Asignatura	Datos del alumno	Fecha
Sistemas operativos – Laboratorio No. 2	Apellidos: De Mendoza Tovar	17/11/2024
	Nombre: Alejandro	

ACTIVIDAD LABORATORIO NO.2

MANEJO DE UN SISTEMA OPERATIVO

PRESENTADO POR:

ALEJANDRO DE MENDOZA

PRESENTADO AL PROFESOR:

ING. ALBERTO FERREIRA CASTRO

FUNDACIÓN UNIVERSITARIA INTERNACIONAL DE LA RIOJA

BOGOTÁ D.C.

17 DE NOVIEMBRE

2024

Asignatura	Datos del alumno	Fecha
Sistemas operativos – Laboratorio No. 2	Apellidos: De Mendoza Tovar Nombre: Alejandro	17/11/2024

TABLA DE CONTENIDO

Asignatura	Datos del alumno	Fecha
Sistemas operativos – Laboratorio No. 2	Apellidos: De Mendoza Tovar Nombre: Alejandro	17/11/2024

PREPARACIÓN DEL ENTORNO

Como preparación del entorno, es necesario indicar de mi parte que el entorno donde se va a desarrollar esta actividad es PowerShell en Windows, la cual es una herramienta que a través de una línea de comandos y el lenguaje Script diseña la administración de los sistemas y la automatización de tareas en una computadora. Ahora desde mi punto de vista se considera que PowerShell es una evolución de la herramienta CMD que ejecutamos en Windows y que se basa de igual manera en la ejecución de comandos. Además, PowerShell es una herramienta que podemos encontrar en las plataformas de Windows, macOS y Linux

DESARROLLO DEL LABORATORIO – EJECUCIÓN Y EXPLICACIÓN DE COMANDOS EN POWERSHELL

Entonces y con base en esto, para el desarrollo de la actividad de este laboratorio No. 2, me voy a centrar en: el desarrollo de comandos en la herramienta PowerShell (como se indicó anteriormente), en las aulas virtuales del profesor Ing. Alberto Ferreira, en la información del aula virtual de Sistemas Operativos, en los libros del aula y en los comandos explicativos adjuntos en la plataforma Padlet de la clase de Sistemas Operativos de la Fundación Universitaria Internacional de la Rioja, de nombres: Get-ComputerInfo, New-Item -ItemType Directory -Name "NombreDelDirectorio", Get-ChildItem, New-Item -ItemType File -Name "nombrearchivo.txt", Get-Process, Stop-Process -Name "nombreproceso", Restart-Computer -Force, Get-NetIPAddress, Test-Connection -ComputerName "direccionIpODominio", Get-LocalUser, Remove-Item -Path "nombredecarpeta" -Recurse -Force, Get-PSDrive -PSProvider FileSystem, Get-WmiObjectWin32_OperatingSystem|Select-Object@ {Name= "MemoriaLibreGB"; Expression= {"{0:N2}"-f ($_.FreePhysicalMemory/1MB)}}, y por último el comando Get-NetAdapter | Select-Object MacAddress, New-LocalUser -Name "nombreusuario" -Password (ConvertTo-SecureString -AsPlainText "Contraseña" -Force) -FullName "Nombre Completo". Ahora, dando inicio a la ejecución y desarrollo de la actividad lo primero es abrir la herramienta de PowerShell y para esto voy a Windows en Inicio, escribo PowerShell y aparece:

Ahora damos clic en abrir y nos aparece la herramienta PowerShell 5.1. Ahora y en mi caso se me pide actualizar a la versión más reciente de PowerShell, por ende, he procedido a generar la actualización de la herramienta, pero es un proceso que no voy a detallar en esta actividad ya que no lo considero relevante y ocupa demasiado espacio en el trabajo a presentar.

DESARROLLO DE LOS COMANDOS EN POWERSHELL

Ahora que ya tengo actualizada la herramienta de PowerShell a 7.4.6 (última versión), procedo a desarrollar los comandos respectivos de la actividad y el primero de ellos es Get-ComputerInfo que sustentaré a continuación.

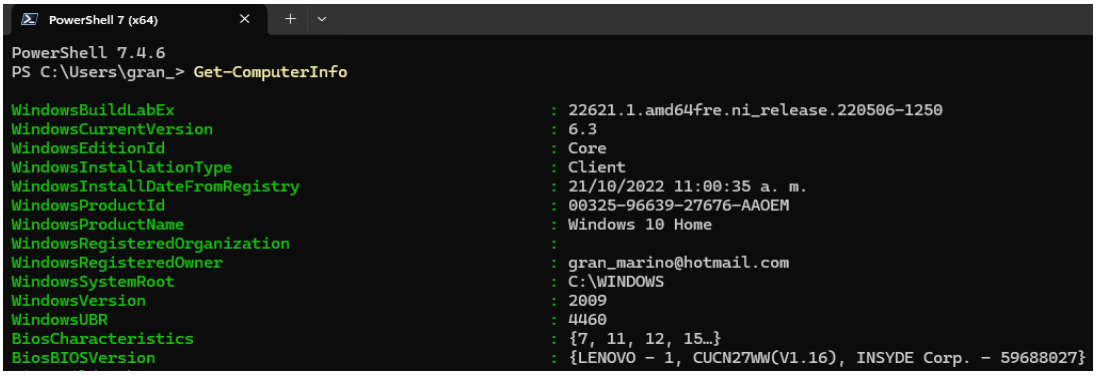
Comando Get-Computerinfo

Este primer comando se utiliza para obtener toda la información sobre el sistema de mi CPU. Esta herramienta es útil cuando quiero ver detalles sobre el tipo de Hardware que tengo, también mi sistema operativo, la red de mi sistema, entre otros., que a continuación voy a detallar:

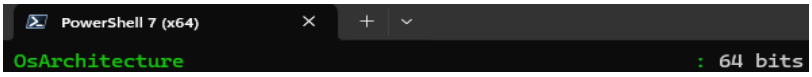
Asignatura	Datos del alumno	Fecha
Sistemas operativos – Laboratorio No. 2	Apellidos: De Mendoza Tovar Nombre: Alejandro	17/11/2024

- **Información del Sistema Operativo:**

En esta información podemos determinar la versión, el nombre, también podemos ver la arquitectura del sistema, el número de compilación, entre otros:

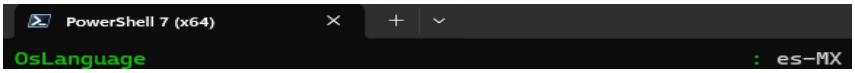


- ✓ OsArchitecture: Arquitectura del sistema (64-bits).



