**UNIVERSIDAD DE LAS FUERZAS ARMADAS-ESPE SEDE SANTO DOMINGO**

**DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN - DCCO-SS**

**CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN**

**PERIODO** : 202450

**ASIGNATURA** : Sistemas Operativos

**TEMA** : Laboratorio 1\_U2

**NOMBRES** : Juan David Jiménez Romero

**NIVEL-PARALELO**  : 15310

**DOCENTE** : Ing. Javier Cevallos. MSc.

**FECHA DE ENTREGA** : 04/07/2024

**SANTO DOMINGO - ECUADOR**

**2023**

Contenido

1.[Introducción 3](#_Toc171020618)

[Sistemas de Objetivos 3](#_Toc171020619)

[Objetivo general del trabajo: 3](#_Toc171020620)

2. Desarrollo………………………………………………………………………………………….4

[3. Conclusiones 16](#_Toc171020621)

[4. Recomendaciones 16](#_Toc171020622)

[5. Bibliografía/ Referencias 17](#_Toc171020623)

6.[Bibliografía 17](#_Toc171020624)

[7. Anexos 18](#_Toc171020625)

# Introducción

En el presente informe de laboratorio se revisará los pasos que se realizaron para la elaboración del laboratorio que se realizó el día 04/07/2024, el laboratorio consiste en: La comparación de entre las líneas de comando para revisar los procesos en Windows y Linux (Debian) que nos permitirá revisar de mejor manera cuantos recursos ocupan cada uno, además de realizar programación en archivos .sh en el terminal de Linux.

Un terminal es un dispositivo de hardware, ya sea de naturaleza electromecánica o electrónica, que se puede usar tanto para ingresar como para transcribir información. Esas tareas se pueden llevar a cabo bien desde un ordenador o echando mano de un sistema informático. (Desarrollo, 2022)

# Sistemas de Objetivos

Objetivo General:

## Objetivo general del trabajo:

La revisión de los comandos en windows y linux para revisar los procesos que se están ejecutando y como detenerlos o iniciarlos además de ver que tanto recurso estos usan.

Objetivos Específicos:

* + 1. Objetivo específico 1

Realizar funciones dentro del terminal de Debian que realice actividades con el solo hecho de ejecutarlos.

* + 1. Objetivo específico 2

Realizar la comparación entre los comandos para revisar los procesos dentro de Windows PowerShell y Terminal.

1. **Desarrollo**

Ejercicios:

1. Contar el número de procesos que se están ejecutando: (Get-Process).count:



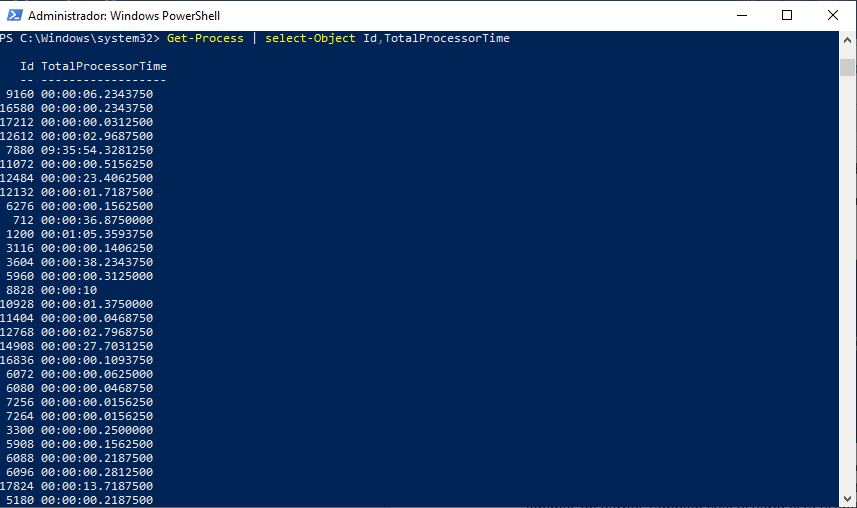
Este comando “(ps).count” muestra el valor numérico de los procesos que se están ejecutando en el momento que se envie el comando.

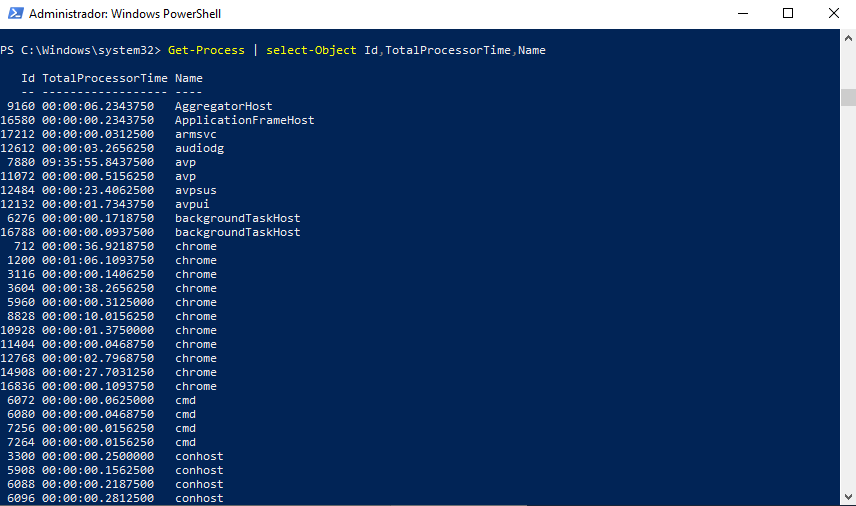
1. Ver los programas que se arrancan cuando un usuario inicia sesión: Get-WmiObject -Class win32\_startupCommand:



Con el comando “Get-WmiObject –Class win32\_startupCommand” este comando sirve para mostrar todos los procesos que se ejecutan cuando inicia un usuario además de darnos también cual usuario está usando el proceso.

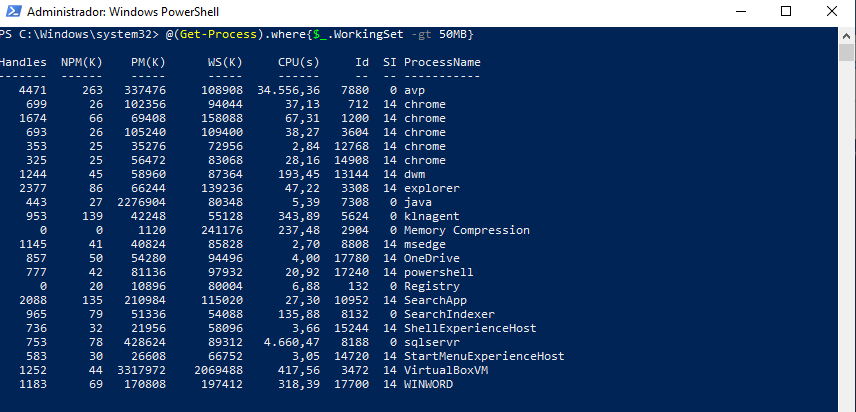
1. Mostrar el tiempo transcurrido en la ejecución de un proceso: Get-Process | Select-Object Id,TotalProcessorTime:





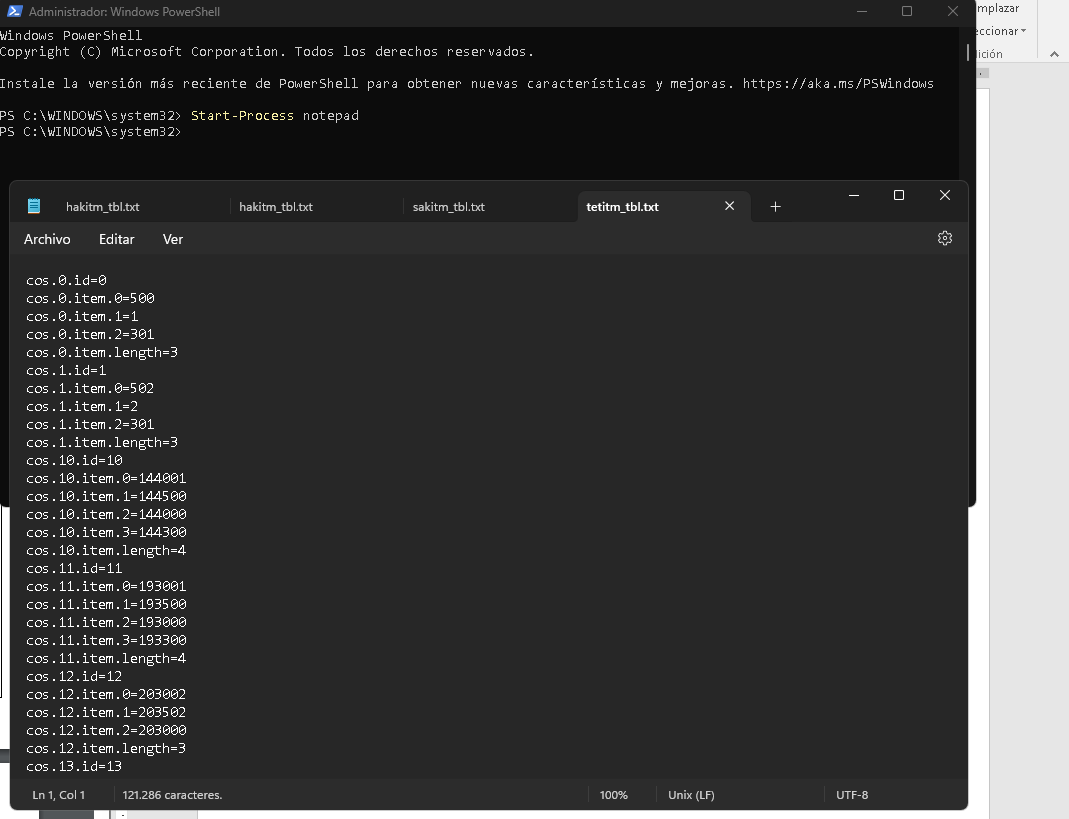
Este comando “Get-Process | select-Object Id,TotalProcessorTime” nos muestra todo el tiempo que se ha estado ejecutando dichoso proceso que se muestra solamente su id, para saber su nombre solo basta añadir después del “select-Object” Name.

