that was run. |
| $Matches | The $Matches variable works with the -match and -not match operators. When you submit scalar input to the -match or -notmatch operator, and either one detects a match, they return a Boolean value and populate the $Matches automatic variable with a hash table of any string values that were matched. For more information about the -match operator, see about\_comparison\_operators. |
| $MyInvocation | Contains an object with information about the current command, such as a script, function, or script block. You can use the information in the object, such as the path and file name of the script ($myinvocation.mycommand.path) or the name of a function ($myinvocation.mycommand.name) to identify the current command. See also $PSScriptRoot |
| $NULL | Contains a NULL or empty value. A scalar value that contains nothing. |
| $OFS | $OFS is a special variable that stores a string that you want to use as an output field separator. Use this variable when you are converting an array to a string. By default, the value of $OFS is » «, but you can change the value of $OFS in your session, by typing $OFS=»*value*«. If you are expecting the default value of » » in your script, module, or configuration output, be careful that the $OFS default value has not been changed elsewhere in your code. |
| $PID | Contains the process identifier (PID) of the process that is hosting the current Windows PowerShell session. Constant, AllScope  If you stop this process, you will brute force kill the PowerShell host that is executing the script. |
| $Profile | Contains the full path of the Windows PowerShell profile for the current user and the current host application. You can use this variable to represent the profile in commands. For example, you can use it in a command to determine whether a profile has been created: test-path $profile Or, you can use it in a command to create a profile: new-item -type file -path $pshome -force You can also use it in a command to open the profile in Notepad: notepad $profile |

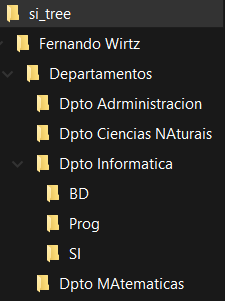
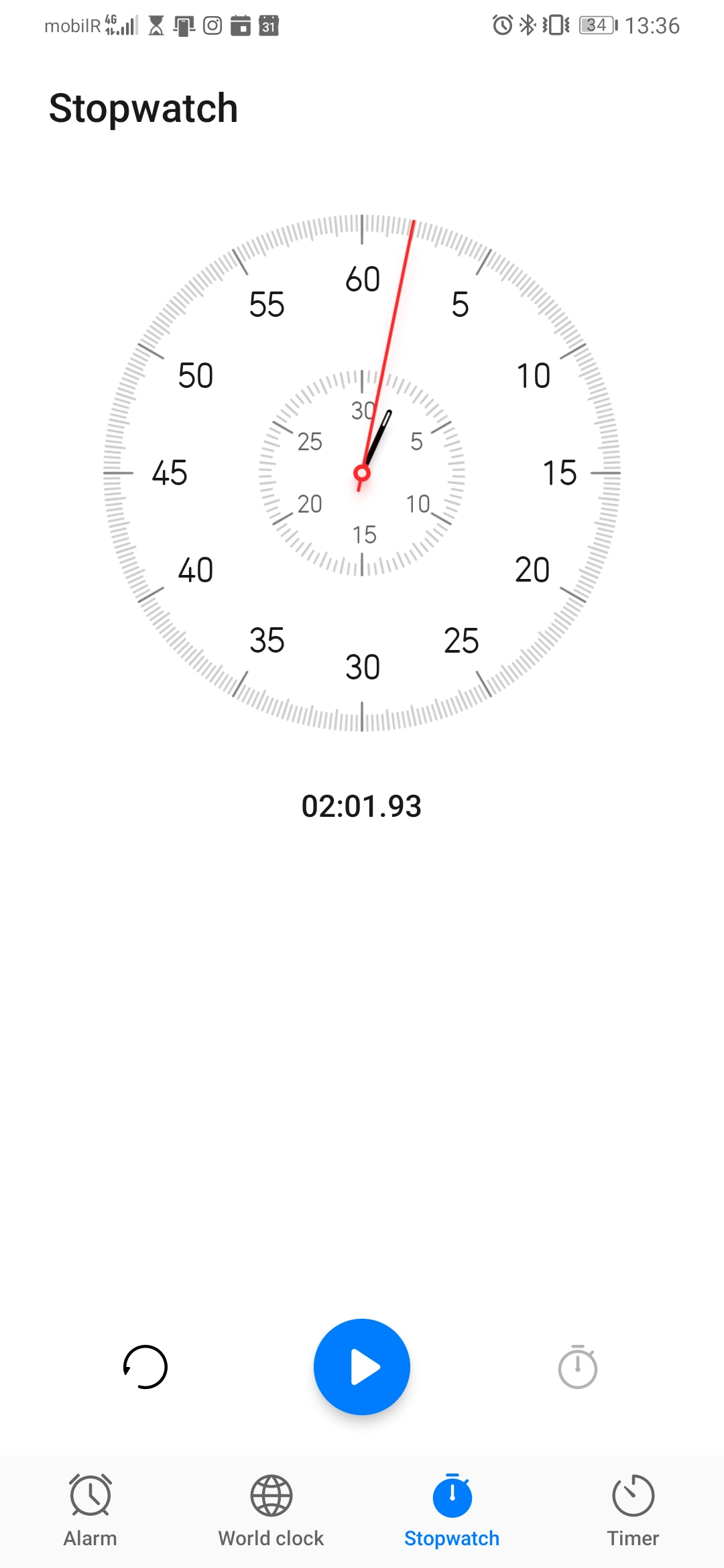
3.8