- ✓ OsName: Nombre del sistema operativo (Windows 11 Home).

- ✓ OsLanguage: Lenguaje configurado en el sistema (es-MX).



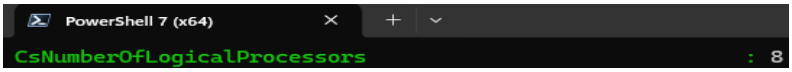
- ✓ OsBuild: El número de compilación 22631.

- **Información del Procesador:**

Nos suministra información o detalles sobre el procesador o procesadores de mi CPU, como el modelo, la cantidad de procesadores, entre otros que a continuación denoto:

- ✓ CsProcessors: Procesadores administrados por el sistema (Intel Core™ i7-1065G7…):

- ✓ CsNumberOfLogicalProcessors: Cantidad de procesadores lógicos en mi sistema (8):



- ✓ CsNumberOfProcessors: Cantidad de procesadores físicos en mi sistema (1):

- **Información de la Memoria RAM:**

Me muestra los detalles sobre la memoria física disponible y utilizada.

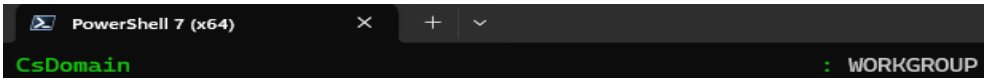
- ✓ CsTotalPhysicalMemory: Total de la memoria disponible para el sistema operativo, incluyendo memoria reservada o no utilizable.

✓ CsPhysicallyInstalledMemory: Cantidad de memoria RAM física que está realmente instalada en el hardware.

- **Información de la Red:**

Me muestra los detalles sobre la configuración de red, como las interfaces de red y las direcciones IP.

✓ CsDomain: Dominio al que está unida la computadora, si está unida a un dominio de red.

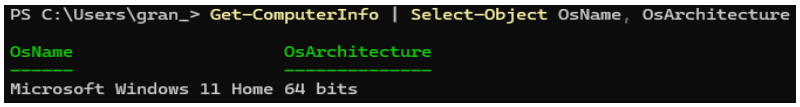


✓ CsDomainRole: Rol de la máquina en la red, "workstation", "server", "member server"…

✓ CsNetworkAdapters: Muestra los adaptadores de red disponibles en el sistema.

- **Filtrar la salida:**

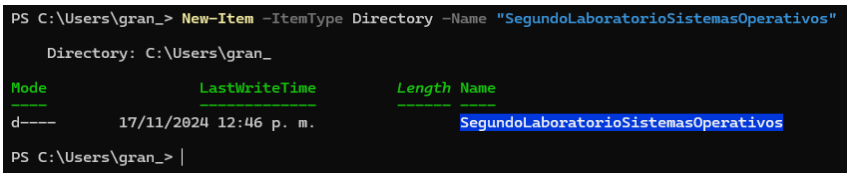
Cuando ejecuto este comando "Get-ComputerInfo" la información puede ser extensa, por lo que puedo filtrarla usando Select-Object o Where-Object, esto, si quiero obtener solo el nombre del sistema operativo y la arquitectura, entonces al aplicar "Get-ComputerInfo | Select-Object OsName, OsArchitecture", me muestra la siguiente imagen segmentada con la información que requiero:



Para concluir Get-ComputerInfo ofrece información básica sobre la red, como el dominio, el rol del equipo, los adaptadores de red y su información puede ser filtrada para mayor facilidad.

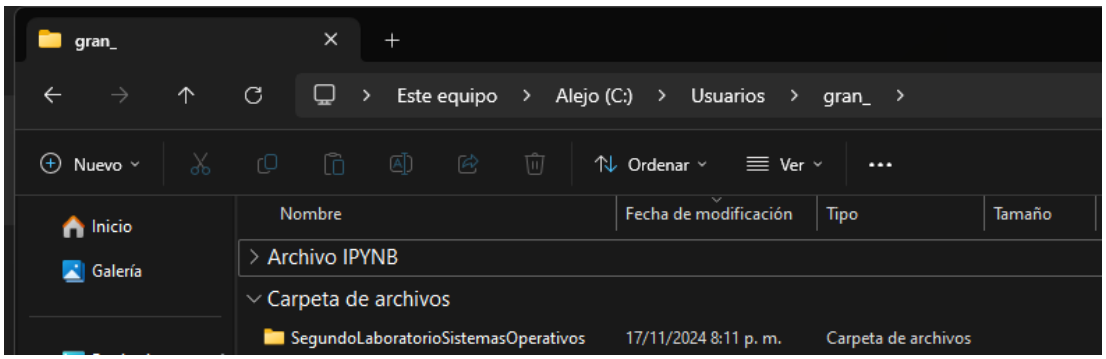
Comando New-Item -ItemType Directory -Name "NombreDelDirectorio"

Este comando se utiliza para crear un nuevo directorio que en este caso lo voy a nombrar como "SegundoLaboratorioSistemasOperativos". A continuación, se muestra la imagen:



Ahora procediendo a denotar la explicación: New-Item: Permite crear un nuevo elemento en un sistema de archivos u otras ubicaciones compatibles con PowerShell (como el registro de Windows o certificados). ItemType Directory: Especifica el tipo de elemento a crear. Al usar "Directory", se indica que el comando debe crear una carpeta o un directorio. Name "NombreDelDirectorio": Define el nombre del directorio que se va a crear y en este caso lo denote con el nombre de "SegundoLaboratorioSistemasOperativos". A continuación, la imagen respectiva del directorio o carpeta creada:

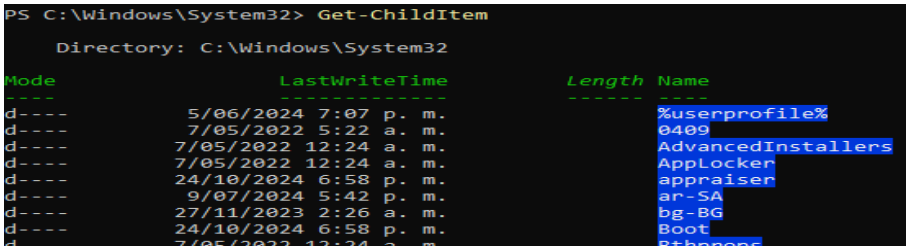
Asignatura	Datos del alumno	Fecha
Sistemas operativos – Laboratorio No. 2	Apellidos: De Mendoza Tovar	17/11/2024
	Nombre: Alejandro	



Ahora, si se desea crear el directorio en una ubicación específica, se utiliza el parámetro -Path. Un ejemplo entonces sería: New-Item -ItemType Directory -Path "C:\MisCarpetas" -Name "Proyectos". Y entonces se crearía la carpeta en C:\MisCarpetas\Proyectos.

