

**Universidad Escuela Colombiana de Ingeniería Julio Garavito Arquitectura
Computacional y Sistemas Operativos
Shell Programming**

Objetivos

1. Familiarizar al estudiante con el ambiente ShellCheck (Unix Shell)
 2. Aprender a escribir programas básicos en Unix Shell
 3. Aprender a escribir programas básicos en Windows PowerShell
-

1. Escriba un programa Shell que:

Limpie la pantalla

Imprima el mensaje "Hello World from Shell".

Conéctese al ambiente ShellCheck (<https://www.shellcheck.net/>) y revise el programa shell.

Modifique el script para PowerShell usando comandos de Windows y en Unix para Solaris.

BASIC SHELL SCRIPT (BASH)

Primero, se guarda el archivo hello.sh y dentro de este, se agrega los comandos necesarios para cumplir con este objetivo, queda así:

```
#!/bin/bash
```

```
# Simple Hello World program in Bash
```

```
# Clear the screen
```

```
clear
```

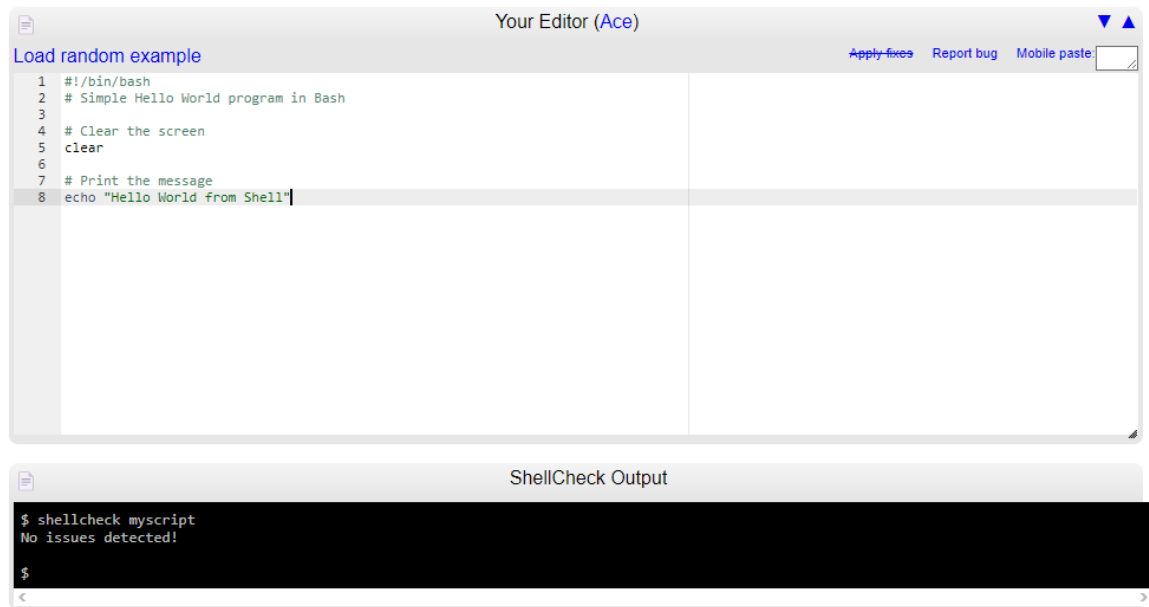
```
# Print the message
```

```
echo "Hello World from Shell"
```

Antes de hacer ejecutable este archivo y probarlo, hay que copy-page el contenido del archivo y validarlo con Shellcheck.

Shell Programming - Nombre: Andersson David Sánchez Méndez ACSO-2 – 1L

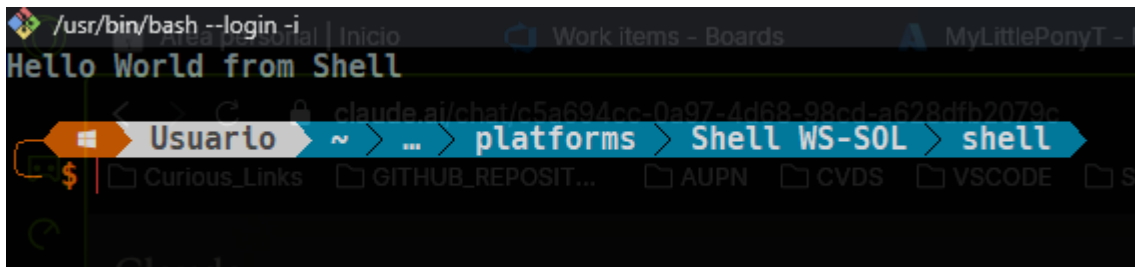
Paste a script to try it out:



Ahora sí, si en Bash se hace ejecutable el archivo con el comando: **chmod +x hello.sh**

Y se ejecuta con: **./hello.sh**

Así, genera lo solicitado:



WINDOWS POWERSHELL VERSION (WINDOWS CORE)

Primero, se guarda el archivo `hello.ps1` y dentro de este, se agrega los comandos necesarios para cumplir con este objetivo, queda así:

```
# Simple Hello World program in PowerShell
```

```
# Clear the screen
```

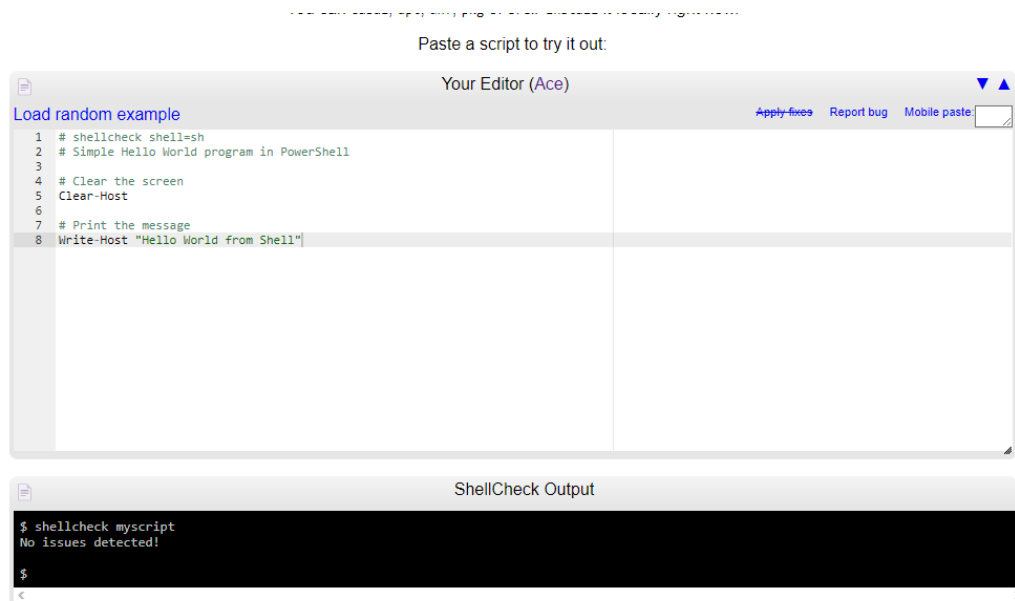
```
Clear-Host
```

```
# Print the message
```

```
Write-Host "Hello World from Shell"
```

Shell Programming - Nombre: Andersson David Sánchez Méndez ACSO-2-1L

Antes de hacer ejecutable este archivo y probarlo, hay que copy-page el contenido del archivo y validarlo con Shellcheck. Aunque en este caso fue necesario añadir la primera línea porque Shellcheck está hecho principalmente para Unix, entonces hay que especificar que este caso es de pwsh.

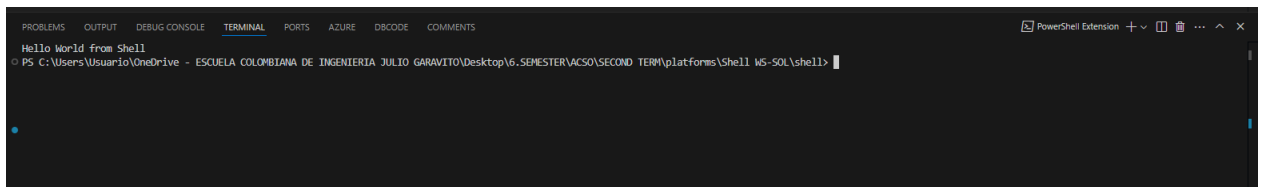


ShellCheck is

Ahora sí, si en powershell se ejecuta con: **.\hello.ps1**

Si ocurren errores de política de ejecución, se pone el comando: **Set-ExecutionPolicy - ExecutionPolicy Bypass -Scope Process**

Así, genera lo solicitado:



SOLARIS UNIX VERSION

Todo es lo mismo que BASIC SHELL SCRIPT (BASH), lo único que cambia es el comando en la primera línea.

```
#!/bin/sh
```

```
# Simple Hello World program for Solaris
```

```
# Clear the screen
```

```
clear
```

Shell Programming - Nombre: Andersson David Sánchez Méndez ACO-2-1L

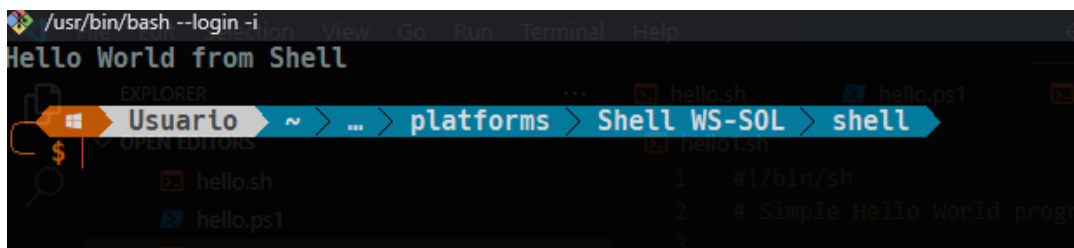
Print the message

echo "Hello World from Shell"

Antes de hacer ejecutable este archivo y probarlo, hay que copy-paste el contenido del archivo y validarlo con Shellcheck.