1. Mostrar los procesos cuya zona de memoria para trabajar es mayor a 50MB: @(Get-Process).where{$\_.WorkingSet -gt 50MB}:



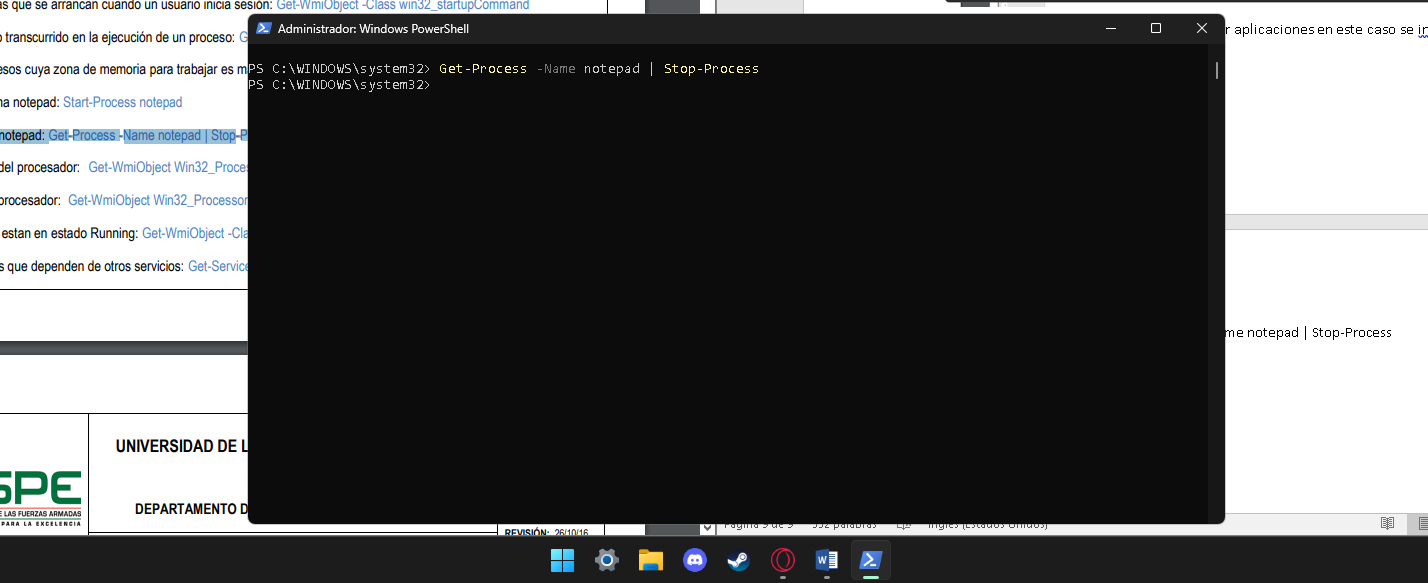
El comando “@(Get-Process).where{$\_.WorkingSet -gt 50MB}” muestra únicamente los procesos que ocupen una cantidad mayor a 50MB memoria útil para ver cuales ocupan mucho recursos de RAM.

1. Iniciar el programa notepad: Start-Process notepad:



El comando “Start-Process” sirve para iniciar aplicaciones en este caso se inicio la aplicación de notepad o blog de notas.

1. Parar el proceso notepad: Get-Process -Name notepad | Stop-Process



El commando “Get-Process –Name notepad | Stop-Process” realiza la function de cerrar procesos en este caso al anteriormente iniciado notepad.

1. Ver información del procesador: Get-WmiObject Win32\_Processor:

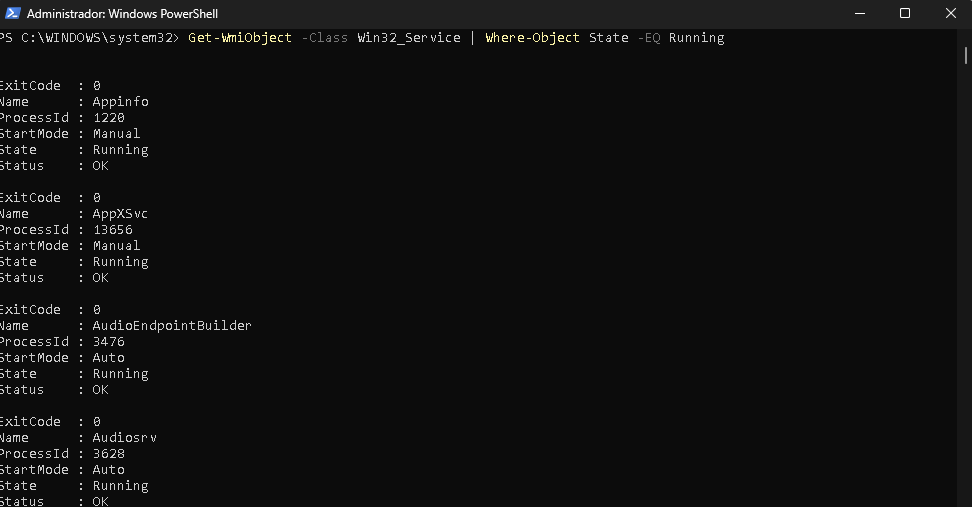
El Comando “Get-WmiObject Win32\_Processor” este nos muestra la información del procesador del equipo.

1. Ver la carga del procesador: Get-WmiObject Win32\_Processor | Select-Object LoadPercentage:

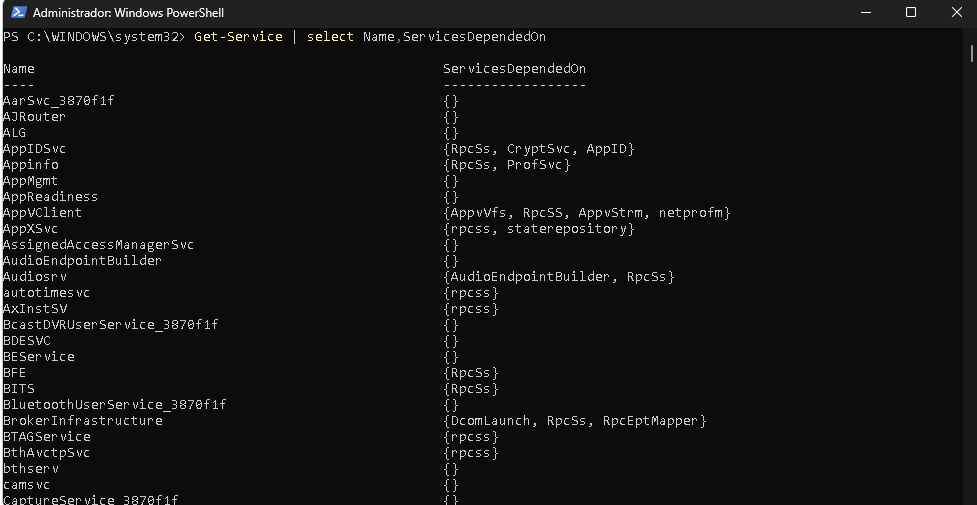


El comando “Get-WmiObject Win32\_Processor | Select-Object LoadPercentage” se encarga de mostrar en consola el porcentaje de carga en el procesador.

1. Ver servicios que estan en estado Running: Get-WmiObject -Class Win32\_Service | Where-Object State -EQ 'Running':

El commando “Get-WmiObject –Class Win32\_Service | Where-Object State –EQ Running” sirve para mostrar en consola todos los servicios que estén corriendo en ese momento.

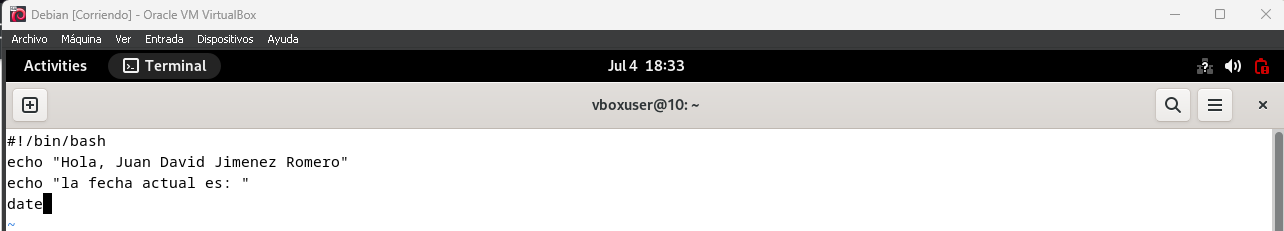
1. Listar los servicios que dependen de otros servicios: Get-Service | select Name,ServicesDependedOn



El comando “Get-Service | select Name,ServicesDependedOn” sirve para mostrar en consola todos los servicios que dependen de otros servicios.

**Ejercicios en Linux (debian):**

**14. Crear un script con nombre fecha.sh, que imprima**:

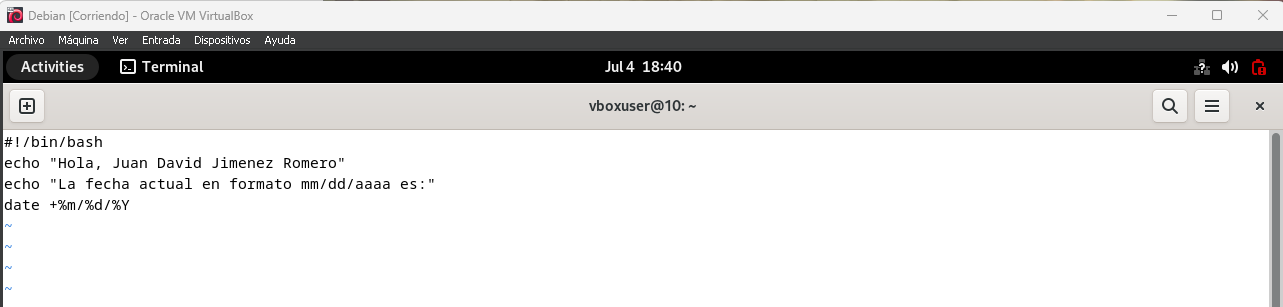


El código dentro del archivo fecha.sh



Y la ejecución del archivo.

**15. Crear un script con nombre fecha1.sh, que imprima:**

Líneas de código dentro del archivo fecha1.sh

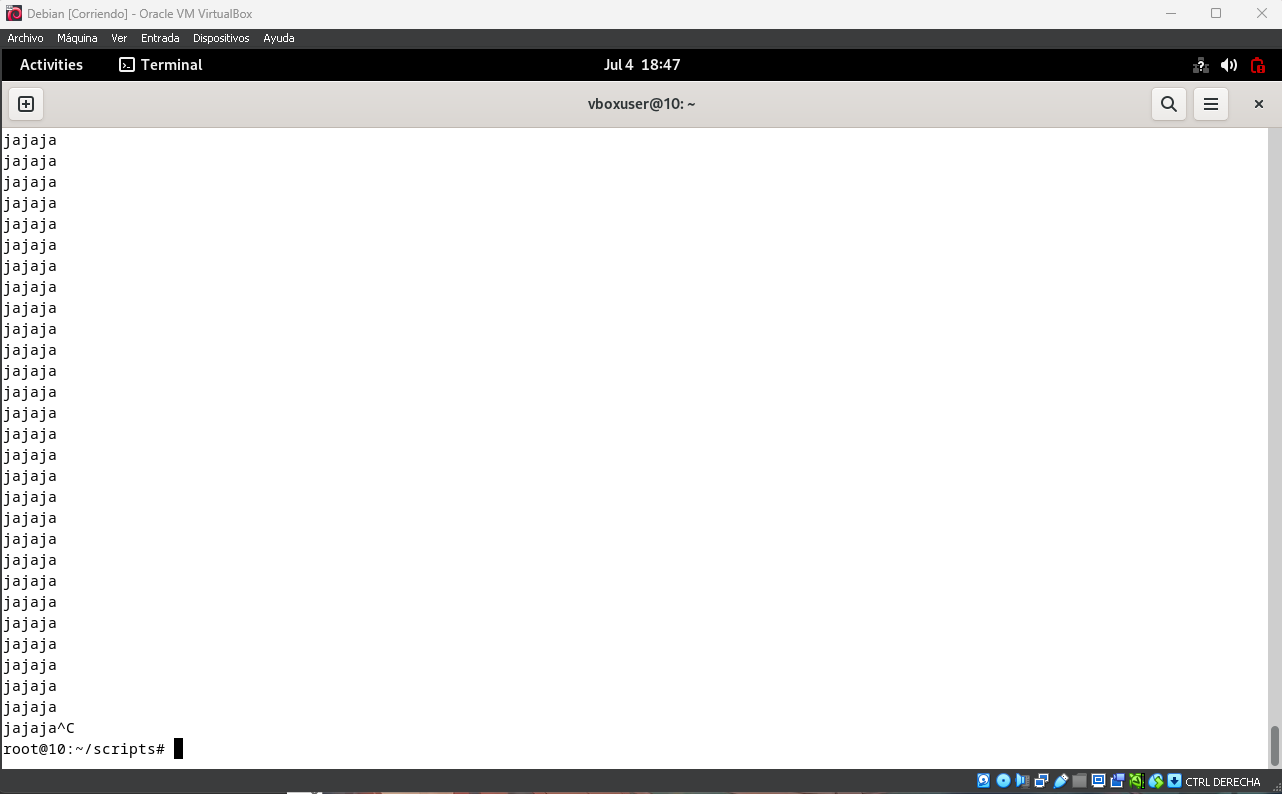


Ejecutando el archivo fecha1.sh

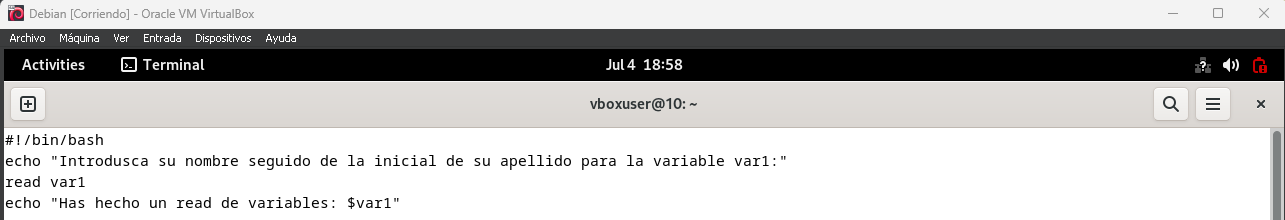
**16. Crear un script con fecha viruschiste.sh que imprima:**