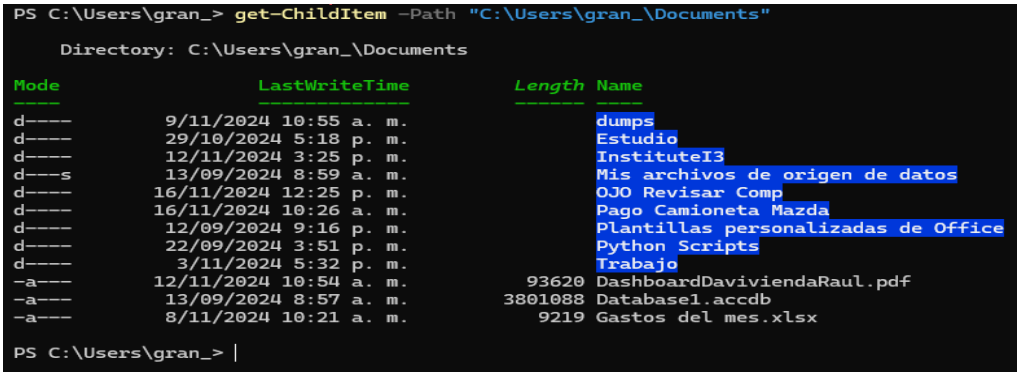
Comando Get-ChildItem

Este comando lo utilizo para listar los elementos dentro de un directorio, como archivos y carpetas, o para enumerar elementos en otras ubicaciones compatibles con PowerShell. Por defecto, este comando me muestra los elementos del directorio actual en el que yo me encuentro trabajando, que en este momento estoy en Windows:



Ahora en este comando es importante precisar los diferentes parámetros:

- **Path**: Especifica la ruta del directorio o ubicación que de mi parte se desea inspeccionar, en mi caso modifique el path a "Documents", el comando entonces queda: Get-ChildItem -Path "C:\Users\gran_\Documents".



- **Filter**: Filtra los resultados por el tipo de archivo específico, como en este caso, archivos de tipo texto: Get-ChildItem -Filter "*.txt".

```
PS C:\Users\gran_> Get-ChildItem -Filter "*.txt"

    Directory: C:\Users\gran_

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---         15/11/2024 6:26 p. m.              0 sistemasoperativos.txt
```

- **Recurse**: Incluye los subdirectorios en la búsqueda. Adicional me permite listar archivos y carpetas en todos los niveles dentro del directorio especificado y todos los elementos en el directorio actual y sus subdirectorios. En este caso voy a utilizar los de Java – Eclipse con el comando Get-ChildItem -Path "C:\Users\gran_\.eclipse"-Recurse, como ejemplo.

```
PS C:\Windows\System32> Get-ChildItem -Path "C:\Users\gran_\.eclipse"-Recurse

    Directory: C:\Users\gran_\.eclipse

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d----          8/09/2024 5:02 p. m.                org.eclipse.oomph.jreinfo
d----         26/10/2024 10:05 a. m.               org.eclipse.oomph.p2
d----          8/09/2024 5:02 p. m.                org.eclipse.oomph.setup
d----          8/09/2024 5:03 p. m.                org.eclipse.oomph.setup.installer

    Directory: C:\Users\gran_\.eclipse\org.eclipse.oomph.jreinfo

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---          8/09/2024 5:02 p. m.            143 infos.txt

    Directory: C:\Users\gran_\.eclipse\org.eclipse.oomph.p2

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d----          8/09/2024 5:03 p. m.                cache
d----          8/09/2024 5:06 p. m.                trust
-a---          8/09/2024 5:02 p. m.             20 agents.info
-a---          8/09/2024 5:03 p. m.            206 defaults.info
-a---          8/09/2024 5:02 p. m.             14 encoding.info

    Directory: C:\Users\gran_\.eclipse\org.eclipse.oomph.p2\cache
```

- **Name**: Muestra los nombres de los archivos y carpetas, sin detalles adicionales. El comando es: Get-ChildItem -Path "C:\Users\gran_\.eclipse"-Name

```
PS C:\Users\gran_> Get-ChildItem -Path "C:\Users\gran_\.eclipse" -Name
org.eclipse.oomph.jreinfo
org.eclipse.oomph.p2
org.eclipse.oomph.setup
org.eclipse.oomph.setup.installer
PS C:\Users\gran_>
```

- **File**: Muestra solo archivos, excluyendo carpetas. El comando es: Get-ChildItem -Path "C:\Users\gran_\.anaconda"-File, en este caso como ejemplo utilizo Anaconda para Python:

```
PS C:\Windows\System32> Get-ChildItem -Path "C:\Users\gran_\.anaconda"-File

    Directory: C:\Users\gran_\.anaconda

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---        10/10/2024 11:03 p. m.          10960 assistant.json
-a---         22/09/2024 4:06 p. m.            897 keyring
```

- **Directory**: Muestra solo directorios, excluyendo archivos. El comando es: Get-ChildItem -Path "C:\Users\gran_\anaconda3"-Directory

```
PS C:\Users\gran_> Get-ChildItem -Path "C:\Users\gran_\anaconda3"-Directory

    Directory: C:\Users\gran_\anaconda3

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d----          3/09/2024 7:27 a. m.                conda-meta
d----          3/09/2024 7:27 a. m.                condabin
d----          3/09/2024 7:11 a. m.                DLLs
d----          3/09/2024 11:25 a. m.               envs
d----          3/09/2024 7:13 a. m.                etc
d----          3/09/2024 7:11 a. m.                include
d----          3/09/2024 7:11 a. m.                Lib
d----          3/09/2024 7:11 a. m.                Library
d----          3/09/2024 7:11 a. m.                libs
d----          3/09/2024 7:12 a. m.                Menu
d----          3/09/2024 11:25 a. m.               pkgs
d----          3/09/2024 7:27 a. m.                Scripts
d----          3/09/2024 7:12 a. m.                share
d----          3/09/2024 7:11 a. m.                shell
d----          3/09/2024 7:11 a. m.                Tools
```