Este es el resultado de asignarle el valor de la fecha actual y de mi ubicación a la variable.

# Archivos y carpetas

1. 1. 



### Rutas absolutas y relativas.

**Rutas absolutas​:** son rutas a carpetas o archivos que incluyen la ruta completa que hay que seguir desde la raíz del sistema de ficheros hasta dicho archivo o carpeta.

**Ej. ruta absoluta:**

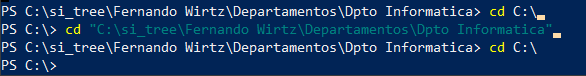
C:\si\_tree\Fernando Wirtz\Departamentos\Dpto Informatica\SI\adios.txt

**Movernos de carpeta utilizando una ruta absoluta:**

cd “C:\si\_tree\Fernando Wirtz\Departamentos\Dpto Informatica”

**Ir a la raíz de la unidad:**

cd C:\

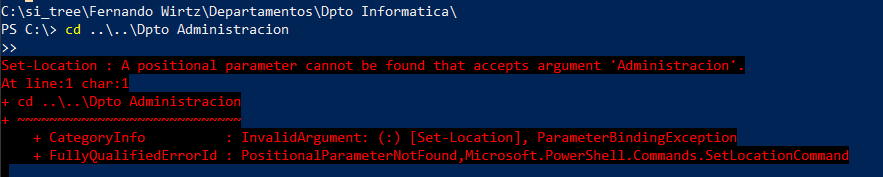


**Rutas relativas​:** son las rutas a carpetas o archivos que hay que seguir desde nuestra posición actual en la jerarquía de carpetas.

**Si nos encontramos en la carpeta SI y queremos ir al directorio anterior: *cd ..***

* **Con lo que ahora estaríamos posicionados en la carpeta:** “C:\si\_tree\Fernando Wirtz\Departamentos\Dpto Informatica\”
* O si queremos ir a a “Dpto Administracion” podemos hacer:

cd ..\..\Dpto Administracion

* O si estamos en Departamentos y queremos ir a Dpto Informatica:

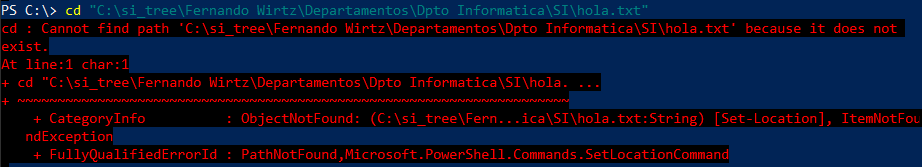
cd “Dpto Informatica”

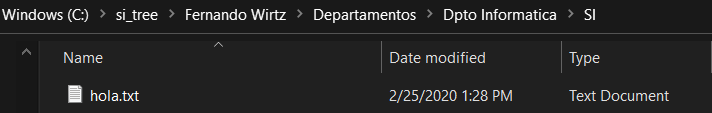
# También es una ruta relativa.