Se hace ejecutable con `chmod +x hello1.sh` y se ejecuta de la misma manera `./hello1.sh`, generando el output correcto.



2. Ejecución automática de una secuencia de comandos
Escriba un programa Shell (y revíselo en ShellCheck) que:

Limpie la pantalla

Imprima el mensaje “El número de líneas del archivo /etc/profile es:” y el número de líneas encontrados en ese archivo.

Modifique el script para PowerShell revisando el archivo C:\Windows\System32\drivers\etc\hosts y en Unix para Solaris.

WINDOWS POWERSHELL (PWSH) SCRIPT

Primero, se guarda el archivo count_lines.ps1 y dentro de este, se agrega los comandos necesarios para cumplir con este objetivo, queda así:

```
# Script to count lines in the hosts file
```

```
# Clear the screen
```

```
Clear-Host
```

```
# Get file path for Windows hosts file
```

```
$filePath = "C:\Windows\System32\drivers\etc\hosts"
```

```
# Count the lines and print the result
```

```
$lineCount = (Get-Content -Path $filePath).Count
```

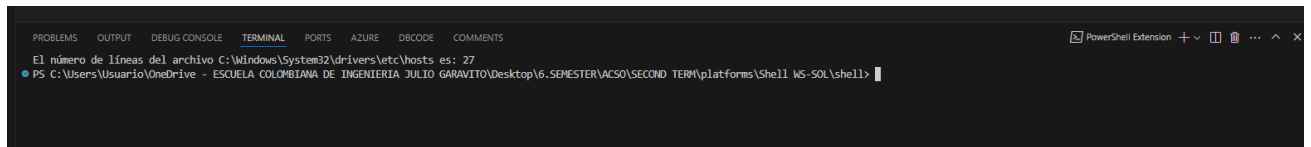
```
Write-Host "El número de líneas del archivo $filePath es: $lineCount"
```

Ahora se valida con Shellcheck:

Shell Programming - Nombre: Andersson David Sánchez Méndez ACSO-2-1L



Se ejecuta con: `.\count_lines.ps1` y se ve el output correcto.



SOLARIS (UNIX) SCRIPT

Primero, se guarda el archivo `count_lines.sh` y dentro de este, se agrega los comandos necesarios para cumplir con este objetivo, queda así:

```
#!/bin/sh
```

```
# Script to count lines in /etc/profile
```

```
# Clear the screen
```

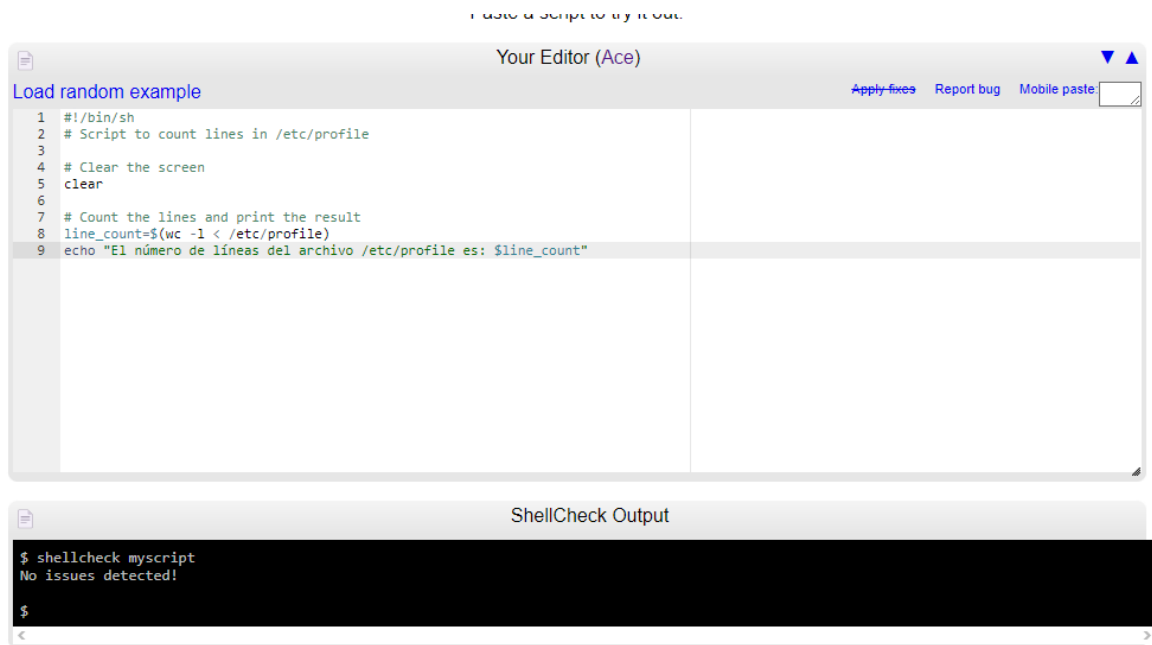
```
clear
```

```
# Count the lines and print the result
```

```
line_count=$(wc -l < /etc/profile)
```

```
echo "El número de líneas del archivo /etc/profile es: $line_count"
```

Ahora se valida con Shellcheck:

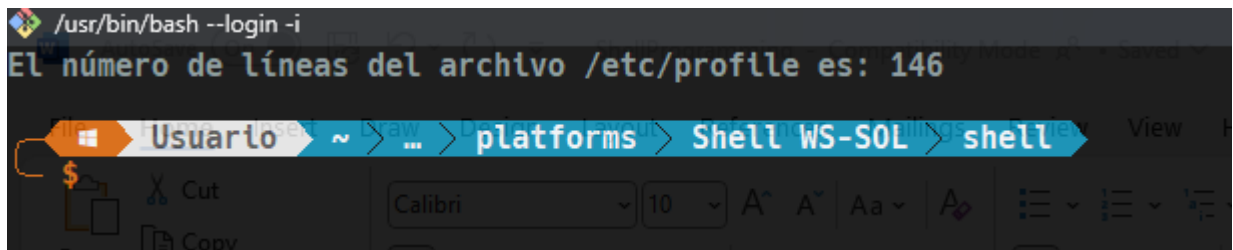


The image shows a code editor window titled "Your Editor (Ace)" with a file named "Load random example". The code is a shell script:

```
1 #!/bin/sh
2 # Script to count lines in /etc/profile
3
4 # Clear the screen
5 clear
6
7 # Count the lines and print the result
8 line_count=$(wc -l < /etc/profile)
9 echo "El número de líneas del archivo /etc/profile es: $line_count"
```

Below the editor is a "ShellCheck Output" window showing the command `$ shellcheck myscript` and the output `No issues detected!`.

Se hace ejecutable con `chmod +x count_lines.sh` y se ejecuta de la misma manera `./count_lines.sh`, generando el output correcto.



The image shows a terminal window with the prompt `/usr/bin/bash --login -i`. The output of the script is displayed: `El número de líneas del archivo /etc/profile es: 146`. The terminal window has a title bar with "Usuario" and a path `platforms > Shell WS-SOL > shell`.

3. Manejo de parámetros

Escriba un programa Shell (y revíselo en ShellCheck) que:

- Limpie la pantalla

- Busque una palabra dada por el usuario en un archivo especificado por el mismo.

La ejecución sería del estilo `buscar palabra.sh palabra buscada archivo de búsqueda`, ejemplo `buscar-palabra.sh casa /etc/passwd`

- Imprima el resultado de la búsqueda.

Modifique el script para PowerShell usando comandos de Windows. Ejemplo `buscar palabra.ps1 casa C:\Windows\inf\xusb22.inf` y en Unix para Solaris.

POWERSHELL SCRIPT

Primero, se guarda el archivo `buscar_palabra.ps1` y dentro de este, se agrega los comandos necesarios para cumplir con este objetivo, queda así:

```
# Script to search for a word in a specified file
```

```
# Clear the screen
```

Shell Programming - Nombre: Andersson David Sánchez Méndez ACSO-2 – 1L

Clear-Host

Check if correct number of parameters were provided

if (\$args.Count -ne 2) {

Write-Host "Error: Se requieren dos parámetros."

Write-Host "Uso: .\buscar_palabra.ps1 <palabra_buscada> <archivo_de_búsqueda>"

exit 1

}

Get parameters

\$palabra = \$args[0]

\$archivo = \$args[1]

Check if the file exists

if (-not (Test-Path -Path \$archivo)) {

Write-Host "Error: El archivo '\$archivo' no existe."

exit 1

}

Search for the word in the file

Write-Host "Buscando '\$palabra' en el archivo '\$archivo':"

Write-Host "-----"

Using Select-String which is PowerShell's equivalent to grep

\$resultados = Select-String -Path \$archivo -Pattern \$palabra -SimpleMatch

if (\$resultados) {

\$resultados | ForEach-Object { Write-Host \$_.Line }

Write-Host "-----"

Write-Host "Se encontraron \$(\$resultados.Count) coincidencias."

} else {

Write-Host "No se encontraron coincidencias para '\$palabra'."