- **Hidden**: Incluye elementos ocultos en los resultados, que no hay en este caso en la carpeta. El comando es: Get-ChildItem -Path “C:\Users\gran_\anaconda3“-Hidden

```
PS C:\Users\gran_> Get-ChildItem -Path "C:\Users\gran_\anaconda3" -Hidden
PS C:\Users\gran_>
```

- **Force**: Muestra todos los elementos, incluidos los ocultos y los de sistema, que normalmente están excluidos. El comando es: Get-ChildItem -Path “C:\Users\gran_\anaconda3“-Force

```
PS C:\Users\gran_> Get-ChildItem -Path "C:\Users\gran_\anaconda3"-Force

    Directory: C:\Users\gran_\anaconda3

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d----          3/09/2024 7:27 a. m.                conda-meta
d----          3/09/2024 7:27 a. m.                condabin
d----          3/09/2024 7:11 a. m.                DLLs
d----          3/09/2024 11:25 a. m.               envs
d----          3/09/2024 7:13 a. m.                etc
d----          3/09/2024 7:11 a. m.                include
d----          3/09/2024 7:11 a. m.                Lib
d----          3/09/2024 7:11 a. m.                Library
d----          3/09/2024 7:11 a. m.                libs
d----          3/09/2024 7:12 a. m.                Menu
d----          3/09/2024 11:25 a. m.               pkgs
d----          3/09/2024 7:27 a. m.                Scripts
d----          3/09/2024 7:12 a. m.                share
d----          3/09/2024 7:11 a. m.                shell
d----          3/09/2024 7:11 a. m.                Tools
-a---         26/06/2024 12:21 p. m.       25160208 _conda.exe
-a---          3/09/2024 7:27 a. m.              0 .nonadmin
-a---          9/07/2015 5:25 p. m.          19136 api-ms-win-core-console-l1-1-0.dll
```

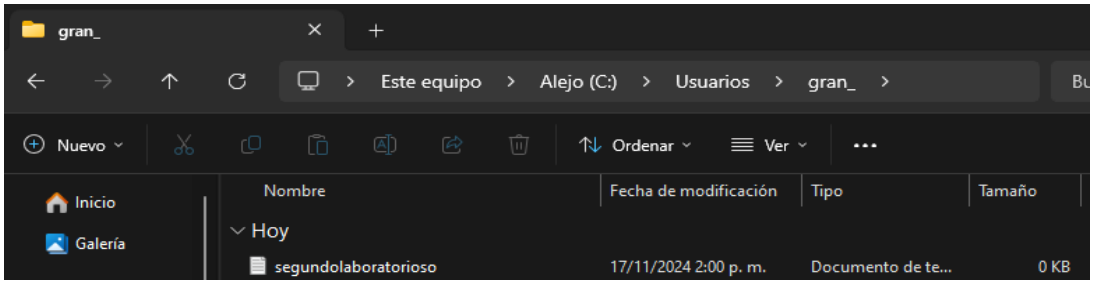
Ahora para finalizar como se puede denotar el comando muestra información estructurada en columnas: **Mode**: Indica el tipo y los atributos del elemento (“d” para directorios y “a” para archivos). **LastWriteTime**: Fecha y hora de la última modificación del elemento. **Length**: Tamaño del archivo en bytes (solo para archivos). **Name**: Nombre del archivo o carpeta.

Comando New-Item -ItemType File -Name "nombrearchivo.txt"

Este comando se utiliza para crear nuevos elementos en el sistema de archivos, como archivos o carpetas. Tener presente que el nombre del archivo que he creado es “**segundolaboratorioso**”:

```
PS C:\Users\gran_> New-Item -ItemType File -Name "segundolaboratorioso.txt"

    Directory: C:\Users\gran_

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---         17/11/2024 2:00 p. m.              0 segundolaboratorioso.txt
```

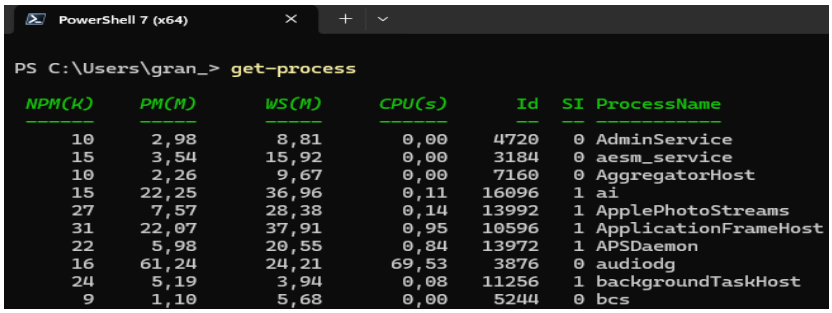
A continuación, procedo a explicar los parámetros del comando: **New-Item**: Esta parte del comando crea un nuevo elemento en el sistema de archivos o en otra ubicación compatible con PowerShell. **ItemType File**: Indica que el tipo de elemento que se va a crear es un archivo. **Name** "nombrearchivo.txt": Especifica el nombre del archivo que se va a crear. En este caso, el archivo se llama segundolaboratorioso.txt. En conclusión, con este comando se creó un archivo vacío (si se quiere agregar información se utiliza set-content o add-content), llamado segundolaboratorioso.txt en el siguiente directorio:



Importante tener presente que, si el archivo ya existe en esta ubicación, PowerShell generará un error.