Tanto en Windows como en Linux:

* El caracter “.” se refiere al “directorio actual”.
* Los caracteres “..” se refieren al “directorio anterior.

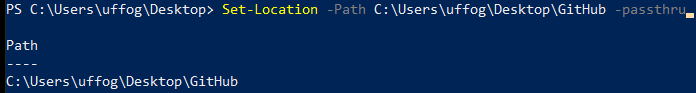
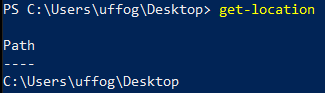
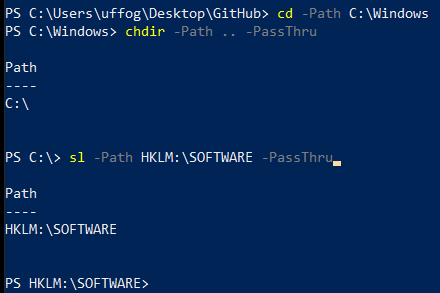
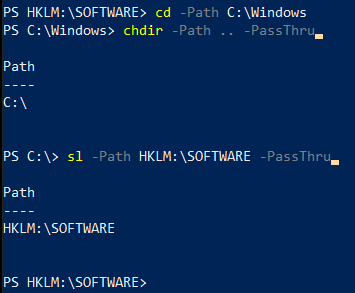
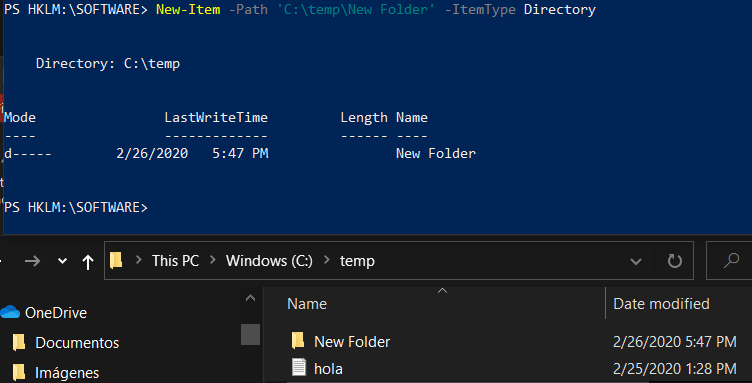
Ha ocurrido un error al intentar ejecutar esta ruta absoluta en PowerShell.



1. 1. 

* Para **establecer una ubicación concreta** se utiliza el cmdlet set-location con la siguiente estructura: 

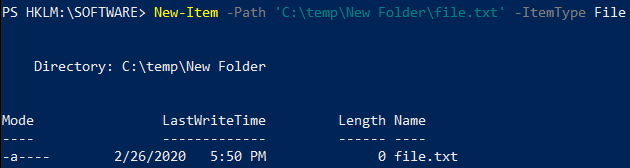
Si escribimos el comando **-PassThru** después de la ruta a la que queremos ir obtendremos inmediatamente la ubicación en la que nos encontramos.

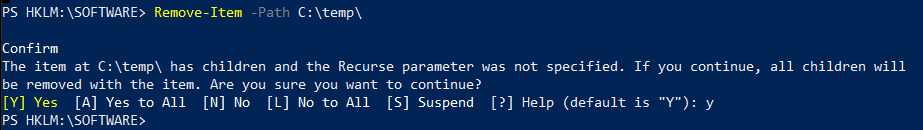
* Para **obtener la localización** en la que te encuentras se utiliza el cmdlet: get-location 
* En la notación estándar de las rutas de acceso relativas, **un punto ( . )** representa la carpeta actual y **dos puntos ( .. )**, el directorio principal de su ubicación actual. Así, si está en la carpeta **C:\Windows**, un punto ( **.** ) representa a **C:\Windows** y dos puntos ( **..** ), a **C:**
* Puede escribir Set-Location o usar cualquiera de los alias de Windows PowerShell integrados para Set-Location (cd, chdir, sl).
* Comando para **crear una nueva carpeta** C:\temp\Nueva carpeta