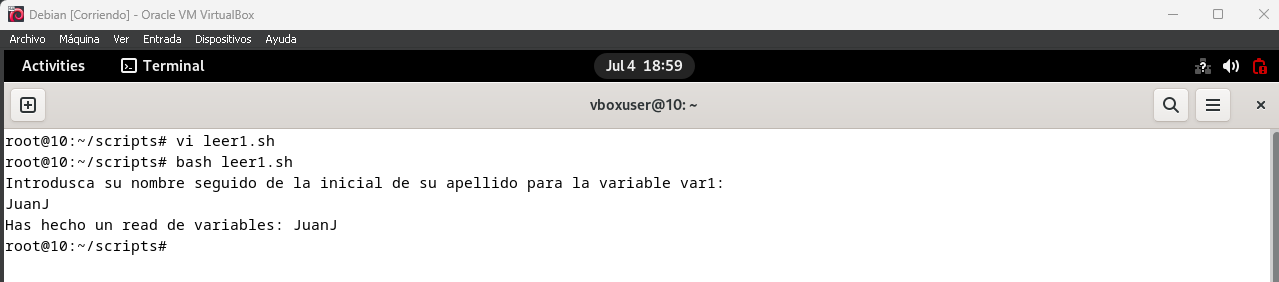
Codigo para realizar la función de escribir en pantalla jajaja infinitamente

Ejecución del archivo “viruschiste.sh”

**17. Muestre el valor que el usuario ingrese por teclado (ingrese su nombre seguido de la inicial de su apellido), nombre al archivo como leer.sh**

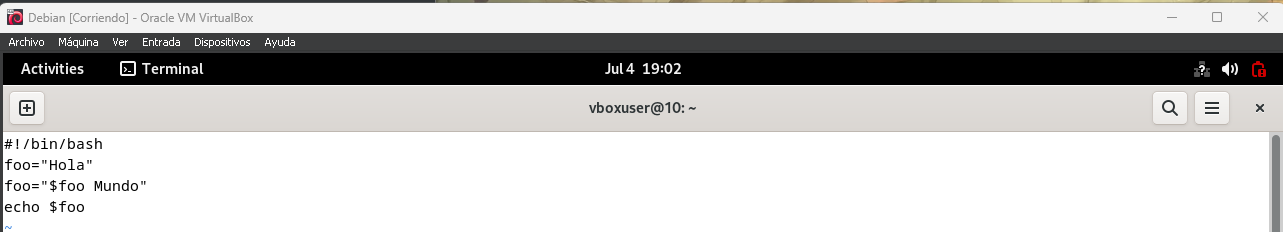


Código para escribir nombre en archivo leer1.sh

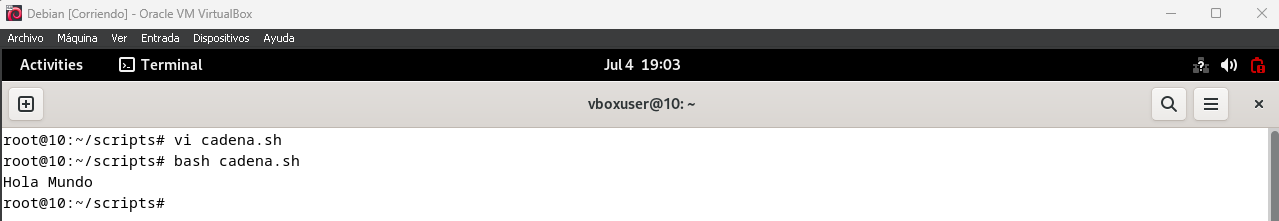


Función realizada por el comando leer1.sh

**18. Unir dos cadenas de texto y mostrar en pantalla**

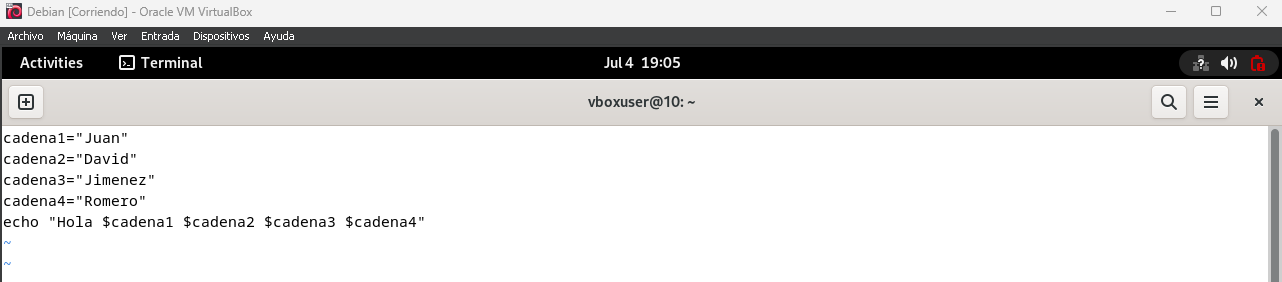


Código para realizar la actividad 18

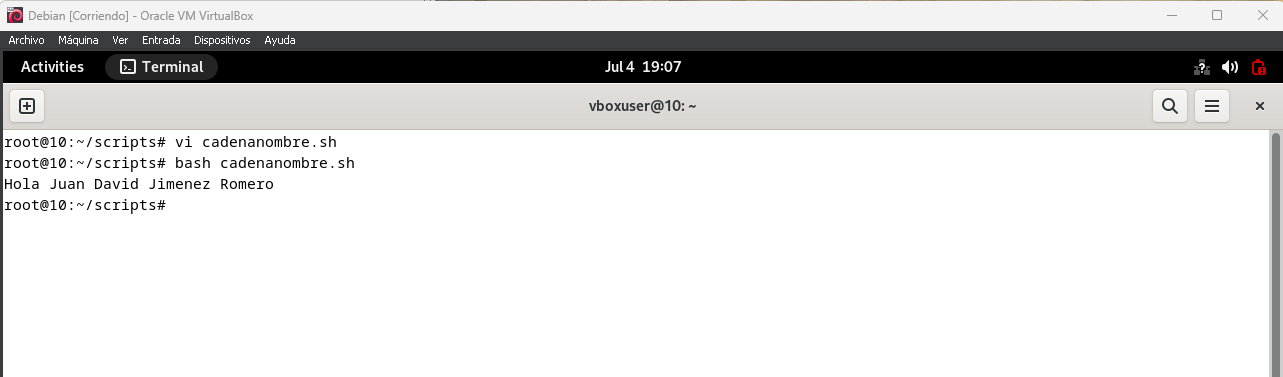


Función al ejecutar el programa cadena.sh

**19. Unir 4 cadenas de texto, cadena 1= su 1er nombre, cadena 2=su 2do nombre, cadena 3= su 1er apellido, cadena 4= su 2do apellido, llamar al archivo como cadenanombre.sh**



Código dentro del archivo cadenanombre.sh



Ejecución del archivo cadenanombre.sh

**20. Operaciones matemáticas: Para sumar 2 números procedemos como sigue, guardar un nuevo script con el nombre suma.sh**



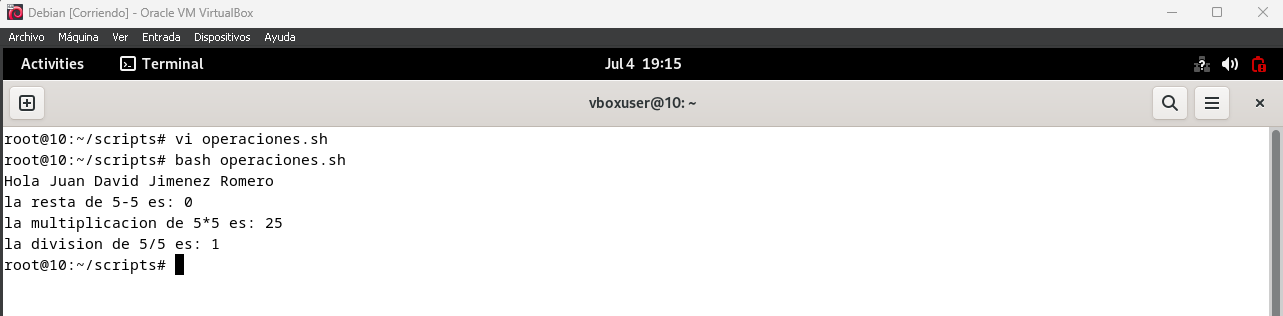
Código dentro del archivo suma.sh

Ejecución del archivo suma.sh

**21. Repita el código del paso 20 pero para resta, multiplicación y división**



Código del archivo operaciones.sh

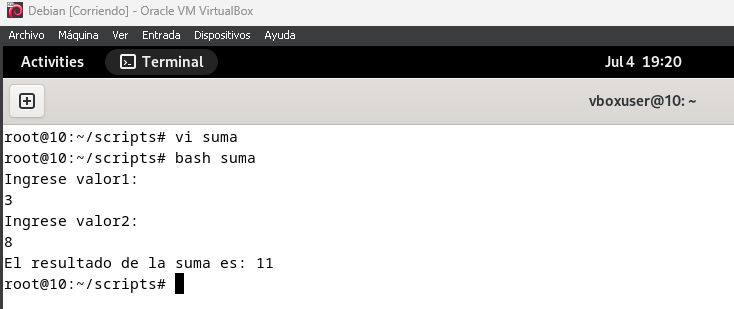


Función del archivo operaciones.sh

**22. Programar un script para hacer las 4 operaciones con valores que ingrese el usuario. Un script para cada operación.**