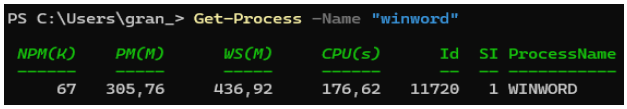
Comando Get-Process

El comando Get-Process se utiliza para obtener información sobre los procesos que están actualmente en ejecución en el sistema operativo. Esto incluye detalles como el nombre del proceso, el ID del proceso, la cantidad de memoria que utiliza, entre otros. A continuación, la imagen respectiva:

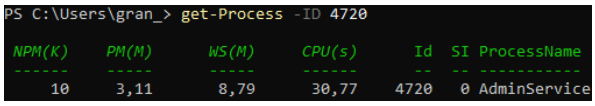


Con base en esto entonces los parámetros son los siguientes:

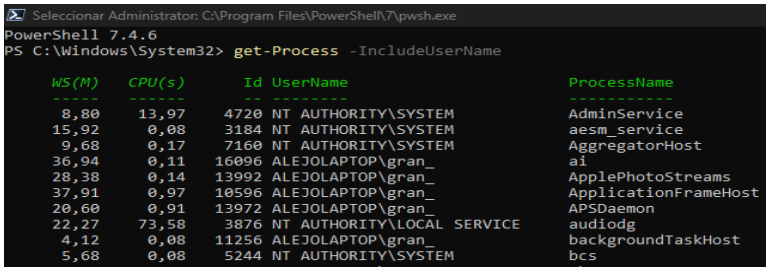
- **Get-Process**: Me permite mostrar la información sobre todos los procesos en ejecución en el sistema operativo, incluyendo su nombre, ID de proceso (PID), la cuenta de usuario que los inició, el uso de CPU y memoria, y otros detalles clave (imagen de arriba).
- **Name**: Me permite especificar un filtro para mostrar solo los procesos con un nombre específico. Un ejemplo puede ser WINWORD, a continuación, la imagen explicativa:



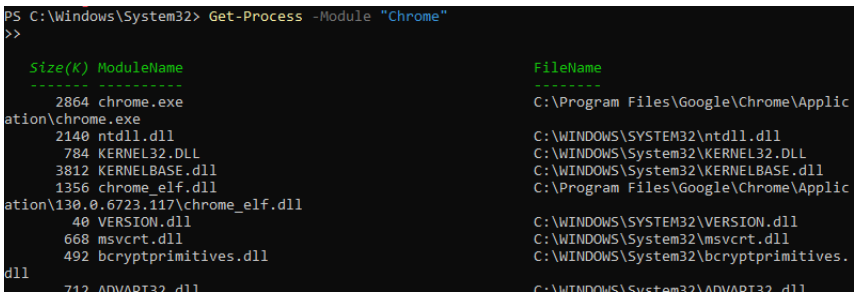
- **Id**: Muestra detalles específicos de un proceso basado en su ID de proceso. Un ejemplo puede ser el ID 4720 del primero proceso mostrado en la lista, entonces el comando sería el siguiente: Get-Process -Id 4720, que es de AdminService:



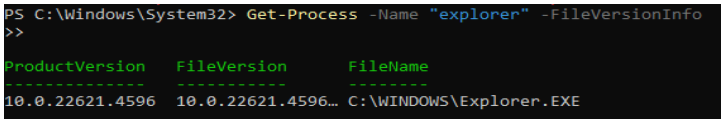
- **IncludeUserName**: Incluye el nombre de usuario que inició el proceso en la salida:



- **Module**: Muestra procesos pertenecientes solo a módulos especificados. Útil para sistemas distribuidos donde se desea limitar la salida a procesos de un módulo específico. A continuación, la imagen respectiva tomando como ejemplo el proceso de Chrome:



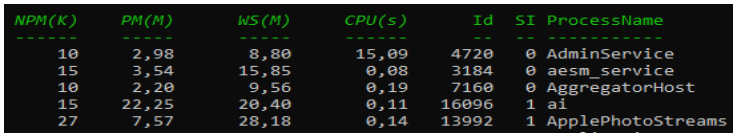
- **FileVersionInfo**: Muestra información sobre la versión de los archivos ejecutables asociados a los procesos. Tomo como ejemplo el archivo de nombre “explorer”:

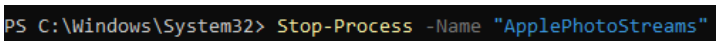


Es por esto que este comando, considero, es muy útil para un monitoreo en tiempo real del rendimiento del sistema, ejecutar un diagnóstico de procesos que consumen demasiados recursos y desarrollar una correcta administración remota de procesos en servidores o equipos cliente.

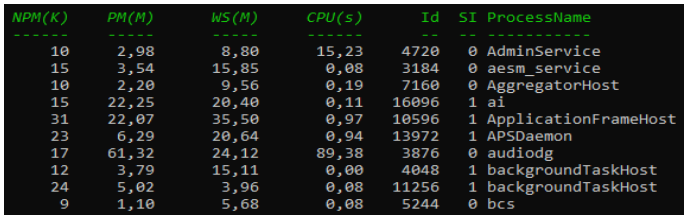
Comando Stop-Process -Name "nombreproceso"

Este comando lo utilizo para frenar o detener uno o varios procesos en ejecución en el sistema operativo. El parámetro -Name permite especificar el nombre del proceso que deseo detener. A continuación, la imagen del proceso de nombre “ApplePhotoStreams“, que voy a detener:





Y al correr de nuevo el Get-Process, como podemos denotar a continuación el proceso ya no aparece, por lo que se detuvo su ejecución:



Ahora, podemos detener un proceso por su ID, y podemos forzar u obligar al CPU a detener un proceso (tener cuidado ya que al detenerlo no guarda cambios), incluso si este no responde o tiene restricciones, y adicionalmente tenemos el -WhatIf que muestra lo que sucedería si ejecutas el comando, sin detener realmente el proceso. Este comando se utiliza más que todo para temas de prueba. Por último, considero que, para detener un proceso, se necesitan los permisos adecuados, pero sobre todo hay que tener cuidado al detener procesos del sistema o esenciales, ya que puede causar inestabilidad en el sistema.