**New-Item -Path 'C:\temp\New Folder' -ItemType Directory**

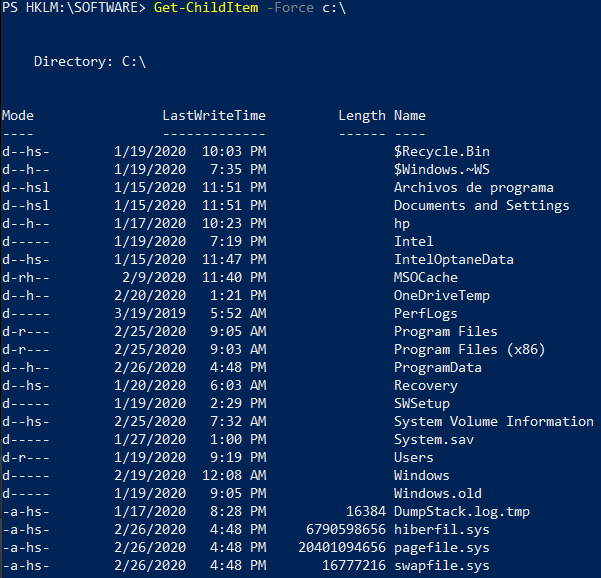
* Crear **un archivo vacío.**

**New-Item -Path 'C:\temp\New Folder\file.txt' -ItemType File**

* Pueden quitarse los elementos contenidos mediante **Remove-Item**, pero pedirá que se confirme la eliminación si el elemento contiene algo más. Por ejemplo, si intenta eliminar la carpeta C:\temp\ que contiene otros elementos, Windows PowerShell le pedirá confirmación antes de eliminar la carpeta. 

Si no se quiere que se solicite confirmación por cada elemento contenido, se especifica el parámetro -**Recurse** al final de la ruta.

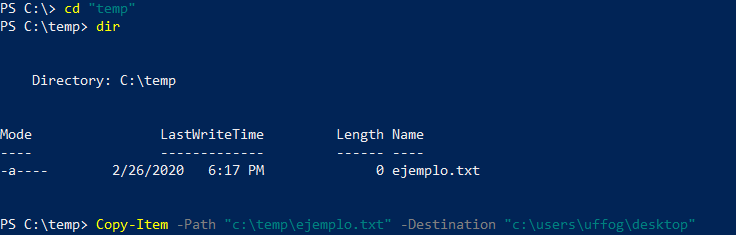
* Con el cmdlet **get-childitem** podremos obtener todos los archivos dentro de una ruta. Para ver incluso los ocultos y los del sistema utilizaremos **-force**

Con **-Recurse** obtendremos la lista de todo lo que se encuentra en esa ubicación incluídos los hijos. 

* Para **copiar** utilizaremos el cmdlet **Copy-Item -path** y **-Destination**

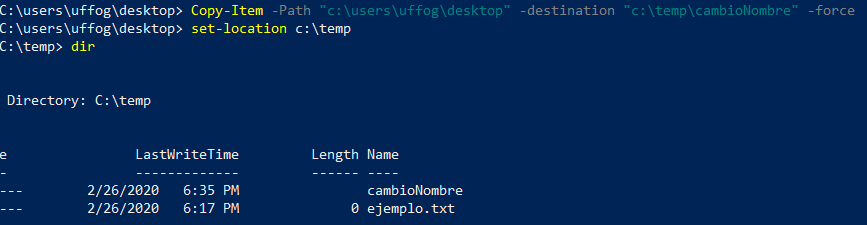
**Copy-Item -Path** *c:\ejemplo.txt* **-Destination** *c:\backup*

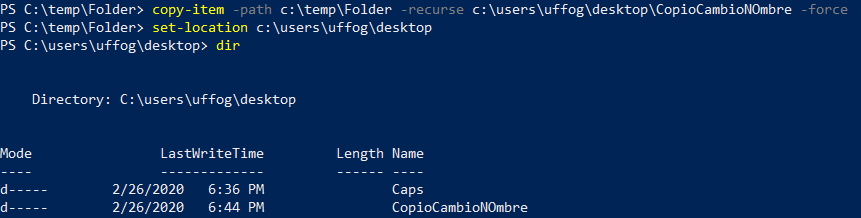
**Una recomendación** a la hora de copiar archivos es asegurarse de que nos encontramos en la carpeta de origen del archivo a copiar ya que sino, dará error. Podemos hacer **dir** antes de copiar para ver todos los archivos disponibles y después de hacerlo, hacer dir en la ubicación final.