}

Ahora se valida con Shellcheck:

Paste a script to try it out:



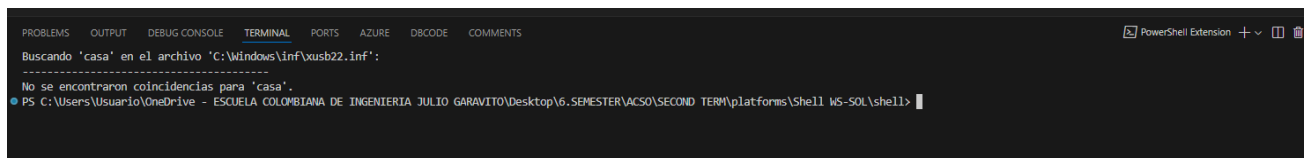
The screenshot shows a code editor window titled "Your Editor (Ace)" with a shell script. Below the editor is a terminal window titled "ShellCheck Output" showing the result of running shellcheck on the script.

```
1 #!/bin/sh
2 # shellcheck shell=sh
3 # Script to search for a word in a specified file (Shell representation of PowerShell)
4
5 # Clear the screen
6 clear
7
8 # Check if correct number of parameters were provided
9 if [ $# -ne 2 ]; then
10     echo "Error: Se requieren dos parámetros."
11     echo "Uso: ./buscar_palabra.sh <palabra_buscada> <archivo_de_búsqueda>"
12     exit 1
13 fi
14
15 # Get parameters
16 palabra="$1"
17 archivo="$2"
18
19 # Check if the file exists
20 if [ ! -f "$archivo" ]; then
21     echo "Error: El archivo '$archivo' no existe."
22     exit 1
23 fi
```

```
$ shellcheck myscript
No issues detected!
$
```

ShellCheck is

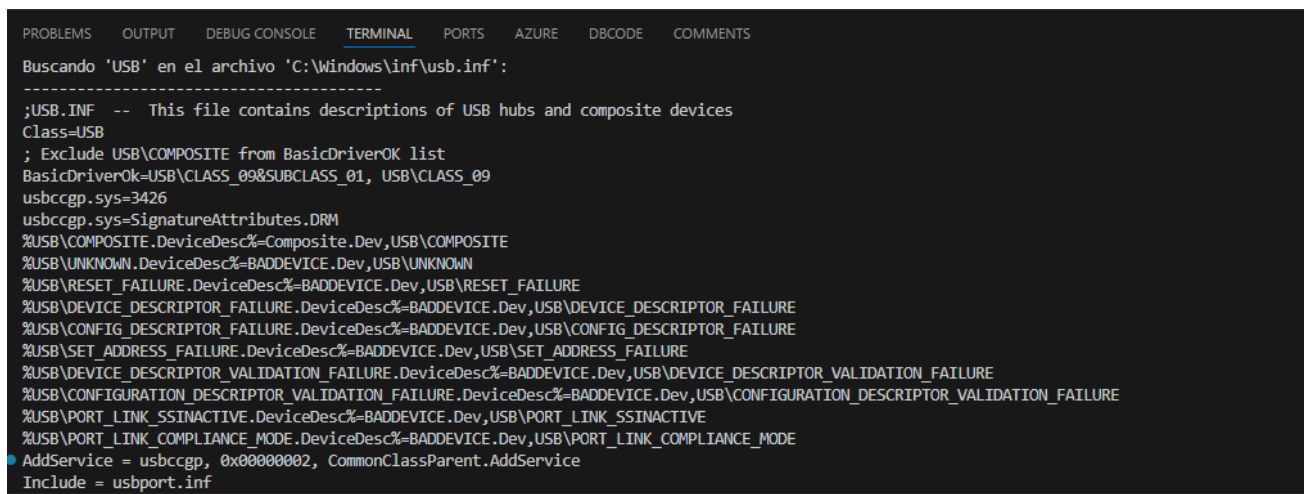
Se ejecuta con `.\buscar_palabra.ps1 casa C:\Windows\inf\xusb22.inf`, generando el output correcto.



The screenshot shows a PowerShell terminal window with the command `.\buscar_palabra.ps1 casa C:\Windows\inf\xusb22.inf` and its output.

```
Buscando 'casa' en el archivo 'C:\Windows\inf\xusb22.inf':
-----
No se encontraron coincidencias para 'casa'.
```

Ahora, para encontrar coincidencias, se pone el comando **Get-ChildItem -Path "C:\Windows\inf" -File | Select-Object Name** para listar todos los archivos. Se elige uno, y se vuelve a ejecutar con el nuevo nombre, por ejemplo, `.\buscar_palabra.ps1 USB C:\Windows\inf\usb.inf`



The screenshot shows a PowerShell terminal window with the command `.\buscar_palabra.ps1 USB C:\Windows\inf\usb.inf` and its output.

```
Buscando 'USB' en el archivo 'C:\Windows\inf\usb.inf':
-----
;USB.INF -- This file contains descriptions of USB hubs and composite devices
Class=USB
; Exclude USB\COMPOSITE from BasicDriverOK list
BasicDriverOk=USB\CLASS_09&SUBCLASS_01, USB\CLASS_09
usbccgp.sys=3426
usbccgp.sys=SignatureAttributes.DRM
%USB\COMPOSITE.DeviceDesc%=Composite.Dev,USB\COMPOSITE
%USB\UNKNOWN.DeviceDesc%=BADDEVICE.Dev,USB\UNKNOWN
%USB\RESET_FAILURE.DeviceDesc%=BADDEVICE.Dev,USB\RESET_FAILURE
%USB\DEVICE_DESCRIPTOR_FAILURE.DeviceDesc%=BADDEVICE.Dev,USB\DEVICE_DESCRIPTOR_FAILURE
%USB\CONFIG_DESCRIPTOR_FAILURE.DeviceDesc%=BADDEVICE.Dev,USB\CONFIG_DESCRIPTOR_FAILURE
%USB\SET_ADDRESS_FAILURE.DeviceDesc%=BADDEVICE.Dev,USB\SET_ADDRESS_FAILURE
%USB\DEVICE_DESCRIPTOR_VALIDATION_FAILURE.DeviceDesc%=BADDEVICE.Dev,USB\DEVICE_DESCRIPTOR_VALIDATION_FAILURE
%USB\CONFIGURATION_DESCRIPTOR_VALIDATION_FAILURE.DeviceDesc%=BADDEVICE.Dev,USB\CONFIGURATION_DESCRIPTOR_VALIDATION_FAILURE
%USB\PORT_LINK_SSINACTIVE.DeviceDesc%=BADDEVICE.Dev,USB\PORT_LINK_SSINACTIVE
%USB\PORT_LINK_COMPLIANCE_MODE.DeviceDesc%=BADDEVICE.Dev,USB\PORT_LINK_COMPLIANCE_MODE
AddService = usbccgp, 0x00000002, CommonClassParent.AddService
Include = usbport.inf
```

UNIX/SOLARIS SHELL SCRIPT

Primero, se guarda el archivo buscar_palabra.sh y dentro de este, se agrega los comandos necesarios para cumplir con este objetivo, queda así:

```
#!/bin/sh

# Script to search for a word in a specified file


# Clear the screen

clear


# Check if correct number of parameters were provided
if [ $# -ne 2 ]; then
    echo "Error: Se requieren dos parámetros."
    echo "Uso: ./buscar_palabra.sh <palabra_buscada> <archivo_de_búsqueda>"
    exit 1
fi


# Get parameters
palabra="$1"
archivo="$2"


# Check if the file exists
if [ ! -f "$archivo" ]; then
    echo "Error: El archivo '$archivo' no existe."
    exit 1
fi


# Search for the word in the file
echo "Buscando '$palabra' en el archivo '$archivo':"
echo "-----"


# Using grep to find matching lines
resultados=$(grep -i "$palabra" "$archivo")
if [ -n "$resultados" ]; then
    echo "$resultados"
    echo "-----"
    coincidencias=$(grep -i -c "$palabra" "$archivo")
```

Shell Programming - Nombre: Andersson David Sánchez Méndez ACSO-2 – 1L

```
    echo "Se encontraron $coincidencias coincidencias."
else
    echo "No se encontraron coincidencias para '$palabra'."
Fi
```

Ahora se valida con Shellcheck:

You can copy, paste, edit, or even execute it directly right now.

Paste a script to try it out:

```
Load random example
1 #!/bin/sh
2 # Script to search for a word in a specified file
3
4 # Clear the screen
5 clear
6
7 # Check if correct number of parameters were provided
8 if [ $# -ne 2 ]; then
9     echo "Error: Se requieren dos parámetros."
10    echo "Uso: ./buscar_palabra.sh <palabra_buscada> <archivo_de_búsqueda>"
11    exit 1
12 fi
13
14 # Get parameters
15 palabra="$1"
16 archivo="$2"
17
18 # Check if the file exists
19 if [ ! -f "$archivo" ]; then
20     echo "Error: El archivo '$archivo' no existe."
21     exit 1
22 fi
```

ShellCheck Output

```
$ shellcheck myscript
No issues detected!
$
```

Se hace ejecutable con **chmod +x buscar_palabra.sh** y se ejecuta con **./buscar_palabra.sh casa /etc/passwd**, generando el output correcto.

```
Error: El archivo '/etc/passwd' no existe.
```

Usuario ~ > ... > platforms > Shell WS-SOL > shell

Ahora, para que muestre algo que exista, hacemos **ls /etc/**, y luego si un comando sobre un directorio que exista, por ejemplo, **./buscar_palabra.sh bash /etc/bash.bashrc**.

```
Buscando 'bash' en el archivo '/etc/bash.bashrc':
-----
# /etc/bash.bashrc: executed by bash(1) for interactive shells.
# System-wide bashrc file
# Fixup git-bash in non login env
-----
Se encontraron 4 coincidencias.
```

Usuario ~ > ... > platforms > Shell WS-SOL > shell

4. Uso de repeticiones y almacenamiento de la respuesta de ejecución de un comando en un archivo

Cree 5 usuarios en su sistema. Incluya descripción para cada uno de ellos.

Escriba un programa Shell (y revíselo en ShellCheck) que:

- Del archivo `/etc/passwd` extraiga SOLAMENTE los nombres de los usuarios y la descripción de los mismos
- Deje el resultado de la ejecución en otro archivo

Modifique el script para PowerShell usando comandos de Windows
y en Unix para Solaris.