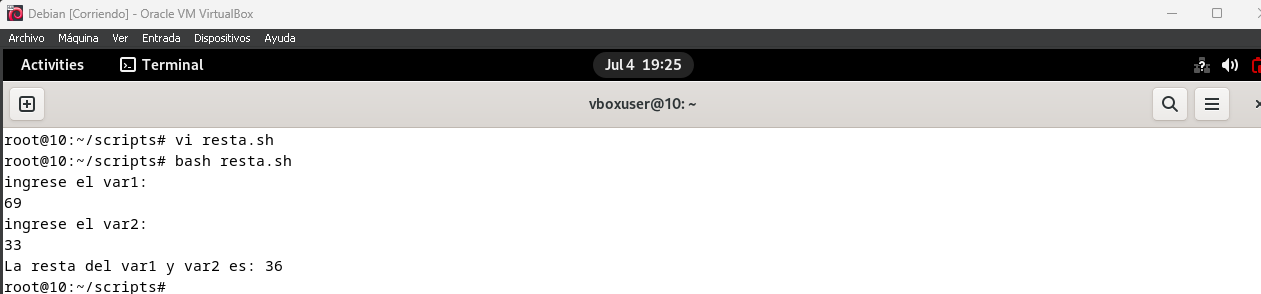
****

Operación suma.sh

Resultado suma.sh



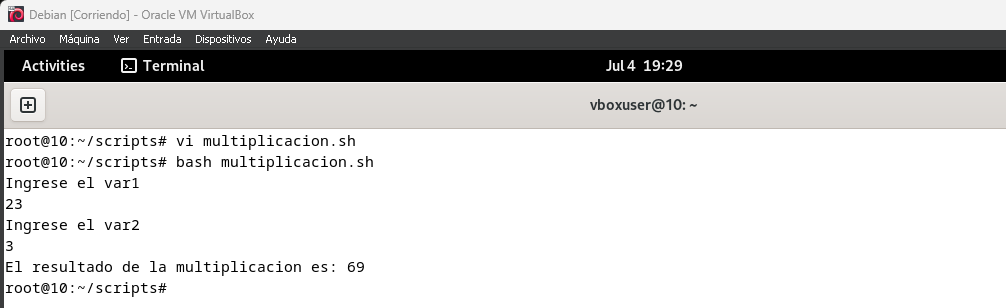
Código de resta.sh



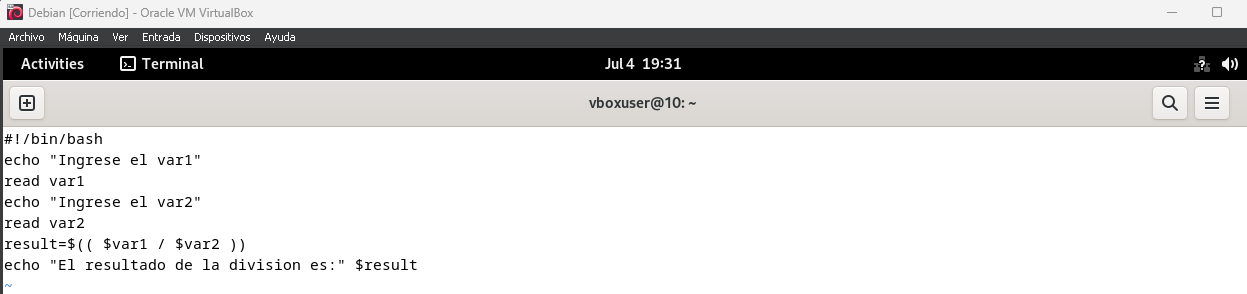
Resultado al momento al ejecutar resta.sh



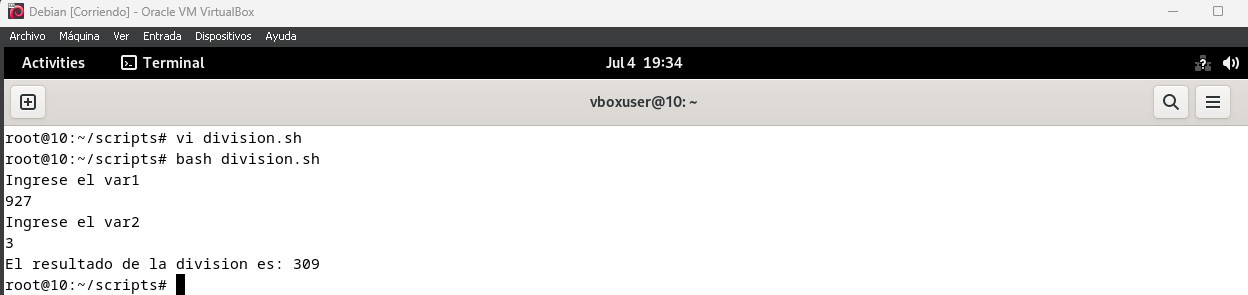
Código de multipliacion.sh



Resultado de la ejecución del archivo multiplicación.sh



El código del programa divison.sh

Resultado de la ejecución del programa división.sh

# Conclusiones

* El conocer los comandos para visualizar los procesos en el PowerShell y en el Terminal nos sirve para saber cuáles programas están realizando un gran uso de recursos del equipo además de permitirnos ver cuál es el origen del proceso y verificar si este es de origen seguro, entre otros.
* El desarrollo de programación mediante “vi” en el terminal de Debian es una forma de ejecutar funciones dentro de la terminal mediante archivos .sh que nos facilitaría mucho al tener que realizar funciones que necesiten varios códigos.
* Se concluye que ambos terminales son ideales para su SO, pero se destaca que debían tiene más accesibilidad en los comandos y en la cantidad de funciones que se pueden hacer en este debido a que se le pueden instalar complementos que ayuden al momento de realizar codificación.

# Recomendaciones

* Se recomienda conocer el significado de los comandos y aprender si existen variaciones más cortas para comandos que podrían ser bastantes largos y difíciles de aprenderse.
* Al momento de realizar codificación con el comando “vi” se recomienda revisar o ver cómo funciona el mismo debido a que este no se controla de manera tradicional, sino que se necesita saber ciertas combinaciones de botones para realizar varias funciones.
* *Se recomienda aprender más sobre programación y comandos en estas aplicaciones* como lo serian el terminal o Windows powershell debido a que son bastantes útiles aprenderse en caso de que se necesite revisar el equipo con posibles fallos.

.

# Bibliografía/ Referencias

# Bibliografía

Desarrollo, P. (2022, febrero 16). *5 tipos de terminales informáticas*. GSC MADRID GRUPO DE SOFTWARE Y CONSULTING. <https://gscmadrid.com/tipos-de-terminales-informaticas/>

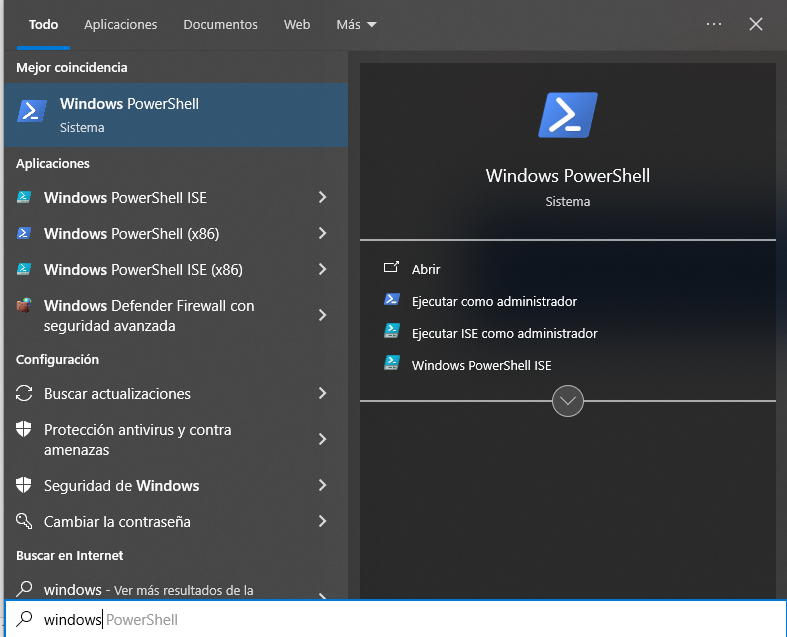
Euroinnova Business School. (2021, julio 28). *Conoce cuáles son las carreras relacionadas con computación y sus salidas laborales*. Euroinnova Business School. <https://www.euroinnova.ec/sistema-operativo>

*Informática Básica: Qué es un sistema operativo*. (s/f). Gcfglobal.org. Recuperado el 25 de mayo de 2024, de <https://edu.gcfglobal.org/es/informatica-basica/que-es-un-sistema-operativo/1/>

*Tipos de sistemas operativos y sus características*. (2023, junio 14). UCMA. <https://www.universitatcarlemany.com/actualidad/blog/tipos-de-sistemas-operativos/>

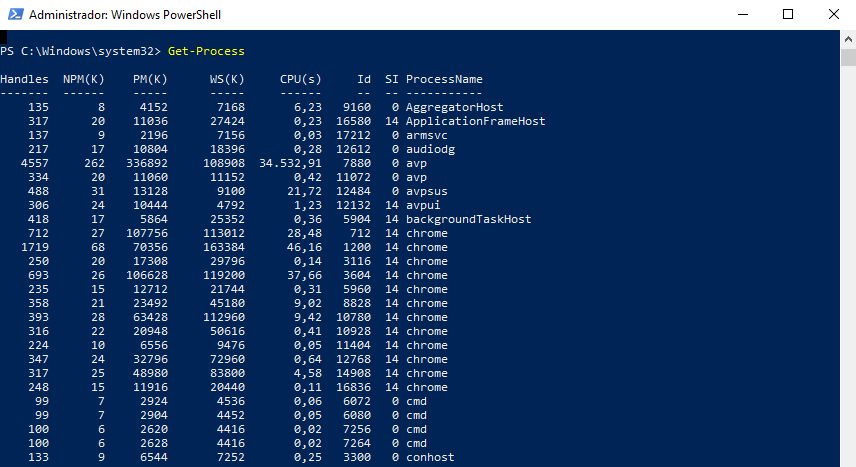
# Anexos

Iniciamos Windows powershell mediante el buscador de Windows:

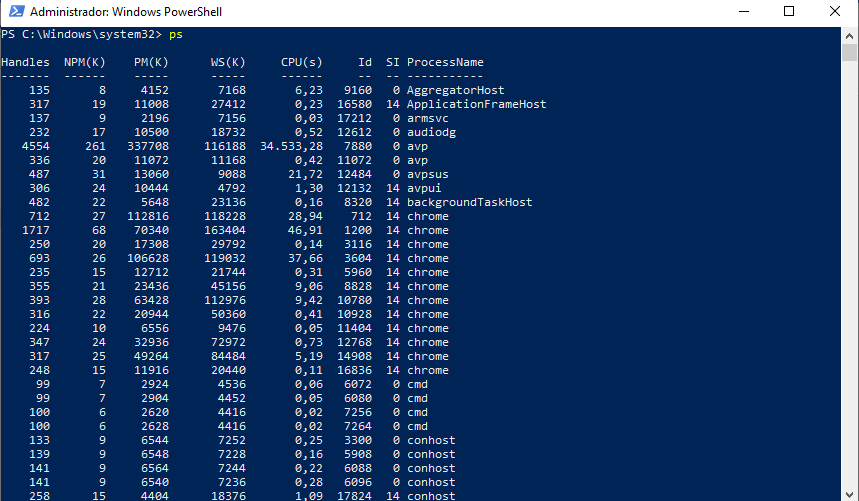


Se nos mostrara la siguiente ventana que nos dice que ya se inició el Windows PowerShell:

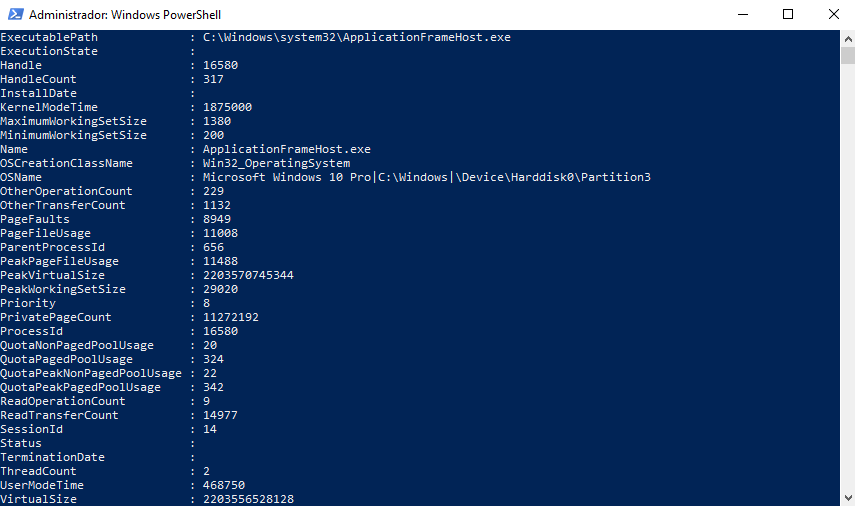




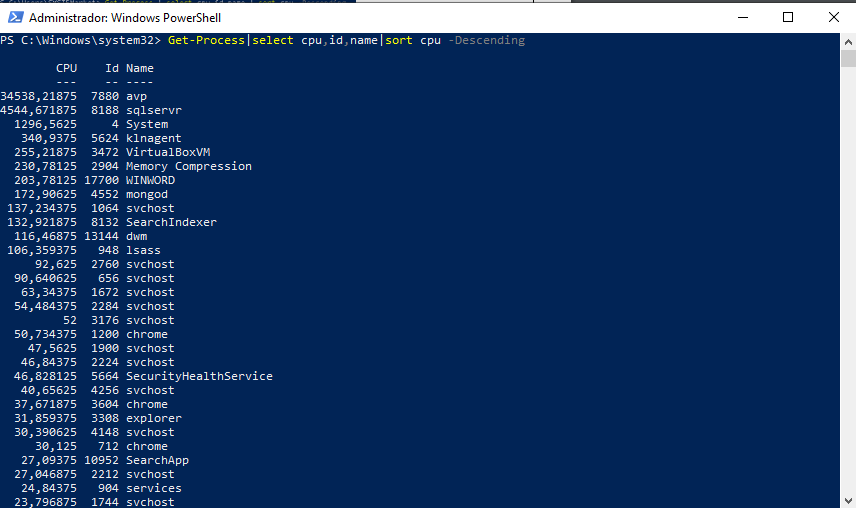
El comando “Get-Process” sirve para mostrar todos los procesos que se están ejecutando ya sea en primer o segundo plano con su id, consumo de CPU y otros datos importantes.



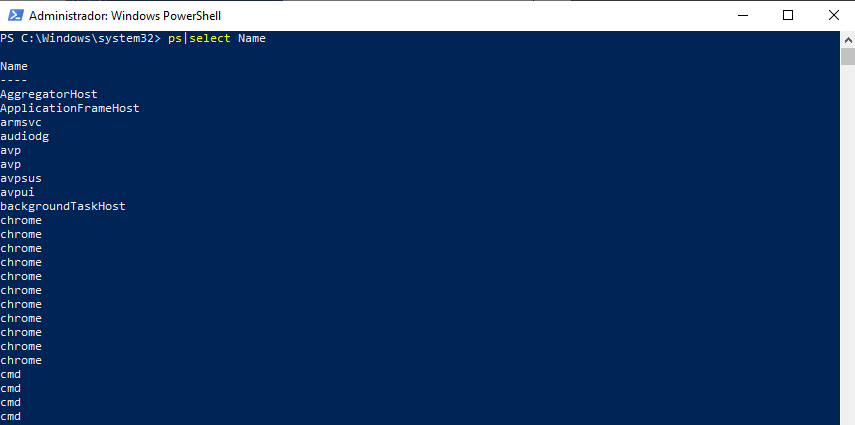
Con el comando “ps” realiza exactamente lo mismo que el comando “Get-Process” solo que lo resume a 2 letras siendo P la letra inicial de Process y la S de la letra final.



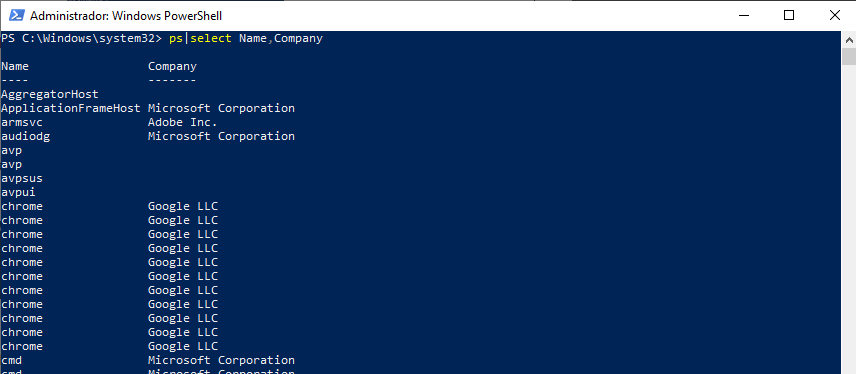
El comando “Get-WmiObject –Class win32\_process” lo que realiza este comando es mostrar todos los procesos que se encuentran en la dirección de Win32 y este nos dice gran parte de los datos y detalles que lo componen como el uso de RAM, la prioridad, entre otros.



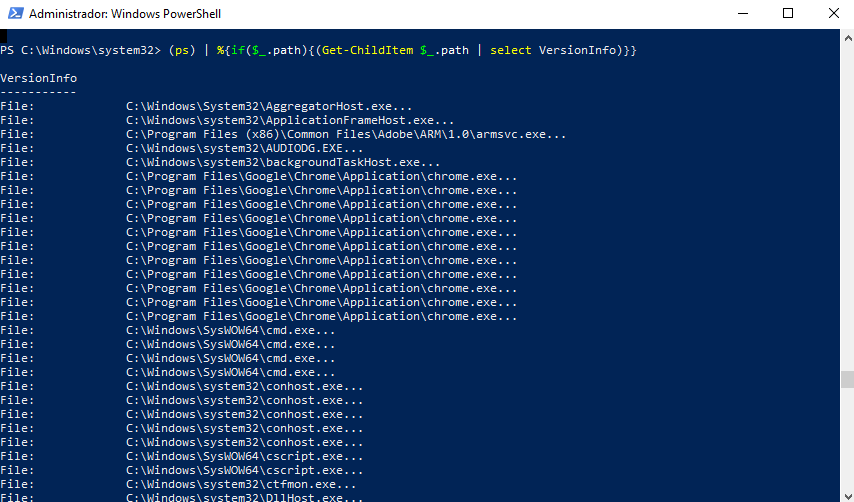
Con el comando “Get-Process|select cpu,id,name|sort cpu –Descending” sirve para mostrar los procesos que tienen un alto consumo de CPU este mostrado de forma descendente con y mostrado con la forma de CPU, id, Name.



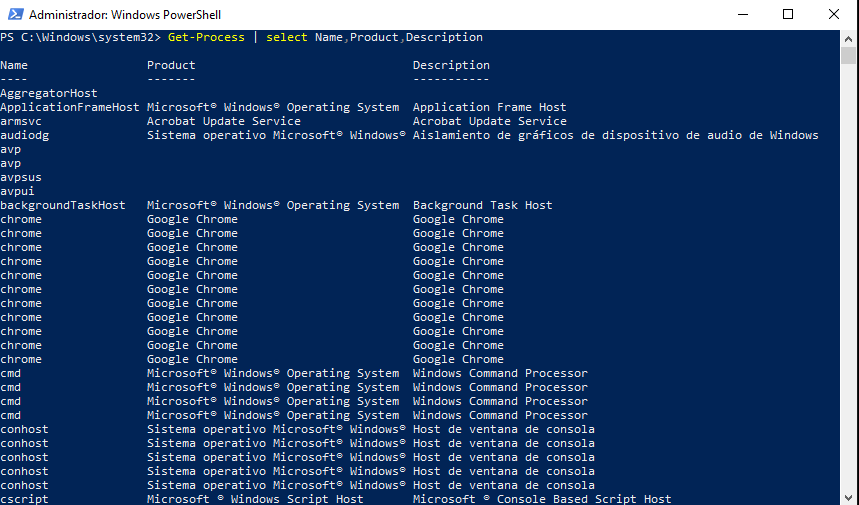
El comando “ps|select Name” estas nomas muestra el nombre de la lista de procesos en orden de abecedario.



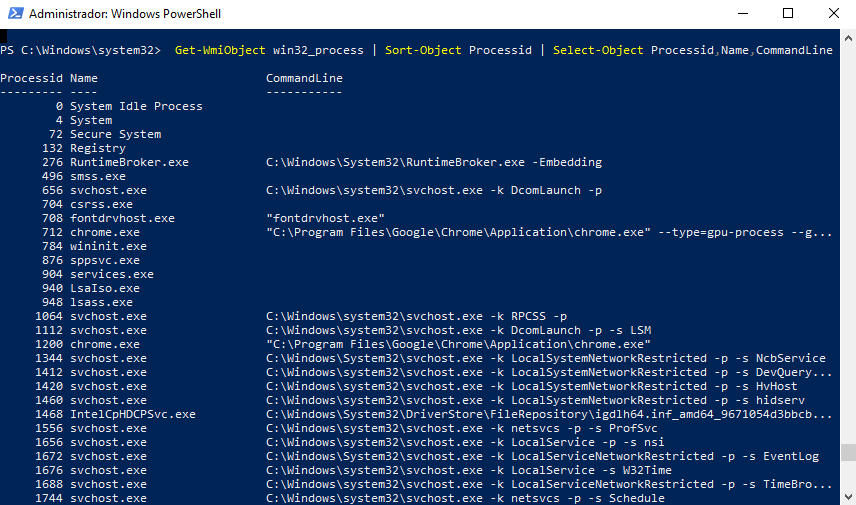
Con el comando “ps|select Name, Company” este realiza la función de mostrar en consola el nombre del proceso y la Compañía que tiene registrado a su nombre.



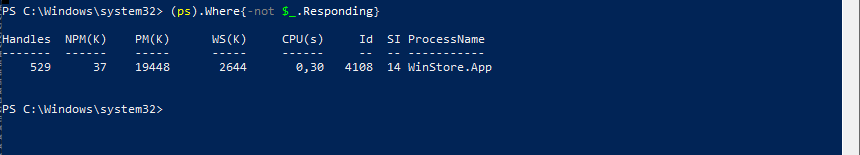
El comando “(ps) | %{if($\_.path){(Get-ChildItem $\_.path | select VersionInfo)}}” nos sirve para mostrarnos la dirección de los procesos donde se encuentra y además nos muestra la información del mismo.