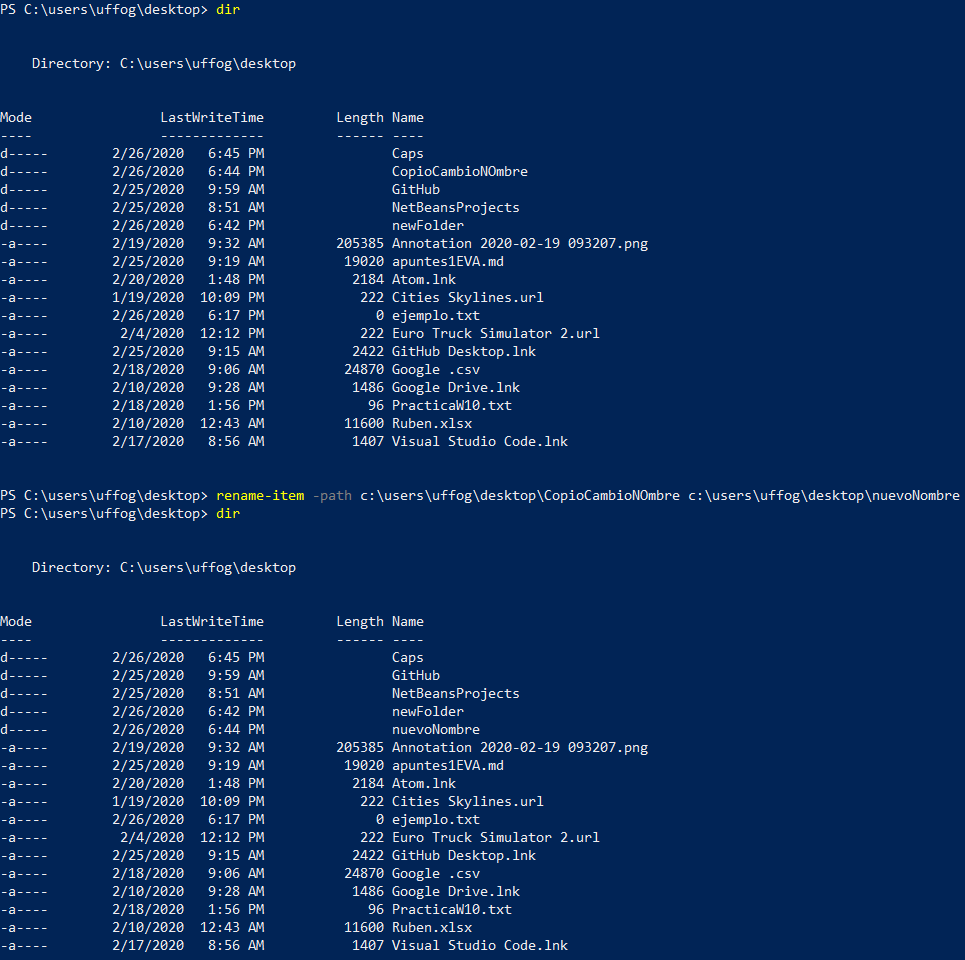


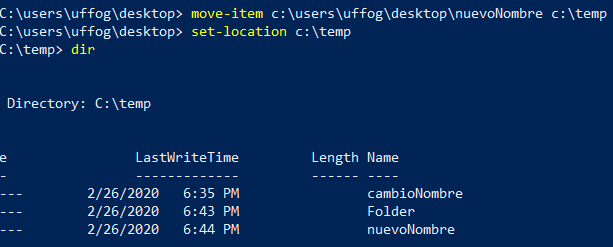
Podemos **copiar un archivo y renombrarlo** en la carpeta destinataria. Tendremos que en el destino poner el NombreNuevo y **-force** al final de la ruta.