POWERSHELL SCRIPT

Primero, se guarda el archivo `crear_usuarios.ps1` y dentro de este, se agrega los comandos necesarios para cumplir con este objetivo, y como segundo paso se crea el archivo de `extraer_usuarios.ps1`, para luego de ejecutar estos dos archivos se genere `usuariosP_descripcion.txt`.

```
# Este script crea 5 usuarios de prueba con descripciones
```

```
# Nota: Debe ejecutarse con privilegios de administrador
```

```
# Verificar si se está ejecutando como administrador
```

```
$esAdmin = ([Security.Principal.WindowsPrincipal]  
[Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole([Security.Principal.WindowsBuiltInRole]::  
Administrator)
```

```
if (-not $esAdmin) {
```

```
    Write-Host "Este script debe ejecutarse con privilegios de administrador"
```

```
    exit 1
```

```
}
```

```
# Función para crear usuario
```

```
function Crear-Usuario {
```

```
    param (
```

```
        [string]$nombre,
```

```
        [string]$descripcion,
```

```
        [string]$password = "P@ssw0rd123"
```

```
)
```

Shell Programming - Nombre: Andersson David Sánchez Méndez ACSO-2 – 1L

```
# Convertir la contraseña a SecureString

$securePassword = ConvertTo-SecureString $password -AsPlainText -Force


# Verificar si el usuario ya existe

$usuarioExiste = Get-LocalUser -Name $nombre -ErrorAction SilentlyContinue


if ($usuarioExiste) {

    Write-Host "El usuario '$nombre' ya existe. Actualizando descripción..."

    Set-LocalUser -Name $nombre -Description $descripcion

} else {

    # Crear el usuario

    New-LocalUser -Name $nombre -Password $securePassword -Description $descripcion -
    FullName $nombre

    Write-Host "Usuario '$nombre' creado con descripción '$descripcion'"

}

}


# Crear usuarios con descripciones

Write-Host "Creando usuarios de prueba..."


# Usuario 1

Crear-Usuario -nombre "usuario_admin" -descripcion "Administrador del Sistema"


# Usuario 2

Crear-Usuario -nombre "usuario_dev" -descripcion "Desarrollador Web"


# Usuario 3

Crear-Usuario -nombre "usuario_dba" -descripcion "Analista de Base de Datos"


# Usuario 4
```

Shell Programming - Nombre: Andersson David Sánchez Méndez ACSO-2 – 1L

```
Crear-Usuario -nombre "usuario_soporte" -descripcion "Soporte Técnico"
```

```
# Usuario 5
```

```
Crear-Usuario -nombre "usuario_test" -descripcion "Estudiante de Prueba"
```

```
Write-Host "Todos los usuarios han sido creados correctamente."
```

```
Write-Host "Puedes ejecutar 'Get-LocalUser usuario_*' para verificarlos."
```

```
exit 0
```

Se valida con Shellcheck:

```
# Definir el archivo de salida
```

```
$ARCHIVO_SALIDA = "usuarios_descripcion.txt"
```

```
# Limpiar el archivo de salida si ya existe
```

```
if (Test-Path $ARCHIVO_SALIDA) {
```

```
    Clear-Content $ARCHIVO_SALIDA
```

```
}
```

```
# Encabezado del archivo de salida
```

```
"===== " | Out-File -FilePath $ARCHIVO_SALIDA
```

```
"USUARIO | DESCRIPCIÓN" | Out-File -FilePath $ARCHIVO_SALIDA -Append
```

```
"===== " | Out-File -FilePath $ARCHIVO_SALIDA -  
Append
```

```
# Obtener información de usuarios locales
```

```
Write-Host "Extrayendo información de usuarios..."
```

```
$usuarios = Get-LocalUser | Select-Object Name, Description
```

Shell Programming - Nombre: Andersson David Sánchez Méndez ACSO-2 – 1L

```
# Escribir la información en el archivo de salida

foreach ($usuario in $usuarios) {

    # Solo incluir usuarios que tengan una descripción

    if ($usuario.Description) {

        "$($usuario.Name) | $($usuario.Description)" | Out-File -FilePath $ARCHIVO_SALIDA -Append

    }

}

Write-Host "La información ha sido guardada en $ARCHIVO_SALIDA"

Write-Host "Contenido del archivo de salida:"

Write-Host "-----"

Get-Content $ARCHIVO_SALIDA

exit 0
```

Se valida con Shellcheck:

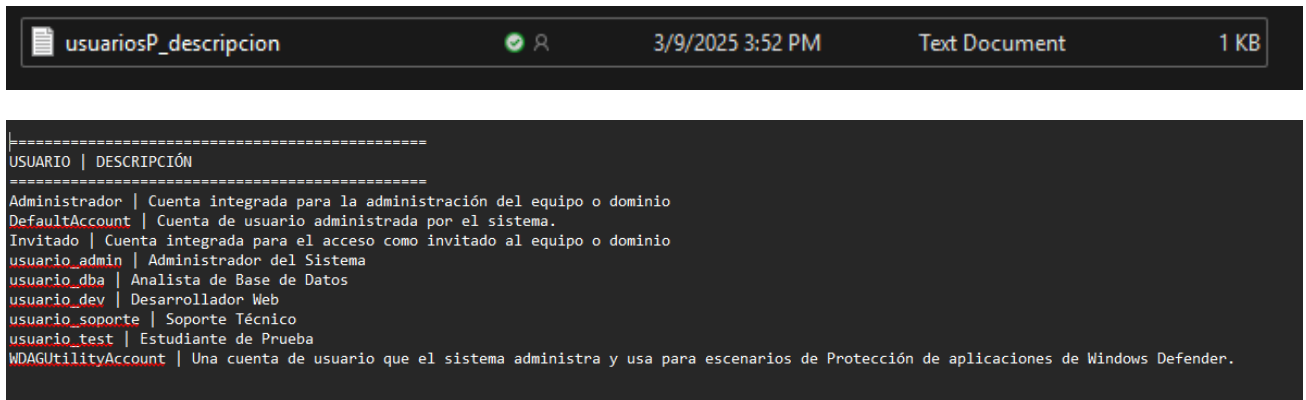
Se ejecuta con `.\crear_usuarios.ps1` y luego `.\extraer_usuarios.ps1`, generando el output correcto.
(modo admin)

```
PS C:\Usuario - root [ 2025-03-09 15:51:36 ] [ ~/OneDrive - ESCUELA COLOMBIANA DE INGENIERIA JULIO GARAVITO/Desktop/6.SEMESTER/ACSO/SECOND TERM/platforms/Shell WS-SOL/shell ]
➔ .\crear_usuarios.ps1
Creando usuarios de prueba...

Usuario 'usuario_admin' creado con descripción 'Administrador del Sistema'
Usuario 'usuario_dev' creado con descripción 'Desarrollador Web'
Usuario 'usuario_dba' creado con descripción 'Analista de Base de Datos'
Usuario 'usuario_soporte' creado con descripción 'Soporte Técnico'
Usuario 'usuario_test' creado con descripción 'Estudiante de Prueba'
Todos los usuarios han sido creados correctamente.
Puedes ejecutar 'Get-LocalUser usuario_*' para verificarlos.
Name      Enabled Description
-----
usuario_admin True    Administrador del Sistema
usuario_dev True    Desarrollador Web
usuario_dba True    Analista de Base de Datos
usuario_soporte True    Soporte Técnico
usuario_test True    Estudiante de Prueba

PS C:\Usuario - root [ 2025-03-09 15:51:52 ] [ ~/OneDrive - ESCUELA COLOMBIANA DE INGENIERIA JULIO GARAVITO/Desktop/6.SEMESTER/ACSO/SECOND TERM/platforms/Shell WS-SOL/shell ]
➔ .\extraer_usuarios.ps1
Extrayendo información de usuarios...
La información ha sido guardada en usuariosP_description.txt
Contenido del archivo de salida:
-----
USUARIO | DESCRIPCIÓN
-----
Administrador | Cuenta integrada para la administración del equipo o dominio
DefaultAccount | Cuenta de usuario administrada por el sistema.
Invitado | Cuenta integrada para el acceso como invitado al equipo o dominio
usuario_admin | Administrador del Sistema
usuario_dba | Analista de Base de Datos
usuario_dev | Desarrollador Web
usuario_soporte | Soporte Técnico
usuario_test | Estudiante de Prueba
WDAGUtilityAccount | Una cuenta de usuario que el sistema administra y usa para escenarios de Protección de aplicaciones de Windows Defender.
```

Y se mira el archivo generado .txt



usuariosP_descripcion 3/9/2025 3:52 PM Text Document 1 KB

```
=====
USUARIO | DESCRIPCIÓN
=====
Administrador | Cuenta integrada para la administración del equipo o dominio
DefaultAccount | Cuenta de usuario administrada por el sistema.
Invitado | Cuenta integrada para el acceso como invitado al equipo o dominio
usuario_admin | Administrador del Sistema
usuario_dba | Analista de Base de Datos
usuario_dev | Desarrollador Web
usuario_soporte | Soporte Técnico
usuario_test | Estudiante de Prueba
WDAGUtilityAccount | Una cuenta de usuario que el sistema administra y usa para escenarios de Protección de aplicaciones de Windows Defender.
```

UNIX/SOLARIS SHELL SCRIPT

Primero, se guarda el archivo crear_usuarios.sh y dentro de este, se agrega los comandos necesarios para cumplir con este objetivo, y como segundo paso se crea el archivo de extraer_usuarios.sh, para luego de ejecutar estos dos archivos se genere usuariosS_descripcion.txt.

```
#!/bin/sh
```

```
# Este script crea 5 usuarios de prueba con descripciones
```

```
# Nota: Debe ejecutarse con privilegios de superusuario (su)
```

```
# Verificar si se está ejecutando como root
```

```
if [ `usr/bin/id -u` -ne 0 ]; then
```

```
    echo "Este script debe ejecutarse con privilegios de superusuario (su)"
```

```
    exit 1
```

```
fi
```

```
# Crear usuarios con descripciones
```

```
echo "Creando usuarios de prueba..."
```

```
# Usuario 1 - En Solaris se usa useradd con parámetros diferentes
```

```
/usr/sbin/useradd -c "Administrador del Sistema" -m -d /export/home/usuario_admin usuario_admin
```