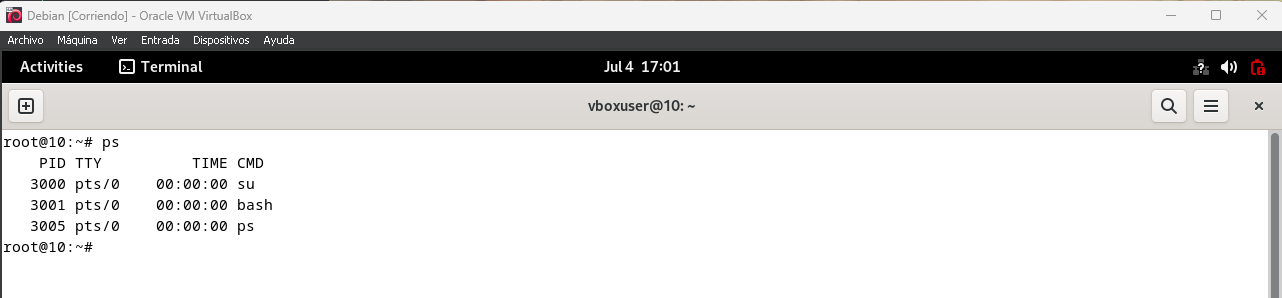
Con el comando “Get-Process | select Name,Product,Description” este comando sirve para listar los procesos solamente los Nombres, el producto y una descripción de los que tienen alguna.



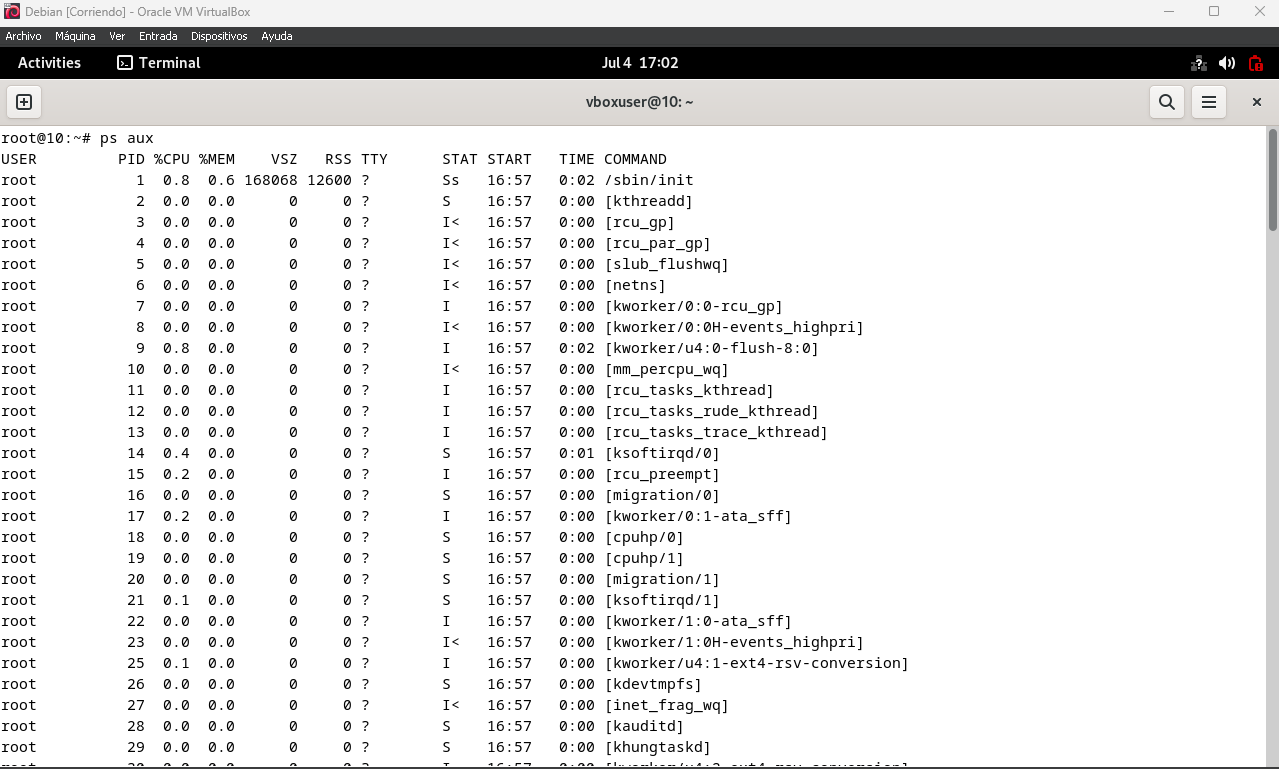
Con el comando “: Get-WmiObject win32\_process | Sort-Object Processid | Select-Object Processid,Name,CommandLine” nos muestra el id del proceso, el nombre, y la ubicación de este de los procesos que se ejecutan al iniciar el equipo o el SO.

El comando “(ps).Where{-not $\_.Responding}” este muestra los procesos que no están respondiendo este caso muestra únicamente 1 que es WinStore.App que sería la tienda de Windows.

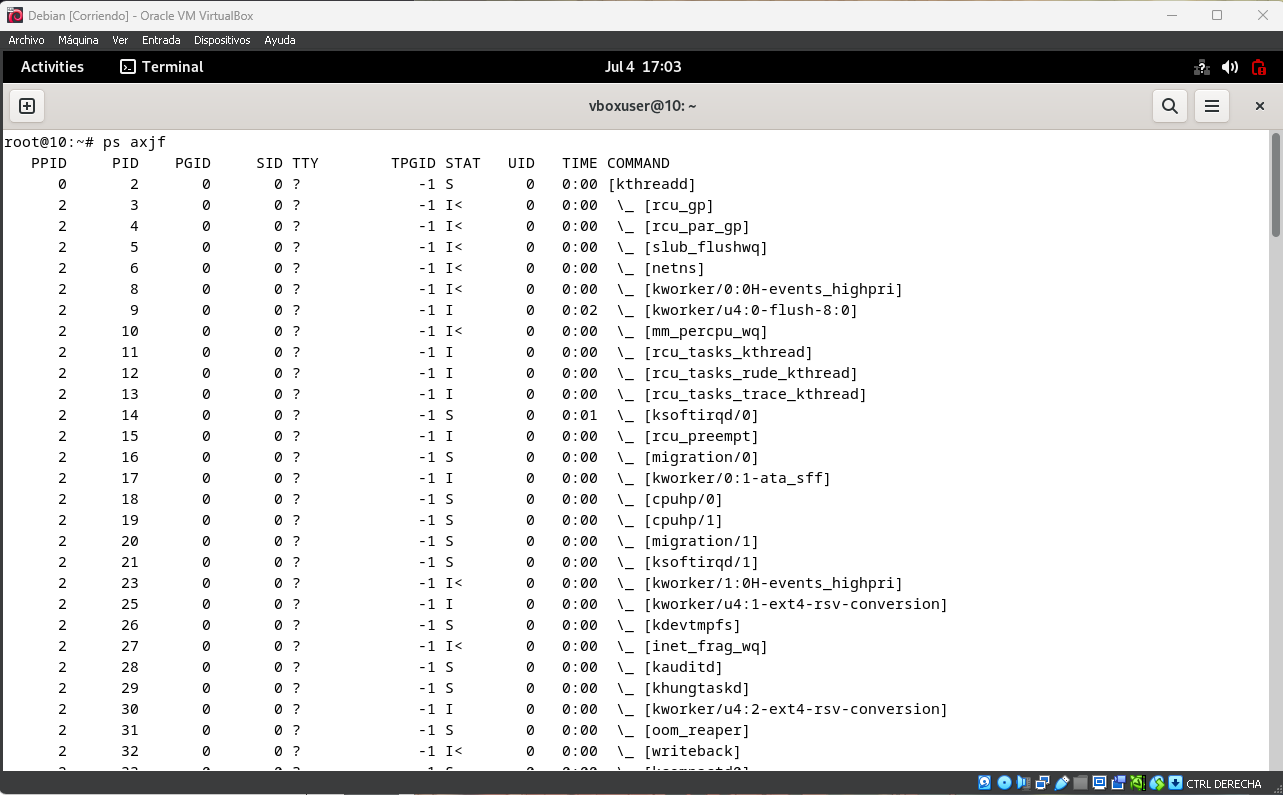
**Actividad en Linux (debían):**



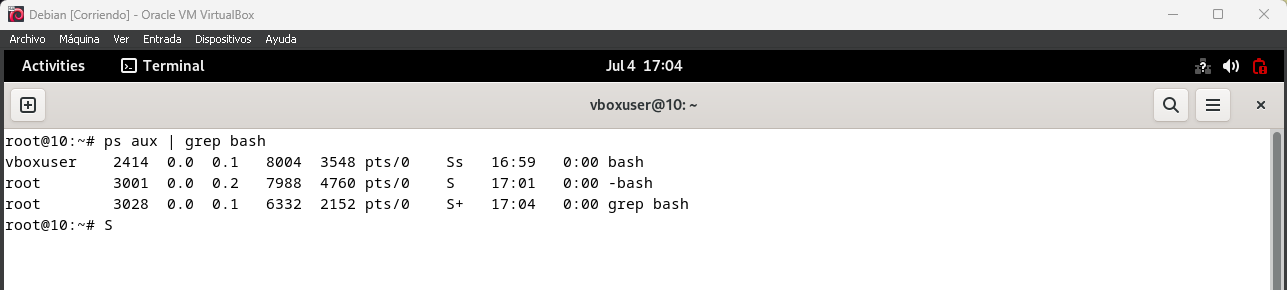
El comando “ps” nos muestra la cantidad de procesos que hay ejecutándose y el tiempo que este ha estado activo.

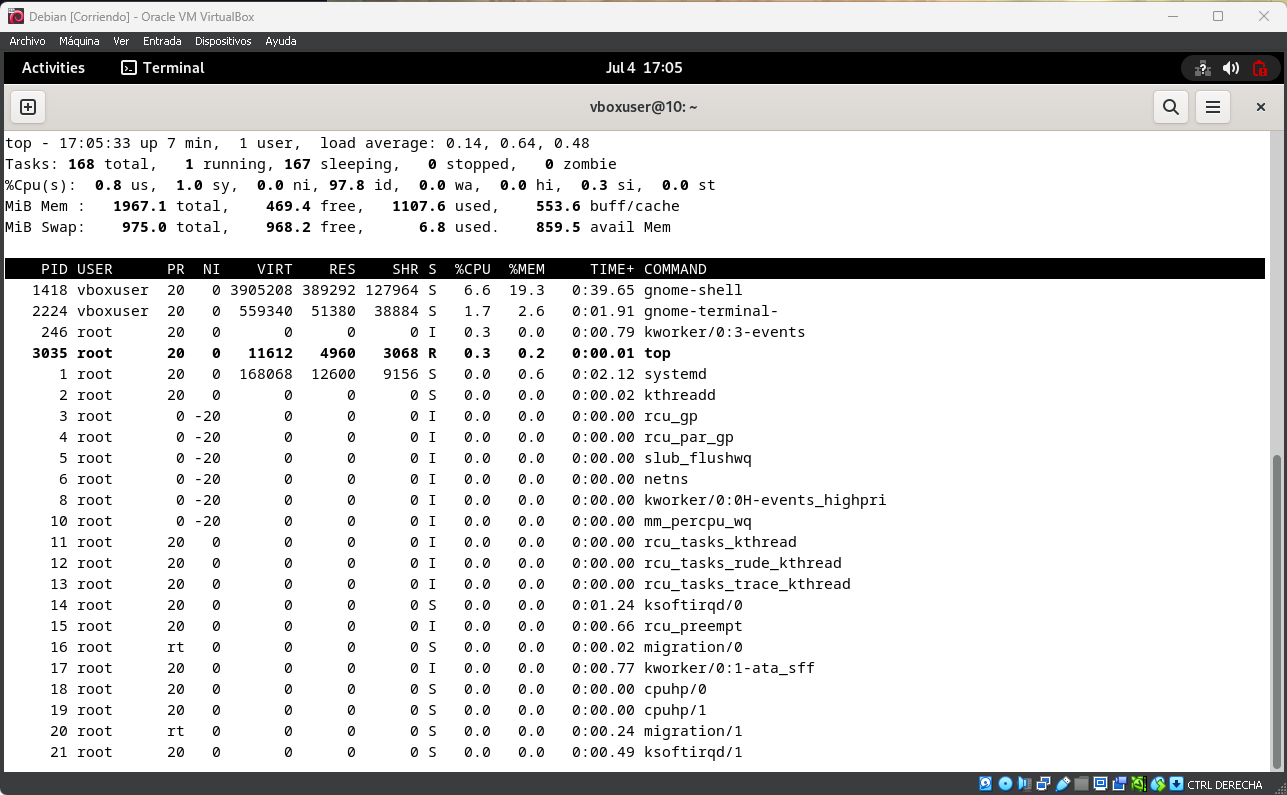


El comando “ps –aux” nos muestra todos los procesos que existen en el sistema en este caso se ven que hay más de los que muestra normalmente.