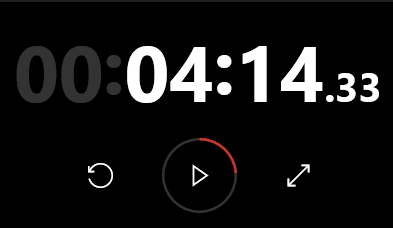
**Copy-Item -Path** *c:\ejemplo.txt* **-Destination** *c:\backup\****cambioNombre****.txt* **-Force**



* Para **copiar directorios completos** el comando es el mismo, utilizando un parámetro que hemos visto antes, “Recurse” para copiar todos los subdirectorios. Podemos incluso cambiarle el nombre a la carpeta de destino con -Force **Copy-Item -Path** *c:\musica* **-Recurse** *c:\archivo\musica* **-Force**
* También podremos renombrar los ficheros o directorios

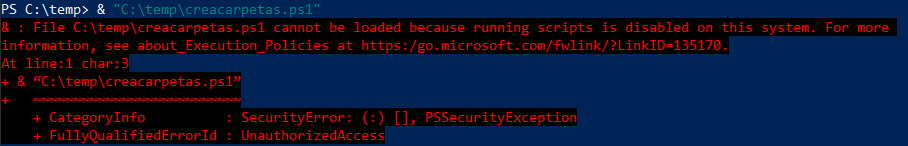
con el cmdlet **Rename-Item + “ubicación destino” + “ubicación cambiando el nombre”**

* Para **mover un elemento** utilizaremos el cmdlet: **move-item+ ubicación destino + ubicación final**

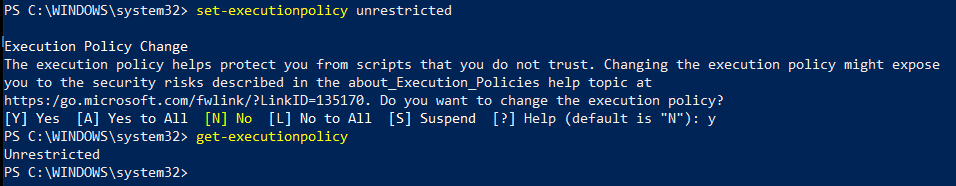
1. 1. ¿Cuáles han sido los tiempos? ¿Qué método es más cómodo, la consola o el explorador de archivos?

Ha resultado más cómodo crear todos los archivos y carpetas con la consola de comandos. En el momento en el que creas la ruta más larga, automáticamente se crean todas las carpetas padres superiores hasta llegar a la carpeta o archivo hijo.

* 1. Al intentar ejecutar el script desde la consola de PowerShell aparece un texto en rojo indicando que no podemos realizar la acción.







* 1. No, para crear todo este tipo de tareas es más fácil y rápido crear un script que puedan ejecutar todos los ordenadores y crear los archivos.
  2. La mayor ventaja es la velocidad en números muy grandes además de la programación de determinadas acciones. Alguna desventaja puede ser el conocimiento de los códigos para poder desarrollarlos.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Dificultad |  |  |  |  |  |  |  | x |  |  |
| Interesante |  |  |  |  |  |  |  |  | x |  |
| Ha aumentado mis conocimientos sobre el tema. |  |  |  |  |  |  |  |  |  | x |

## 

## Bibliografía.

## 

Medrano, V. (2016). *[Cybertruco]Gestión básica de ficheros y directorios en Powershell - Ciberseguridad,seguridad informática,redes y programación.*. [online] Cyberseguridad.net. Available at: <https://cyberseguridad.net/index.php/490-cybertruco-gestion-basica-de-ficheros-y-directorios-en-powershell> [Accessed 26 Feb. 2020].

GitHub, @olprod. (2017). *Administrar la ubicación actual - PowerShell*. [online] Docs.microsoft.com. Available at: <https://docs.microsoft.com/es-es/powershell/scripting/samples/managing-current-location?view=powershell-7> [Accessed 26 Feb. 2020].

Docs.microsoft.com. (2017). *Trabajar con archivos y carpetas - PowerShell*. [online] Available at: <https://docs.microsoft.com/es-es/powershell/scripting/samples/working-with-files-and-folders?view=powershell-7#creating-files-and-folders> [Accessed 26 Feb. 2020].

Dawson, T. (2019). *Introducción a la línea de comandos de Windows con PowerShell*. [online] Programminghistorian.org. Available at: <https://programminghistorian.org/es/lecciones/introduccion-a-powershell#navegar-entre-directorios-con-set-location--sl-cd> [Accessed 26 Feb. 2020].