Shell Programming - Nombre: Andersson David Sánchez Méndez ACSO-2 – 1L

```
echo "Usuario 'usuario_admin' creado con descripción 'Administrador del Sistema'"
```

```
# Usuario 2
```

```
/usr/sbin/useradd -c "Desarrollador Web" -m -d /export/home/usuario_dev usuario_dev
```

```
echo "Usuario 'usuario_dev' creado con descripción 'Desarrollador Web'"
```

```
# Usuario 3
```

```
/usr/sbin/useradd -c "Analista de Base de Datos" -m -d /export/home/usuario_dba usuario_dba
```

```
echo "Usuario 'usuario_dba' creado con descripción 'Analista de Base de Datos'"
```

```
# Usuario 4
```

```
/usr/sbin/useradd -c "Soporte Técnico" -m -d /export/home/usuario_soporte usuario_soporte
```

```
echo "Usuario 'usuario_soporte' creado con descripción 'Soporte Técnico'"
```

```
# Usuario 5
```

```
/usr/sbin/useradd -c "Estudiante de Prueba" -m -d /export/home/usuario_test usuario_test
```

```
echo "Usuario 'usuario_test' creado con descripción 'Estudiante de Prueba'"
```

```
echo "Todos los usuarios han sido creados correctamente."
```

```
echo "Puedes ejecutar 'grep usuario_/etc/passwd' para verificarlos."
```

```
exit 0
```

Se valida con Shellcheck:

```
#!/bin/sh
```

```
# Definir el archivo de salida
```

```
ARCHIVO_SALIDA="usuarios_descripcion.txt"
```

Shell Programming - Nombre: Andersson David Sánchez Méndez ACSO-2 – 1L

```
# Limpiar el archivo de salida si ya existe
> "$ARCHIVO_SALIDA"

# Extraer nombres de usuario y descripciones del archivo /etc/passwd
echo "Extrayendo nombres de usuario y descripciones..."
echo "===== " >> "$ARCHIVO_SALIDA"
echo "USUARIO | DESCRIPCIÓN" >> "$ARCHIVO_SALIDA"
echo "===== " >> "$ARCHIVO_SALIDA"

# En Solaris, usamos comandos específicos del sistema
# El campo GECOS está en la quinta columna del archivo passwd
/usr/bin/cut -d: -f1,5 /etc/passwd | while read linea
do
    # Separar el nombre de usuario y la descripción usando cut de Solaris
    USUARIO=`echo "$linea" | /usr/bin/cut -d: -f1`
    DESCRIPCION=`echo "$linea" | /usr/bin/cut -d: -f2`

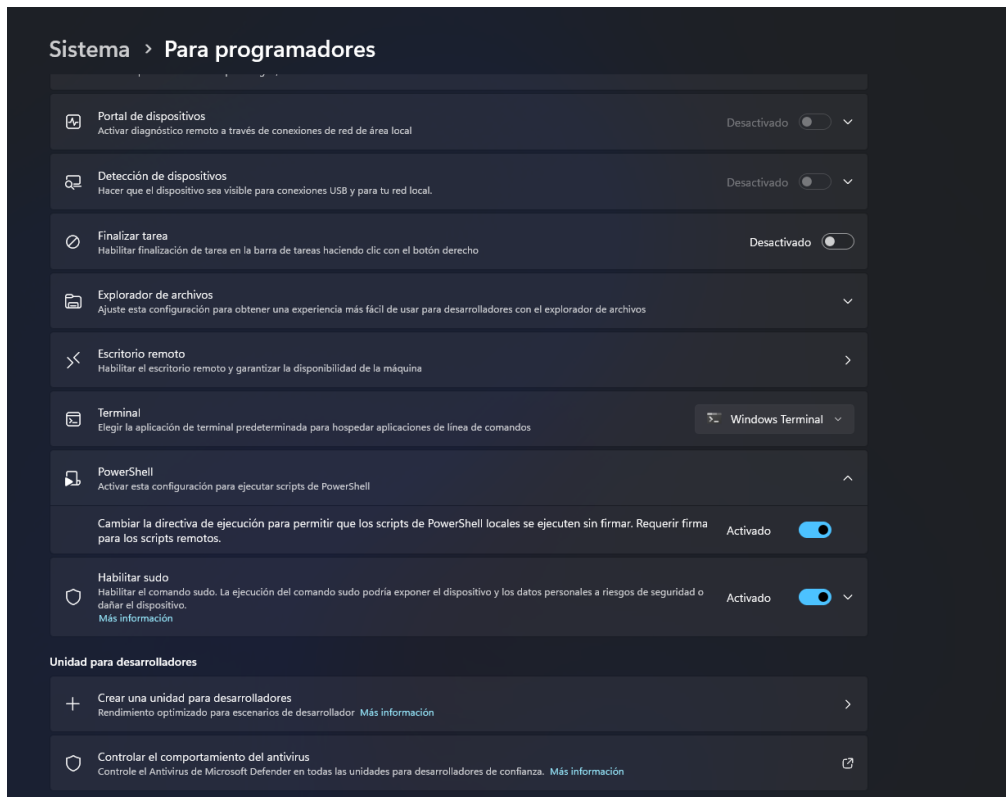
    # Solo incluir entradas que tengan una descripción
    if [ ! -z "$DESCRIPCION" ]; then
        echo "$USUARIO | $DESCRIPCION" >> "$ARCHIVO_SALIDA"
    fi
done

echo "La información ha sido guardada en $ARCHIVO_SALIDA"
echo "Contenido del archivo de salida:"
echo "-----"
/usr/bin/cat "$ARCHIVO_SALIDA"

exit 0
```

Se valida con Shellcheck:

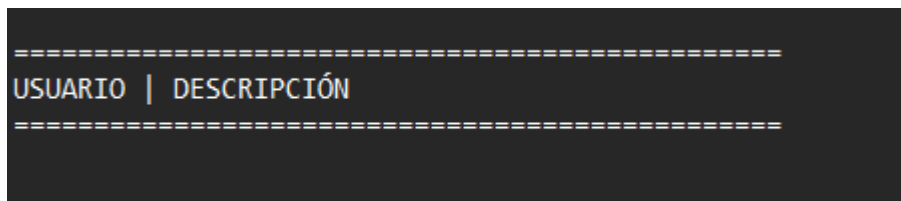
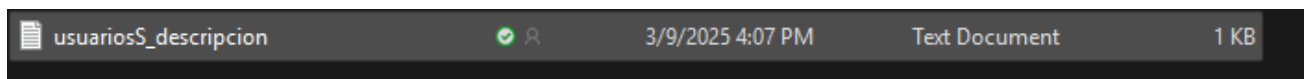
Se hace ejecutable con **chmod +x crear_usuarios.sh extraer_usuarios.sh** y se ejecuta con **sudo ./crear_usuarios.sh** o **su -c "./crear_usuarios.sh"** y luego **./extraer_usuarios.sh**, generando el output correcto.



```
Usuario ~ > ... > platforms > Shell WS-SOL > shell
$ chmod +x crear_usuarios.sh extraer_usuarios.sh

Usuario ~ > ... > platforms > Shell WS-SOL > shell
$ sudo ./crear_usuarios.sh
El usuario canceló la operación

Usuario ~ > ... > platforms > Shell WS-SOL > shell
$ ./extraer_usuarios.sh
Extrayendo nombres de usuario y descripciones...
usr/bin/cut: /etc/passwd: No such file or directory
La información ha sido guardada en usuariosS_descripcion.txt
Contenido del archivo de salida:
=====
USUARIO | DESCRIPCIÓN
=====
```



Shell Programming - Nombre: Andersson David Sánchez Méndez ACSO-2-1L

Para que me sirva en gitbash fue necesario descargar Ubuntu y crear mi usuario para tener el terminal de Linux y poder ejecutar los comandos correspondientes con el superusuario.

```
Usuario ~\platforms> Shell WS-SOL > shell
$ wsl --install
Instalando: Ubuntu
Se ha instalado Ubuntu.
Iniciando Ubuntu...
Installing, this may take a few minutes!!!
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username: andersson
New password:
Retype new password:
passwd: password updated successfully
Installation successful!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 5.15.167.4-microsoft-standard-WSL2 x86_64)

 * Documentation: https://help.ubuntu.com
 * Management:   https://landscape.canonical.com
 * Support:      https://ubuntu.com/pro

System information as of Sun Mar  9 16:15:46 -05 2025

System load:  0.22               Processes:            67
Usage of /:   0.1% of 1006.85GB   Users logged in:     0
Memory usage: 2%                IPv4 address for eth0: 172.31.215.74
Swap usage:   0%

This message is shown once a day. To disable it please create the
/home/andersson/.hushlogin file.
andersson@DESKTOP-EMM939Q:~$
```

```
andersson@DESKTOP-EMM939Q:/mnt/c/Users/Usuario/OneDrive - ESCUELA COLOMBIANA DE INGENIERIA JULIO GARAVITO/Desktop/6.SEMESTER/ACSO/SECOND TERM/platforms/Shell WS-SOL$ ls
ShellProgramming.docx ShellProgramming.pdf
andersson@DESKTOP-EMM939Q:/mnt/c/Users/Usuario/OneDrive - ESCUELA COLOMBIANA DE INGENIERIA JULIO GARAVITO/Desktop/6.SEMESTER/ACSO/SECOND TERM/platforms/Shell WS-SOL$ cd shell
andersson@DESKTOP-EMM939Q:/mnt/c/Users/Usuario/OneDrive - ESCUELA COLOMBIANA DE INGENIERIA JULIO GARAVITO/Desktop/6.SEMESTER/ACSO/SECOND TERM/platforms/Shell WS-SOL/shell$
andersson@DESKTOP-EMM939Q:/mnt/c/Users/Usuario/OneDrive - ESCUELA COLOMBIANA DE INGENIERIA JULIO GARAVITO/Desktop/6.SEMESTER/ACSO/SECOND TERM/platforms/Shell WS-SOL/shell$ chmod +x crear_usuarios.sh
andersson@DESKTOP-EMM939Q:/mnt/c/Users/Usuario/OneDrive - ESCUELA COLOMBIANA DE INGENIERIA JULIO GARAVITO/Desktop/6.SEMESTER/ACSO/SECOND TERM/platforms/Shell WS-SOL/shell$ sudo ./crear_usuarios.sh
[sudo] password for andersson:
creando usuarios de prueba...
useradd: warning: chown on '/export/home/usuario_admin' failed: No such file or directory
useradd: warning: chown on '/export/home/usuario_admin' failed: No such file or directory
usuario 'usuario_admin' creado con descripción 'Administrador del Sistema'
usuario 'usuario_dev' creado con descripción 'Desarrollador Web'
usuario 'usuario_dba' creado con descripción 'Analista de Base de Datos'
usuario 'usuario_soporte' creado con descripción 'Soporte Técnico'
usuario 'usuario_test' creado con descripción 'Estudiante de Prueba'
Todos los usuarios han sido creados correctamente.
Puedes ejecutar 'grep usuario_ /etc/passwd' para verificarlos.
```

```
andersson@DESKTOP-EMM939Q:/mnt/c/Users/Usuario/OneDrive - ESCUELA COLOMBIANA DE INGENIERIA JULIO GARAVITO/Desktop/6.SEMESTER/ACSO/SECOND TERM/platforms/Shell WS-SOL/shell$ ./extraer_usuarios.sh
Extrayendo nombres de usuario y descripciones...
La información ha sido guardada en usuarios5_descripcion.txt
Contenido del archivo de salida:
=====
USUARIO | DESCRIPCIÓN
=====
root | root
daemon | daemon
bin | bin
sys | sys
sync | sync
games | games
man | man
lp | lp
mail | mail
news | news
uucp | uucp
proxy | proxy
www-data | www-data
backup | backup
lister | Mailing List Manager
irc | ircd
nobody | nobody
systemd-network | system Network Management
systemd-timesync | system Time Synchronization
dhcpd | DHCP Client Daemon
systemd-resolve | system Resolver
polkitd | User for polkitd
andersson |
usuario_admin | Administrador del Sistema
usuario_dev | Desarrollador Web
usuario_dba | Analista de Base de Datos
usuario_soporte | Soporte Técnico
usuario_test | Estudiante de Prueba
```