Comando Restart-Computer -Force

Este comando lo utilizo para reiniciar mi CPU. La opción -Force me asegura que el sistema se reinicie inmediatamente, sin solicitar confirmaciones adicionales ni cerrar aplicaciones de manera controlada.

```
PS C:\Windows\System32> Restart-Computer -Force
```

Ahora con este comando puedo: "**Reiniciar una computadora remota"** indicando el nombre de la CPU y el comando es: Restart-Computer -ComputerName "NombreComputadora" -Force. "**Reiniciar varias computadoras al mismo tiempo**", separando sus nombres por comas y entre comillas. El comando es: Restart-Computer -ComputerName "PC1", "PC2", "PC3" -Force. "**Esperar hasta que la computadora esté nuevamente disponible**" y para esto utilizo el parámetro -Wait para que PowerShell espere a que el sistema esté accesible después del reinicio. El comando es: Restart-Computer -Force -Wait. Y por último puedo "**especificar credenciales para computadoras remotas"** si necesito autenticarme. El comando es: $cred = Get-Credential Restart - Computer -ComputerName "NombreComputadora" -Credential $cred -Force. Ahora, el uso de -Force puede forzar el cierre de aplicaciones abiertas sin guardar cambios, lo que podría provocar pérdida de datos. Por esto considero es un comando de cuidado, especialmente en entornos de producción o con aplicaciones críticas abiertas.

Comando Get-NetIPAddress

Este comando lo utilizó para obtener las direcciones IP asociadas a los adaptadores de red de mi CPU, y lo que hace es mostrar información sobre las direcciones IP configuradas en mi sistema, incluyendo IPv4 e IPv6, la interfaz de red, y detalles como la máscara de subred y el estado:

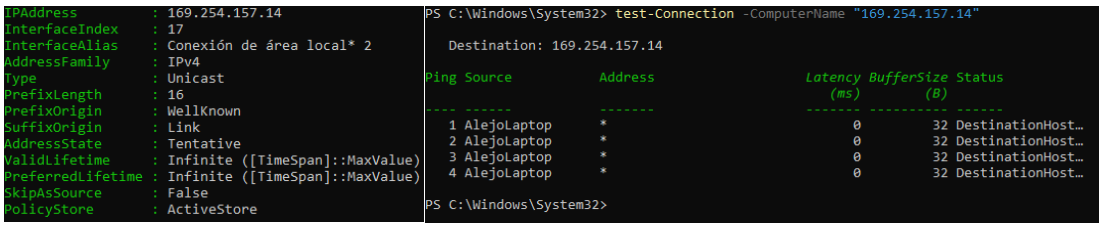
```
PS C:\Windows\System32> Get-NetIPAddress
>>

IPAddress         : fe80::bced:ba53:9e:4a69%17
InterfaceIndex    : 17
InterfaceAlias    : Conexión de área local* 2
AddressFamily     : IPv6
Type              : Unicast
PrefixLength      : 64
PrefixOrigin      : WellKnown
SuffixOrigin      : Link
AddressState      : Deprecated
ValidLifetime     : Infinite ([TimeSpan]::MaxValue)
PreferredLifetime : Infinite ([TimeSpan]::MaxValue)
SkipAsSource      : False
PolicyStore       : ActiveStore

IPAddress         : fe80::5ec3:b9d9:1de8:b94d%14
InterfaceIndex    : 14
InterfaceAlias    : Conexión de área local* 1
```

Ahora con base en los parámetros estos pueden ser: **"Filtrar por IPv4 solamente"**. Y para esto utilizo el comando: Get-NetIPAddress -AddressFamily IPv4. **"Filtrar por IPv6 solamente"**. Y para esto utilizo el comando: Get-NetIPAddress -AddressFamily IPv6. "**Obtener direcciones IP de un adaptador específico**", y en este caso puedo hacerlo por el *nombre del adaptador*, cuyo comando es Get-NetIPAddress -InterfaceAlias "Wi-Fi" y por el *índice del interfaz*, cuyo comando es Get-NetIPAddress -InterfaceIndex 3. **"Filtrar las direcciones activas únicamente**". Y para esto utilizo el comando: Get-NetIPAddress | Where-Object {$. Status -eq "Up"}. **"Mostrar solo campos específicos**" (ejemplo: IP y máscara de subred). Y para esto utilizo el comando: Get-NetIPAddress | Select-Object InterfaceAlias, IPAddress, PrefixLength. Por último, "**exportar la información a un archivo**" ejecutando el siguiente comando: Get-NetIPAddress | Export-Csv -Path "C:\ruta\direcciones_ip.csv" -NoTypeInformation

Comando Test-Connection -ComputerName "direccionIpODominio"

Esto comando lo utilizo para probar la conectividad de red entre mi CPU y otro dispositivo, es una sentencia similar a la ejecución del comando ping en CMD, y para esto utilizo mi IPv4, a continuación las dos imágenes:

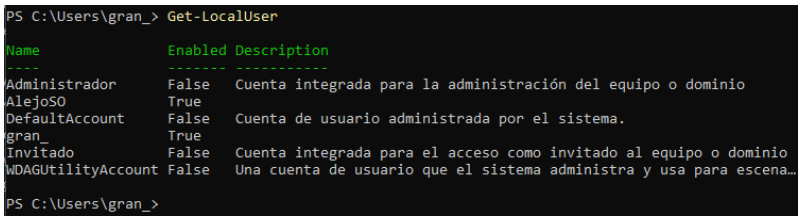
Asignatura	Datos del alumno	Fecha
Sistemas operativos – Laboratorio No. 2	Apellidos: De Mendoza Tovar Nombre: Alejandro	17/11/2024



Entonces como se puede denotar este comando envía paquetes (ping) al destino especificado (ya sea una dirección IP o un nombre de dominio) y muestra los resultados de la prueba. Ahora es importante indicar que en caso que no tenga el numero el IPv4, puedo ***"probar la conectividad con un nombre de dominio como Google"***, para lo que ejecutaría este comando: Test-Connection -ComputerName "www.google.com". También puedo "***especificar el número de pings"*** que por defecto son 4 pero lo puedo subir a 10 con el siguiente comando: Test-Connection -ComputerName "169.254.157.14" -Count 10. De igual manera puedo ***"obtener el tiempo de respuesta sin detalles adicionales"*** con el siguiente comando: Test-Connection -ComputerName "169.254.157.14" -Quiet. Y por último puedo ***"probar una conexión en un puerto específico como Windows Server"***, con el siguiente comando: Test-Connection -ComputerName "169.254.157.14" -Port 80. Como conclusión, la salida de este comando muestra información sobre el número de paquetes enviados, recibidos, el porcentaje de pérdida de paquetes y el tiempo de respuesta promedio, por lo cual considero es eficiente a la hora de revisar conectividad.