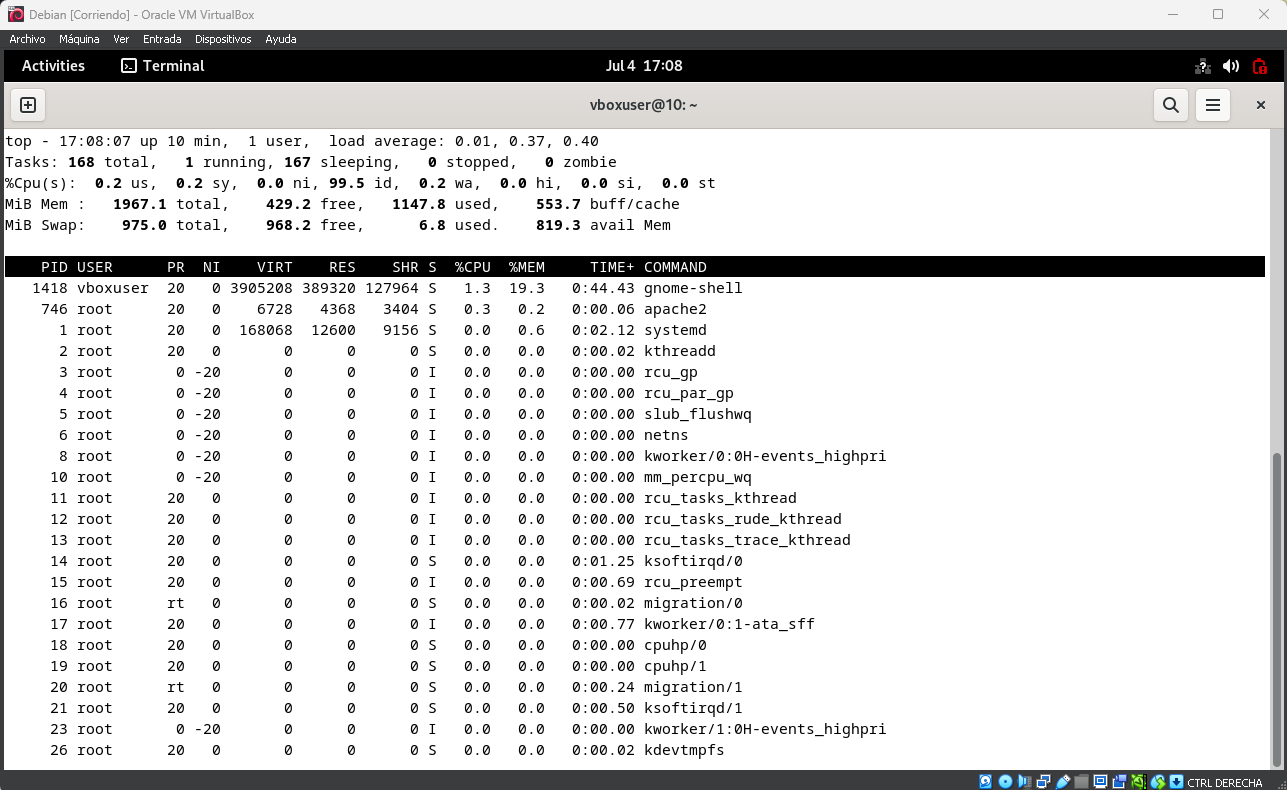
Con el comando “ps axjf” nos muestra un árbol jerarquico con la ruta donde se encuentra cada uno de los procesos.

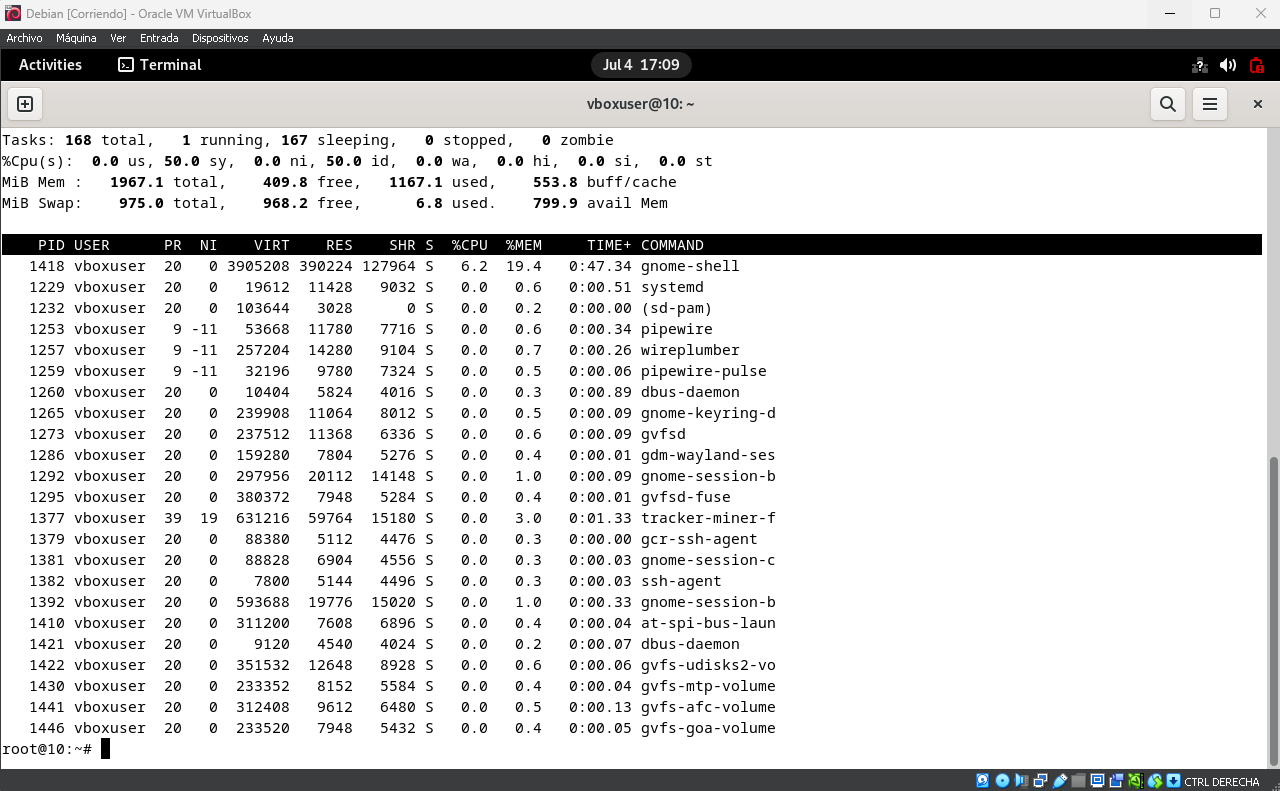
El comando “ps aux | grep bash” nos muestra los procesos que son exclusivamente de bash.

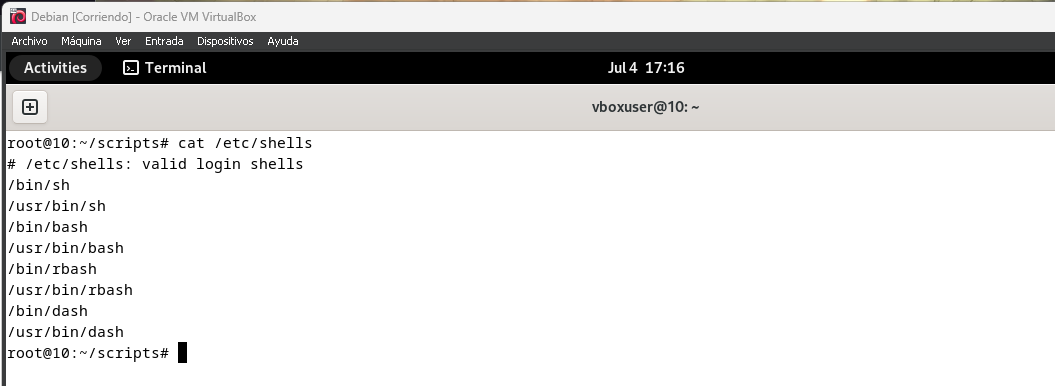


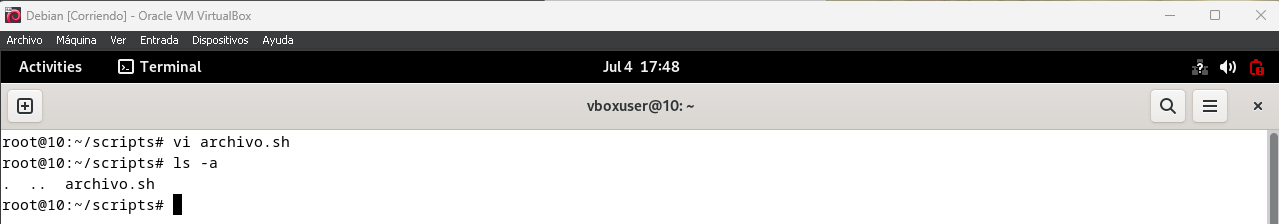
Con el comando “top” visualizamos el panel de tareas por asi decirlo de Linux en el terminal donde nos muestran los procesos ejecutándose y cuanto ocupan cada uno en memoria, entre otros.

Con el comando “top –d 5” solamente se mostrará esa interfaz durante 5 segundos eso dependiendo del número que se coloque después de la d.

Con el comando “top –o %CPU” mostrara de forma ordenada dependiendo del consumo de la CPU en forma descendente.