Y en el .txt:

```
=====
USUARIO | DESCRIPCIÓN
=====
root | root
daemon | daemon
bin | bin
sys | sys
sync | sync
games | games
man | man
lp | lp
mail | mail
news | news
uucp | uucp
proxy | proxy
www-data | www-data
backup | backup
list | Mailing List Manager
irc | ircd
nobody | nobody
systemd-network | systemd Network Management
systemd-timesync | systemd Time Synchronization
dhcpcd | DHCP Client Daemon,,,
systemd-resolve | systemd Resolver
polkitd | User for polkitd
andersson | ,,,
usuario_admin | Administrador del Sistema
usuario_dev | Desarrollador Web
usuario_dba | Analista de Base de Datos
usuario_soporte | Soporte Técnico
usuario_test | Estudiante de Prueba
```

5. Manejo de condicionales

Al listar las características de un archivo con el comando `ls -l`, el campo de permisos está compuesto por 10 caracteres, ¿Qué significa cada uno de ellos? y qué valores pueden tomar?

```
andersson@DESKTOP-EMM939Q:/mnt/c/Users/usuario/OneDrive - ESCUELA COLOMBIANA DE INGENIERIA JULIO GARAVITO/desktop/6.SEMESTER/ACSO/SECOND TERM/platforms/Shell WS-SOL/shell$ ls -ll
total 36
-rwxrwxrwx 1 andersson andersson 1103 Mar  9 14:34 buscar_palabra.ps1
-rwxrwxrwx 1 andersson andersson 966 Mar  9 14:40 buscar_palabra.sh
-rwxrwxrwx 1 andersson andersson 337 Mar  9 13:14 count_lines.ps1
-rwxrwxrwx 1 andersson andersson 221 Mar  9 13:21 count_lines.sh
-rwxrwxrwx 1 andersson andersson 1994 Mar  9 15:29 crear_usuarios.ps1
-rwxrwxrwx 1 andersson andersson 1474 Mar  9 15:31 crear_usuarios.sh
-rwxrwxrwx 1 andersson andersson 1142 Mar  9 15:31 extraer_usuarios.ps1
-rwxrwxrwx 1 andersson andersson 1179 Mar  9 15:32 extraer_usuarios.sh
-rwxrwxrwx 1 andersson andersson 136 Mar  9 12:53 hello.ps1
-rwxrwxrwx 1 andersson andersson 125 Mar  9 12:31 hello.sh
-rwxrwxrwx 1 andersson andersson 127 Mar  9 13:00 hello1.sh
-rwxrwxrwx 1 andersson andersson 680 Mar  9 15:52 usuariosP_descripcion.txt
-rwxrwxrwx 1 andersson andersson 741 Mar  9 16:23 usuarios5_descripcion.txt
```

Primer carácter: Tipo de archivo

- : Archivo regular
- d: Directorio
- l: Enlace simbólico
- c: Archivo especial de caracteres
- b: Archivo especial de bloques
- p: FIFO (tubería con nombre)
- s: Socket

Caracteres 2-4: Permisos del propietario (usuario)

- r: Permiso de lectura
- w: Permiso de escritura
- x: Permiso de ejecución
- : Sin permiso

Caracteres 5-7: Permisos del grupo

r: Permiso de lectura
w: Permiso de escritura
x: Permiso de ejecución
-: Sin permiso

Caracteres 8-10: Permisos de otros (resto del mundo)

r: Permiso de lectura
w: Permiso de escritura
x: Permiso de ejecución
-: Sin permiso

Además, los caracteres x pueden ser reemplazados por:

s: Set-UID o Set-GID (en posiciones de usuario o grupo)
t: Sticky bit (en posición de otros)

Escriba un programa Shell (y revíselo en ShellCheck) que:

- Liste los archivos ubicados en un directorio dado por el usuario que tengan los permisos buscados por el mismo usuario.
Modifique el script para PowerShell usando comandos de Windows
- La ejecución sería del estilo: *buscar_archivos.sh /etc/ -rw-r-r-*
- Muestre el resultado en pantalla

Modifique el script para PowerShell usando comandos de Windows y en Unix para Solaris.

La ejecución sería del estilo: *buscar_archivos.ps1 C:\Windows\inf \FullControl*

POWERSHELL SCRIPT

Primero, se guarda el archivo `buscar_archivos.ps1` y dentro de este, se agrega los comandos necesarios para cumplir con este objetivo.

Script para buscar archivos con permisos específicos en Windows

Uso: `.\buscar_archivos.ps1 C:\Windows\inf FullControl`

param(

[Parameter(Mandatory=\$true, Position=0)]

[string]\$Directorio,

[Parameter(Mandatory=\$true, Position=1)]

[string]\$Permisos

)

Verificar si el directorio existe

if (-not (Test-Path -Path \$Directorio -PathType Container)) {

Write-Host "Error: El directorio '\$Directorio' no existe"

exit 1

}

Mapeo de permisos comunes a derechos de acceso en PowerShell

\$MapeoPermisos = @{

"FullControl" = [System.Security.AccessControl.FileSystemRights]::FullControl

"Read" = [System.Security.AccessControl.FileSystemRights]::Read

"Write" = [System.Security.AccessControl.FileSystemRights]::Write

"ReadAndExecute" = [System.Security.AccessControl.FileSystemRights]::ReadAndExecute

"Modify" = [System.Security.AccessControl.FileSystemRights]::Modify

}

Verificar si el permiso existe en el mapeo

if (-not \$MapeoPermisos.ContainsKey(\$Permisos)) {

Write-Host "Error: Permiso '\$Permisos' no reconocido"

Write-Host "Permisos válidos: FullControl, Read, Write, ReadAndExecute, Modify"

exit 1

}

\$PermisoBuscado = \$MapeoPermisos[\$Permisos]

Write-Host "Buscando archivos en '\$Directorio' con permisos '\$Permisos'..."

Obtener archivos del directorio

\$Archivos = Get-ChildItem -Path \$Directorio -File

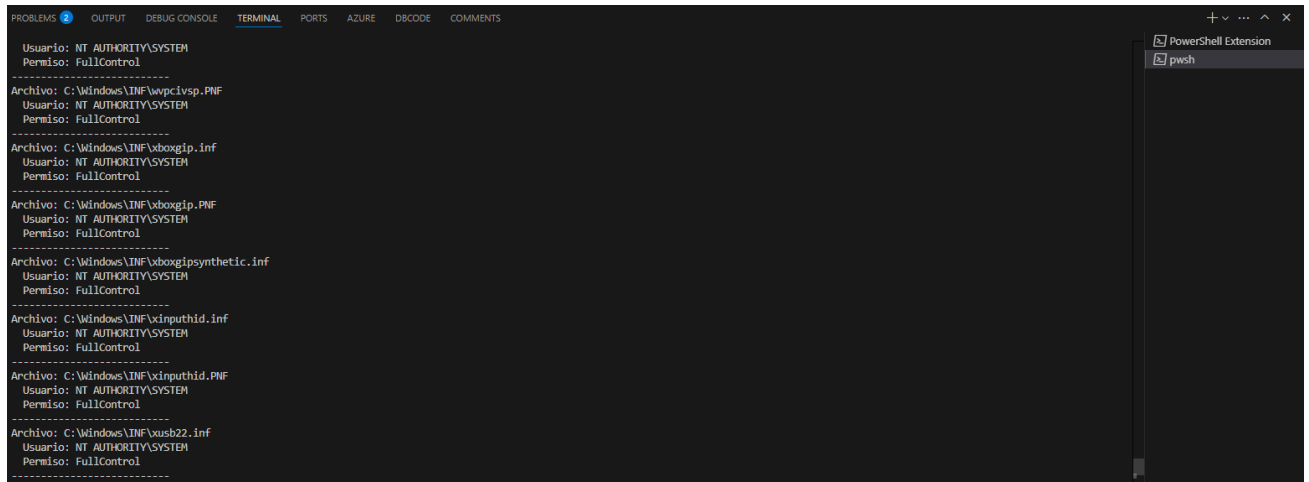
```
foreach ($Archivo in $Archivos) {  
    try {  
        $Acl = Get-Acl $Archivo.FullName  
        $Accesos = $Acl.Access  
  
        # Verificar si alguna regla de acceso coincide con el permiso buscado  
        foreach ($Acceso in $Accesos) {  
            if (($Acceso.FileSystemRights -band $PermisoBuscado) -eq $PermisoBuscado) {  
                Write-Host "Archivo: $($Archivo.FullName)"  
                Write-Host " Usuario: $($Acceso.IdentityReference)"  
                Write-Host " Permiso: $($Acceso.FileSystemRights)"  
                Write-Host "-----"  
                break  
            }  
        }  
    } catch {  
        Write-Host "Error al procesar $($Archivo.FullName): $_"  
    }  
}  
  
Write-Host "Búsqueda finalizada."
```

Se valida en Shellcheck:

Para permitir la ejecución de scripts en pwsh si está bloqueado se usa: **Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned.**

Se ejecuta con **.\buscar_archivos.ps1 C:\Windows\inf FullControl**, generando el output correcto.