Comando Get-LocalUser

Se utiliza para obtener información sobre las cuentas de usuario locales en tu sistema. Muestra una lista de todas las cuentas de usuario que están configuradas en mi CPU, incluyendo detalles como el nombre de usuario, el estado de la cuenta (habilitada o deshabilitada), y si está bloqueada o no:

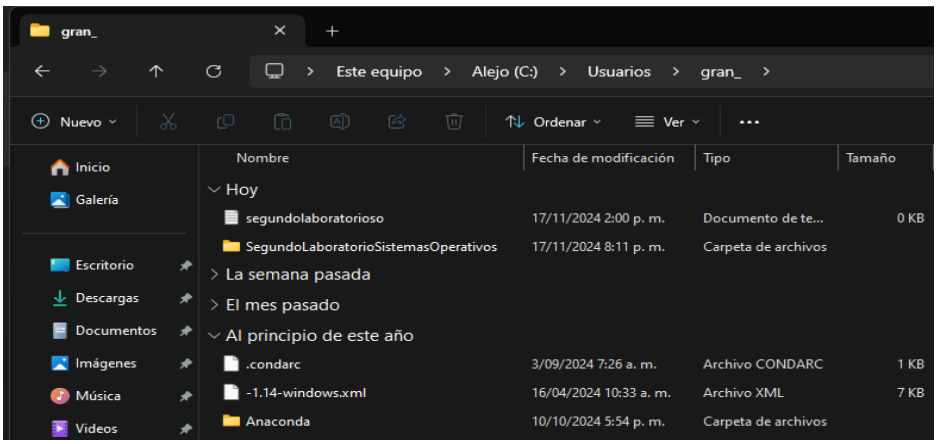


Ahora con este comando puedo: "***ver los detalles de una cuenta específica"*** como por ejemplo la cuenta de "**administrador"** con el siguiente comando: Get-LocalUser -Name "Administrador". También puedo "***filtrar las cuentas habilitadas o deshabilitadas"*** con este comando: Get-LocalUser | Where-Object {$_.Enabled -eq $true}, y por último "***mostrar solo el nombre de los usuarios locales"*** con el siguiente comando: Get-LocalUser | Select-Object Name. Es importante precisar que, al ejecutar, se muestra una tabla con detalles vistos (imagen de arriba), como: **Name**: El nombre de la cuenta de usuario. **Enabled**: Indica si la cuenta está habilitada y **Description**: Descripción de la cuenta (si está configurada).

Comando Remove-Item -Path "nombredecarpeta" -Recurse -Force

Este comando se utiliza para eliminar archivos, carpetas y otros elementos. La opción **-Recurse** permite eliminar carpetas y su contenido de forma recursiva, mientras que **-Force** permite forzar la eliminación, incluso de archivos ocultos o de solo lectura. A continuación, las imagenes respectiva, donde voy a eliminar la carpeta "SegundoLaboratorioSistemasOperativos":

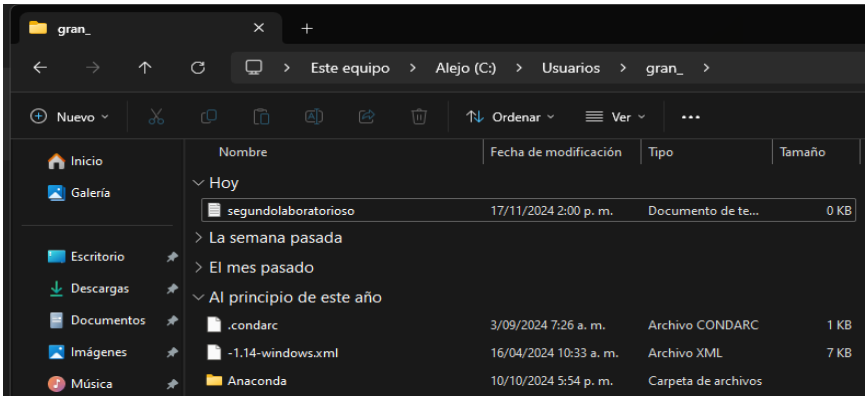
- **Imagen antes de la ejecución del comando:**



- **Comando a ejecutar:**

```
PS C:\Windows\System32> Remove-Item -Path "C:\Users\gran_\SegundoLaboratorioSistemasOperativos" -Recurse -Force
```

- **Después de ejecutar el comando:**



Y como se puede denotar la carpeta ya no existe o fue eliminada. Ahora, el uso de -Recurse y -Force pueden eliminar permanentemente los datos sin posibilidad de recuperación, especialmente si se ejecuta en carpetas importantes, entonces hay que revisar bien la ruta antes de ejecutar.

Comando Get-PSDrive -PSProvider FileSystem

Este comando se utiliza para mostrar todos los discos (unidades) y carpetas mapeadas que están disponibles en mi sistema y que usan el proveedor de FileSystem. El proveedor de FileSystem se refiere a los sistemas de archivos locales, como discos duros, unidades de red y otros recursos relacionados con almacenamiento en mi CPU. A continuación, la imagen respectiva:

```
PS C:\Windows\System32> Get-PSDrive -PSProvider FileSystem
>>

Name           Used (GB)     Free (GB) Provider      Root
----           ---------     --------- --------      ----
C                 195,57        279,80 FileSystem    C:\
Temp              195,57        279,80 FileSystem    C:\Users\gran_\AppData\Local\Temp\

PS C:\Windows\System32>
```

Entonces como explicación; **Get-PSDrive**: Obtiene la lista de unidades (drives) disponibles. Y, por otra parte **-PSProvider FileSystem**: Filtra los resultados para mostrar solo los elementos que pertenecen al proveedor de sistema de archivos. Ahora y para culminar es importante indicar que la salida de este comando incluye columnas como: **Name**: El nombre de la unidad (por ejemplo, C: D:). **Used (GB)**: La cantidad de espacio usado. **Free (GB)**: La

cantidad de espacio libre disponible. **Provider**: El tipo de proveedor (en este caso, FileSystem). **Root**: El directorio raíz de la unidad.