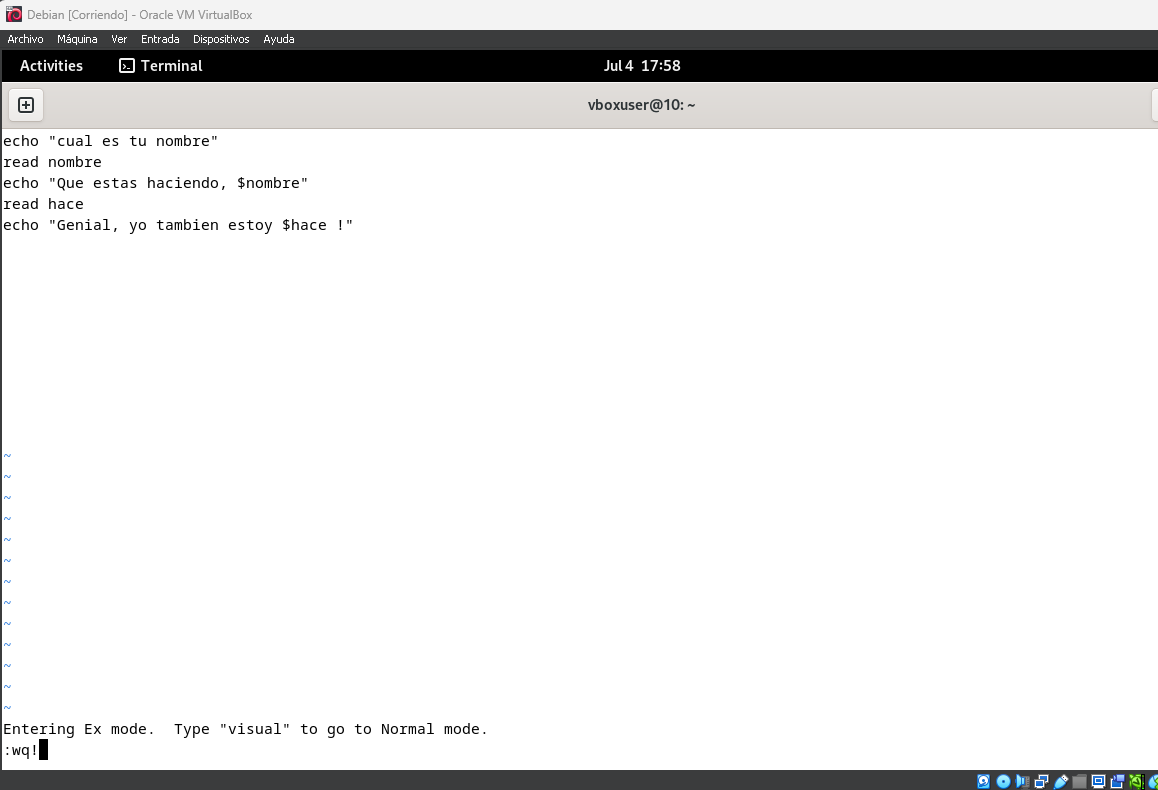
Con el comando “top –u (NombreDelUsuario)” mostrara los procesos ejecutados únicamente por ese usuario.

Revisamos los shells de bash con el comando “cat /etc/shells” generalmente vienen ya incluidos en la mayoría de distribuciones de Linux

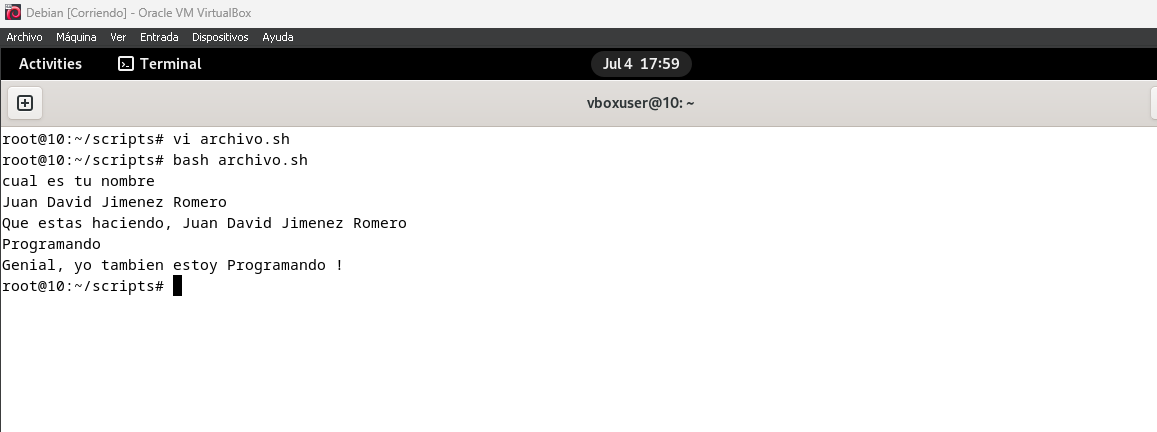


Con el comando vi “archivo.sh” dentro de la carpeta creada con el nombre de scripts con el comando “mkdir”, entramos al Shell de Linux donde ponemos dentro el comando “#!/bin/sh” que es como la librería que usa y abajo un comando en este caso coloque el de “top” y luego pulsando el botón “esc + shift + (alguna letra)” colocamos la siguiente línea “wq!” que es para guardar los archivos en “vi” y con el comando “ls -a” visualizamos que esté hay guardado.

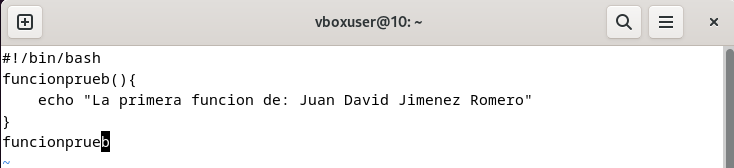
y al momento de ejecutar el archivo.sh con el comando “bash (nombre archivo)” este ejecuta el comando top puesto dentro del archivo.sh

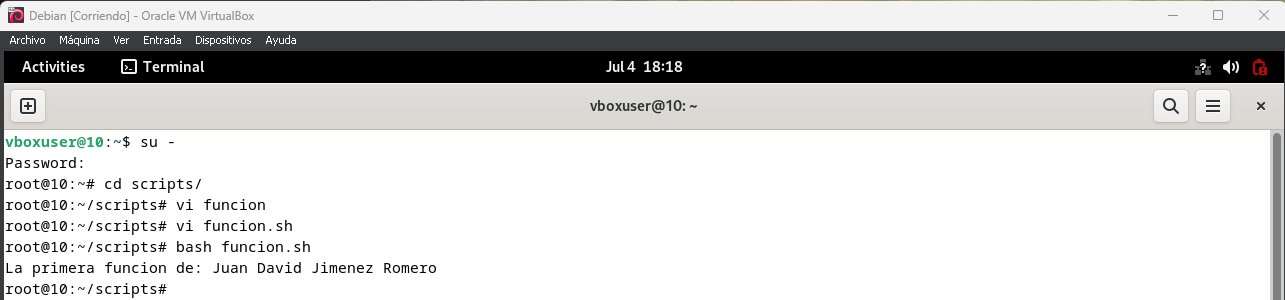


Ingresamos el ejemplo de practica dentro del archivo.sh y lo guardamos

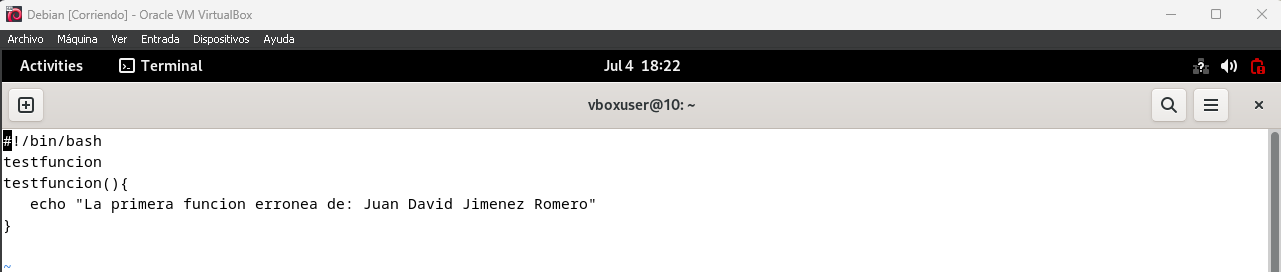


Resultado al ejecutar el archivo.sh con el comando bash

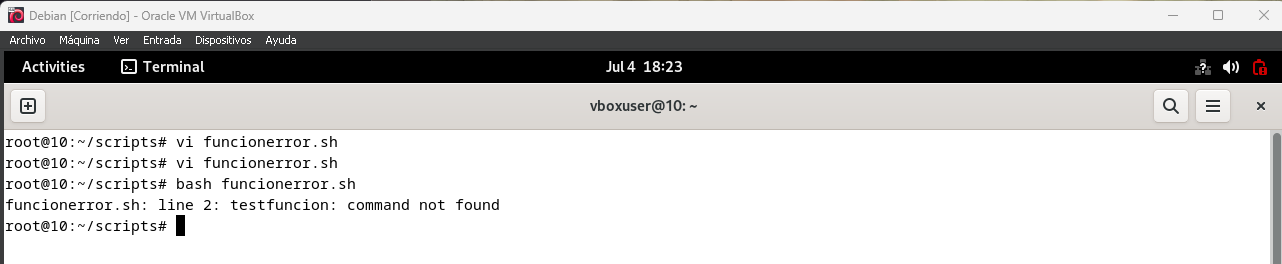




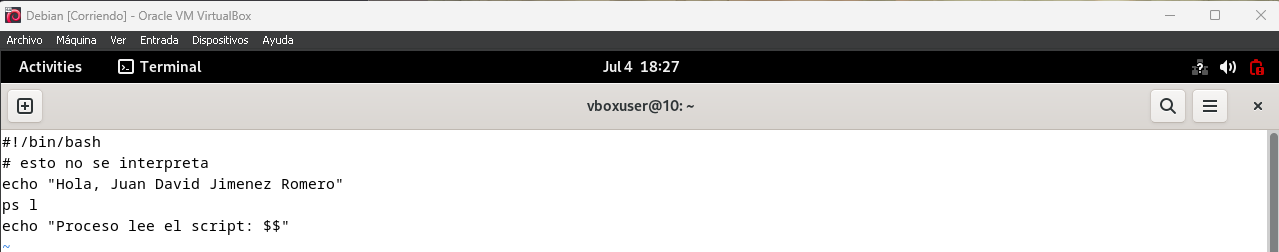
Creamos una función dentro de otro archivo .sh nombrado como función.sh y dentro realizamos una función que muestre en pantalla un mensaje



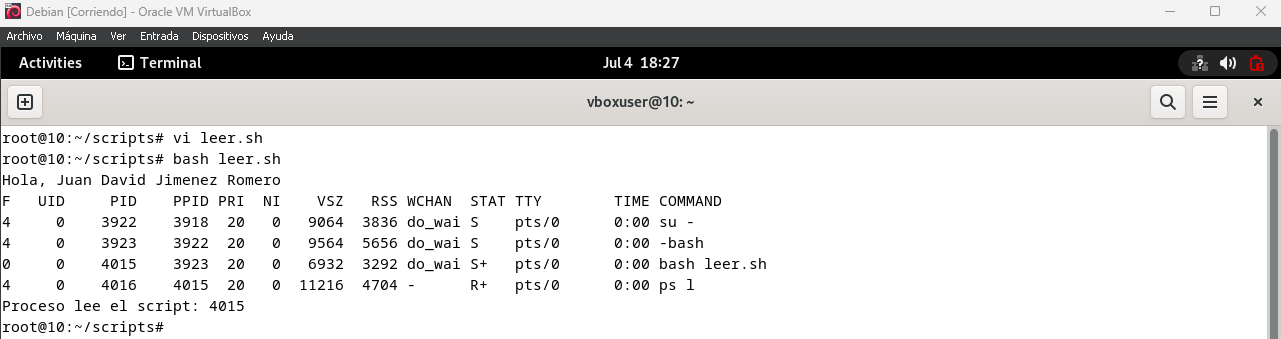
Creamos otro archivo .sh llamado funcionerror.sh y le realizamos lo mismo que el anterior solo que la llamada a la función esta primero que la función para ver que sale error.



Nos sale un error y nos dice en que línea se encuentra el error en este caso la 2 debido a que está haciendo la llamada a la función inexistente.



Creamos un archivo .sh con el nombre de leer.sh y lo llenamos con el siguiente código



Usando el comando “bash” con el archivo leer.sh y lo ejecutamos nos muestra lo de la imagen.