Shell Programming - Nombre: Andersson David Sánchez Méndez ACSO-2 – 1L



```
Usuario: NT AUTHORITY\SYSTEM
Permiso: FullControl
-----
Archivo: C:\Windows\INF\wmpicvsp.PNF
Usuario: NT AUTHORITY\SYSTEM
Permiso: FullControl
-----
Archivo: C:\Windows\INF\vboxgip.inf
Usuario: NT AUTHORITY\SYSTEM
Permiso: FullControl
-----
Archivo: C:\Windows\INF\vboxgip.PNF
Usuario: NT AUTHORITY\SYSTEM
Permiso: FullControl
-----
Archivo: C:\Windows\INF\vboxgipsynthetic.inf
Usuario: NT AUTHORITY\SYSTEM
Permiso: FullControl
-----
Archivo: C:\Windows\INF\inputid.inf
Usuario: NT AUTHORITY\SYSTEM
Permiso: FullControl
-----
Archivo: C:\Windows\INF\inputid.PNF
Usuario: NT AUTHORITY\SYSTEM
Permiso: FullControl
-----
Archivo: C:\Windows\INF\xusb22.inf
Usuario: NT AUTHORITY\SYSTEM
Permiso: FullControl
-----
```

UNIX/SOLARIS SHELL SCRIPT

Primero, se guarda el archivo buscar_archivos.sh y dentro de este, se agrega los comandos necesarios para cumplir con este objetivo

```
#!/bin/sh
```

```
# Script para buscar archivos con permisos específicos en Solaris
```

```
# Uso: ./buscar_archivos_solaris.sh /ruta/directorio/ -rw-r--r--
```

```
# Verificar número de argumentos
```

```
if [ $# -ne 2 ]; then
```

```
    echo "Uso: $0 <directorio> <permisos>"
```

```
    echo "Ejemplo: $0 /etc/ -rw-r--r--"
```

```
    exit 1
```

```
fi
```

```
DIRECTORIO="$1"
```

```
PERMISOS="$2"
```

```
# Verificar que el directorio existe
```

```
if [ ! -d "$DIRECTORIO" ]; then
```

```
    echo "Error: El directorio '$DIRECTORIO' no existe"
```

Shell Programming - Nombre: Andersson David Sánchez Méndez ACSO-2 – 1L

```
exit 1

fi

# Verificar longitud del patrón de permisos

if [ `echo "$PERMISOS" | wc -c` -ne 11 ]; then

    echo "Error: Formato de permisos incorrecto. Debe ser como: -rw-r--r--"

    exit 1

fi

echo "Buscando archivos en '$DIRECTORIO' con permisos '$PERMISOS'..."

# Buscar archivos en Solaris

ls -l "$DIRECTORIO" | grep "^$PERMISOS" | while read -r line; do

    echo "$line"

done

echo "Búsqueda finalizada."
```

Se valida en Shellcheck:

Se hace ejecutable con **chmod +x buscar_archivos.sh** y se ejecuta con **./buscar_archivos.sh /etc/ -rw-r--r--**, generando el output correcto.

```
AnderssonDESKTOP [m3300] /mnt/c/Users/Userio/OneDrive - ESCUELA COLOMBIANA DE INGENIERIA JULIO GARAVITO/Desktop/6. SEMESTER/ACSO/SECOND TERM/platforms/Shell WS-SOL/shells ./buscar_archivos.sh /etc/ -rw-r--r--
Buscando archivos en '/etc/' con permisos '-rw-r--r--'...
-rw-r--r-- 1 root root 2444 Jul 5 2023 adduser.conf
-rw-r--r-- 1 root root 2310 Mar 31 2024 bash.bashrc
-rw-r--r-- 1 root root 45 Jan 24 2020 bash_completion
-rw-r--r-- 1 root root 987 Aug 2 2022 bindresvport.blacklist
-rw-r--r-- 1 root root 6280 Jan 6 15:14 ca-certificates.conf
-rw-r--r-- 1 root root 1136 Mar 30 2024 crontab
-rw-r--r-- 1 root root 2967 Apr 12 2024 debconf.conf
-rw-r--r-- 1 root root 11 Apr 22 2024 debian_version
-rw-r--r-- 1 root root 1706 Jul 5 2023 deluser.conf
-rw-r--r-- 1 root root 1429 Mar 31 2024 dhcpcd.conf
-rw-r--r-- 1 root root 685 Apr 8 2024 d2ccrdb.conf
-rw-r--r-- 1 root root 106 Jan 6 15:13 environment
-rw-r--r-- 1 root root 1853 Oct 17 2022 ethtypes
-rw-r--r-- 1 root root 27 Jan 6 15:13 fstab
-rw-r--r-- 1 root root 694 Apr 8 2024 fuse.conf
-rw-r--r-- 1 root root 2584 Jan 30 2024 gat.conf
-rw-r--r-- 1 root root 3986 Aug 7 2024 gprofng.rc
-rw-r--r-- 1 root root 938 Mar 9 16:23 group
-rw-r--r-- 1 root root 917 Mar 9 16:23 group-
-rw-r--r-- 1 root root 92 Apr 22 2024 host.conf
-rw-r--r-- 1 root root 16 Mar 9 16:16 hostname
-rw-r--r-- 1 root root 416 Mar 9 16:16 hosts
-rw-r--r-- 1 root root 1875 Mar 31 2024 lcpntrc
-rw-r--r-- 1 root root 26 Aug 23 2024 issue
-rw-r--r-- 1 root root 19 Aug 23 2024 issue.net
-rw-r--r-- 1 root root 17607 Mar 9 16:16 ld.so.cache
-rw-r--r-- 1 root root 34 Aug 2 2022 ld.so.conf
-rw-r--r-- 1 root root 267 Apr 22 2024 legal
-rw-r--r-- 1 root root 191 Mar 30 2024 libaudit.conf
-rw-r--r-- 1 root root 2900 Mar 30 2024 locale.alias
-rw-r--r-- 1 root root 13 Jan 6 15:14 locale.conf
-rw-r--r-- 1 root root 9565 Jan 6 15:14 locale.gen
-rw-r--r-- 1 root root 12345 Feb 22 2024 login.defs
-rw-r--r-- 1 root root 586 Apr 8 2024 logrotate.conf
-rw-r--r-- 1 root root 104 Aug 23 2024 lsb-release
-rw-r--r-- 1 root root 111 Mar 31 2024 magic
-rw-r--r-- 1 root root 111 Mar 31 2024 magic.mime
-rw-r--r-- 1 root root 5230 Apr 8 2024 mappath.config
-rw-r--r-- 1 root root 75112 Jul 12 2022 mime.types
-rw-r--r-- 1 root root 744 Apr 8 2024 mk2fs.conf
-rw-r--r-- 1 root root 212 Jan 6 15:14 modules
-rw-r--r-- 1 root root 1424 May 23 2022 namerc
-rw-r--r-- 1 root root 767 Mar 30 2024 netconfig
-rw-r--r-- 1 root root 91 Apr 22 2024 networks
-rw-r--r-- 1 root root 926 Jan 6 15:14 nsuitch.conf
```

6. Menú en Shell usando la instrucción case.

Escriba un programa Shell (y revíselo en ShellCheck) que:

Realice un menú usando Shell que permita escoger la ejecución de uno de los Shell anteriores todas las veces que se quiera

Adicionalmente incluya una opción Terminar que permita salir del Shell.

Tanto para Windows Core como Solaris.

POWERSHELL SCRIPT

Primero, se guarda el archivo menu.ps1 y dentro de este, se agrega los comandos necesarios para cumplir con este objetivo.

```
# Menu program for Windows PowerShell
```

```
# This script displays a menu and executes the selected previous exercises
```

```
function Show-Menu {
```

```
    Clear-Host
```

```
    Write-Host "=====
```

```
    Write-Host "  MENU DE PROGRAMAS SHELL - WINDOWS"
```

```
    Write-Host "=====
```

```
    Write-Host "1. Hello World"
```

```
    Write-Host "2. Número de líneas en C:\Windows\System32\drivers\etc\hosts"
```

```
    Write-Host "3. Buscar palabra en archivo"
```

```
    Write-Host "4. Extraer usuarios y descripción"
```

```
    Write-Host "5. Buscar archivos con permisos específicos"
```

```
    Write-Host "6. Terminar"
```

```
    Write-Host "=====
```

```
    Write-Host "Seleccione una opción (1-6): " -NoNewline
```

```
}
```

```
# Main program loop
```

```
$running = $true
```

```
while ($running) {
```

```
    # Display the menu
```

```
    Show-Menu
```

```
# Read user input
```

```
$option = Read-Host
```