Comando Get-WmiObject Win32_OperatingSystem | Select-Object @ {Name = "MemoriaLibreGB"; Expression= {"{0: N2}"-f($_.FreePhysicalMemory/1MB)}}

Este comando es una forma de obtener la cantidad de memoria libre en el sistema. Este comando convierte la memoria libre de bytes a gigabytes (GB) y la muestra con dos decimales.:

```
PS C:\Users\gran_> Get-WmiObject Win32_OperatingSystem | Select-Object @{Name=
"MemoriaLibreGB";Expression={"{0:N2}"-f($_.FreePhysicalMemory/1MB)}}
>>

MemoriaLibreGB
--------------
10,03

PS C:\Users\gran_>
```

Ahora como podemos denotar este comando indica que hay 10.03 gigas en mi CPU de memoria libre en el sistema.

Comando Get-NetAdapter | Select-Object MacAddress

Este comando se utiliza para obtener la dirección MAC de todos los adaptadores de red en mi sistema. Sin embargo, es importante recordar que la dirección MAC es única para cada adaptador de red, y este comando me devuelve una lista de esas direcciones. A continuación, la imagen respectiva:

```
PS C:\Windows\System32> Get-NetAdapter | Select-Object MacAddress
>>

MacAddress
----------
E4-AA-EA-CB-92-E9
E4-AA-EA-CB-92-EA

PS C:\Windows\System32>
```

Como se puede denotar, cada dirección MAC corresponde a un adaptador de red en mi dispositivo. En caso que tenga varios adaptadores (uno para Wi-Fi y otro para Ethernet), me mostrará varias direcciones MAC.

Comando New-LocalUser -Name "nombreusuario" -Password (ConvertTo-SecureString -AsPlainText "Contraseña" -Force) -FullName "Nombre Completo"

Este comando crea un usuario local en mi sistema, que en este caso voy a llamarlo como **AlejoSO**. A continuación, la imagen respectiva:

```
PS C:\Windows\System32> Import-Module Microsoft.PowerShell.LocalAccounts -UseWindowsPowerShell
>> New-LocalUser -Name "AlejoSO" -Password (ConvertTo-SecureString -AsPlainText "12345" -Force) -Fu
llName "Alejo De Mendoza"
>>
WARNING: Module Microsoft.PowerShell.LocalAccounts is loaded in Windows PowerShell using WinPSCompa
tSession remoting session; please note that all input and output of commands from this module will
be deserialized objects. If you want to load this module into PowerShell please use 'Import-Module
-SkipEditionCheck' syntax.

Name    Enabled Description
----    ------- -----------
AlejoSO True
```

Es importante indicar que en este caso **debí importar el módulo de compatibilidad de Windows PowerShell usando Import-Module**, o trae un error en donde indica que para esta creación se debe ejecutar PowerShell 5.1 y no el 7.4.6, lo que claramente se puede hacer y no debe traer ningún problema, pero, yo de mi parte preferí ejecutarlo en PowerShell 7. Por último y para culminar a continuación voy a explicar los parámetros del comando: **New-LocalUser**: Se utiliza para crear un nuevo usuario local en el sistema. **Name "nombreusuario"**: Especifica el nombre de usuario para la nueva cuenta que en este caso es AlejoSO. **Password (ConvertTo-SecureString -AsPlainText "Contraseña" -Force)**: Define la contraseña del usuario que en este caso es “12345”. **ConvertTo-SecureString:** convierte la contraseña (en texto plano) a un formato seguro que PowerShell puede usar.

AsPlainText: Indica que estás ingresando la contraseña en texto plano. **Force**: omite advertencias sobre usar contraseñas en texto plano. **FullName "Nombre Completo"**: Define el nombre completo del usuario para fines descriptivos y en este caso es Alejo De Mendoza.

ANÁLISIS DE RESULTADOS

En este estudio, he explorado diversas herramientas y comandos en PowerShell, abordando temas como administración del sistema, red, usuarios y discos. Ahora, PowerShell es una herramienta poderosa para administrar sistemas operativos Windows (y ahora multiplataforma) mediante comandos y scripts. Su versatilidad radica en: Automatización de tareas, Administración avanzada del sistema e Integración con otros servicios y herramientas. Comandos como New-LocalUser, Get-LocalUser, y Remove-LocalUser me permiten administrar cuentas de usuarios locales, desde la creación hasta la eliminación, así como verificar el estado de habilitación/deshabilitación de sus cuentas. Además, algunos comandos como New-LocalUser, están diseñados para Windows PowerShell (5.1) y es por esto que PowerShell 7 puede requerir módulos adicionales o usar métodos alternativos para la misma funcionalidad. Por otra parte, comandos como Test-Connection, Get-NetAdapter, y Get-NetIPAddress proporcionan información detallada sobre conectividad, adaptadores y direcciones IP, además que PowerShell permite probar conexiones con protocolos como ICMP (ping) y TCP y es por esto que considero que es crucial comprender la configuración de red local y asegurarme de que no haya bloqueos (como en ICMP) para ejecutar correctamente algunos comandos. Ahora, con Get-ComputerInfo, puedo obtener una vista general del sistema como detalles del hardware: procesadores, memoria física y lógica. Configuración del sistema operativo: versión, espacio en disco, entre otros. Y es un comando que puede ser filtrado para tener facilidad en la obtención de la información que requiero. Para finalizar y no extenderme demasiado preciso indicar que PowerShell no es solo una herramienta; es un lenguaje de scripting robusto que me permite optimizar la administración del sistema, automatizar tareas repetitivas u incrementar la eficiencia en la gestión de redes y servidores, lo que la hace una herramienta esencial en mi campo de la ingeniería informática.

BIBLIOGRAFÍA

A continuación, la bibliografía implementada:

- ✓ Tema 7: Gestión De Memoria. Agosto 2024 2Q.
- ✓ Clases virtuales con el profesor Ing. Alberto Ferreira Castro.
- ✓ Silberschatz, A. (2006). Fundamentos de sistemas operativos (pp. 252-261). Madrid: McGraw-Hill. Disponible en el aula virtual en virtud del artículo 32.4 de la Ley de Propiedad Intelectual.
- ✓ Nutt, G. (2004). Sistemas operativos (pp. 382-387). Madrid: Addison-Wesley. Disponible en el aula virtual en virtud del artículo 32.4 de la Ley de Propiedad Intelectual.
- ✓ Carretero, J. (2007). Sistemas operativos. Una visión aplicada (pp. 312-316). Madrid: McGraw-Hill. Disponible en el aula virtual en virtud del artículo 32.4 de la Ley de Propiedad Intelectual.
- ✓ Plataforma Padlet, Información brindada por el profesor Ing. Alberto Castro Laerte.