Process the selected option using switch statement (PowerShell's equivalent to case)

```
switch ($option) {  
    "1" {  
        Write-Host "Ejecutando programa Hello World..."  
        # Suponiendo que el script se llama hello_world.ps1  
        & ".\hello.ps1"  
        Read-Host "Presione Enter para continuar..."  
    }  
    "2" {  
        Write-Host "Ejecutando programa de conteo de líneas..."  
        # Suponiendo que el script se llama contar_lineas.ps1  
        & ".\count_lines.ps1"  
        Read-Host "Presione Enter para continuar..."  
    }  
    "3" {  
        Write-Host "Ejecutando programa de búsqueda de palabras..."  
        $palabra = Read-Host "Ingrese la palabra a buscar"  
        $archivo = Read-Host "Ingrese el archivo donde buscar"  
        # Suponiendo que el script se llama buscar_palabra.ps1  
        & ".\buscar_palabra.ps1" $palabra $archivo  
        Read-Host "Presione Enter para continuar..."  
    }  
    "4" {  
        Write-Host "Ejecutando programa de extracción de usuarios..."  
        # Suponiendo que el script se llama extraer_usuarios.ps1  
        & ".\extraer_usuarios.ps1"  
        Read-Host "Presione Enter para continuar..."  
    }  
    "5" {  
        Write-Host "Ejecutando programa de búsqueda de archivos por permisos..."  
        $directorio = Read-Host "Ingrese el directorio donde buscar"  
        $permisos = Read-Host "Ingrese los permisos a buscar (ejemplo: FullControl)"  
        # Suponiendo que el script se llama buscar_archivos.ps1  
        & ".\buscar_archivos.ps1" $directorio $permisos  
        Read-Host "Presione Enter para continuar..."  
    }  
}
```

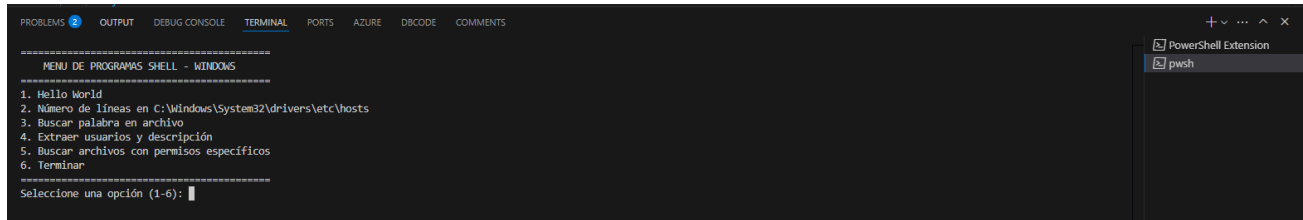
Shell Programming - Nombre: Andersson David Sánchez Méndez ACSO-2 – 1L

```
}  
"6" {  
    Write-Host "Saliendo del programa. ¡Hasta luego!"  
    $running = $false  
}  
default {  
    Write-Host "Opción inválida. Por favor, seleccione una opción válida (1-6)."  
    Read-Host "Presione Enter para continuar..."  
}  
}  
}  
}
```

Se valida en Shellcheck:

Para permitir la ejecución de scripts en pwsh si está bloqueado se usa: **Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser.**

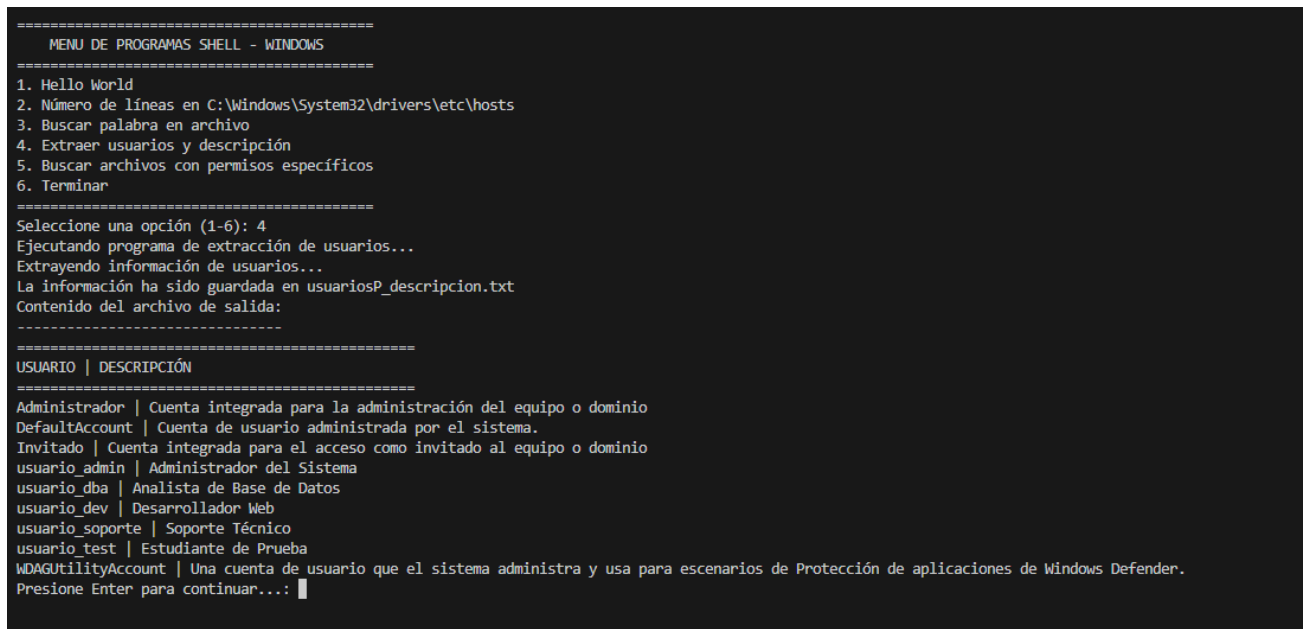
Se ejecuta con **./menu.ps1**, generando el output correcto.



The screenshot shows a PowerShell terminal window with the following output:

```
=====
MENU DE PROGRAMAS SHELL - WINDOWS
=====
1. Hello World
2. Número de líneas en C:\Windows\System32\drivers\etc\hosts
3. Buscar palabra en archivo
4. Extraer usuarios y descripción
5. Buscar archivos con permisos específicos
6. Terminar
=====
Seleccione una opción (1-6):
```

Y para cada opción funciona correctamente, por ejemplo, opción 4



The screenshot shows the terminal output after selecting option 4:

```
=====
MENU DE PROGRAMAS SHELL - WINDOWS
=====
1. Hello World
2. Número de líneas en C:\Windows\System32\drivers\etc\hosts
3. Buscar palabra en archivo
4. Extraer usuarios y descripción
5. Buscar archivos con permisos específicos
6. Terminar
=====
Seleccione una opción (1-6): 4
Ejecutando programa de extracción de usuarios...
Extrayendo información de usuarios...
La información ha sido guardada en usuariosP_descripcion.txt
Contenido del archivo de salida:
=====
USUARIO | DESCRIPCIÓN
=====
Administrador | Cuenta integrada para la administración del equipo o dominio
DefaultAccount | Cuenta de usuario administrada por el sistema.
Invitado | Cuenta integrada para el acceso como invitado al equipo o dominio
usuario_admin | Administrador del Sistema
usuario_dba | Analista de Base de Datos
usuario_dev | Desarrollador Web
usuario_soporte | Soporte Técnico
usuario_test | Estudiante de Prueba
WDAGUtilityAccount | Una cuenta de usuario que el sistema administra y usa para escenarios de Protección de aplicaciones de Windows Defender.
Presione Enter para continuar...:
```

UNIX/SOLARIS SHELL SCRIPT

Primero, se guarda el archivo menu.sh y dentro de este, se agrega los comandos necesarios para cumplir con este objetivo.

```
#!/bin/sh
```

```
# Menu program for Solaris Unix Shell
```

```
# This script displays a menu and executes the selected previous exercises
```

```
while true; do
```

```
    # Clear the screen
```

```
    clear
```

```
    # Display the menu
```

```
    echo "=====
```

```
    echo "  MENU DE PROGRAMAS SHELL - SOLARIS"
```

```
    echo "=====
```

```
    echo "1. Hello World"
```

```
    echo "2. Número de líneas en /etc/profile"
```

```
    echo "3. Buscar palabra en archivo"
```

```
    echo "4. Extraer usuarios y descripción"
```

```
    echo "5. Buscar archivos con permisos específicos"
```

```
    echo "6. Terminar"
```

```
    echo "=====
```

```
    echo -n "Seleccione una opción (1-6): "
```

```
    # Read user input
```

```
    read option
```

```
    # Process the selected option using case statement
```

```
    case $option in
```

```
        1)
```

```
            echo "Ejecutando programa Hello World..."
```

```
# Suponiendo que el script se llama hello_world.sh
```

```
./hello.sh
```

```
read -p "Presione Enter para continuar..."
```

```
;;
```

2)

```
echo "Ejecutando programa de conteo de líneas..."
```

```
# Suponiendo que el script se llama contar_lineas.sh
```

```
./count_lines.sh
```

```
read -p "Presione Enter para continuar..."
```

```
;;
```

3)

```
echo "Ejecutando programa de búsqueda de palabras..."
```

```
echo -n "Ingrese la palabra a buscar: "
```

```
read palabra
```

```
echo -n "Ingrese el archivo donde buscar: "
```

```
read archivo
```

```
# Suponiendo que el script se llama buscar_palabra.sh
```

```
./buscar_palabra.sh "$palabra" "$archivo"
```

```
read -p "Presione Enter para continuar..."
```

```
;;
```

4)

```
echo "Ejecutando programa de extracción de usuarios..."
```

```
# Suponiendo que el script se llama extraer_usuarios.sh
```

```
./extraer_usuarios.sh
```

```
read -p "Presione Enter para continuar..."
```

```
;;
```

5)

```
echo "Ejecutando programa de búsqueda de archivos por permisos..."
```

```
echo -n "Ingrese el directorio donde buscar: "
```

```
read directorio
```

Shell Programming - Nombre: Andersson David Sánchez Méndez ACSO-2 – 1L

```
echo -n "Ingrese los permisos a buscar (formato -rw-r--r--): "
```

```
read permisos
```

```
# Suponiendo que el script se llama buscar_archivos.sh
```

```
./buscar_archivos.sh "$directorio" "$permisos"
```

```
read -p "Presione Enter para continuar..."
```

```
;;
```

6)

```
echo "Saliendo del programa. ¡Hasta luego!"
```

```
exit 0
```

```
;;
```

*)

```
echo "Opción inválida. Por favor, seleccione una opción válida (1-6)."
```

```
read -p "Presione Enter para continuar..."
```

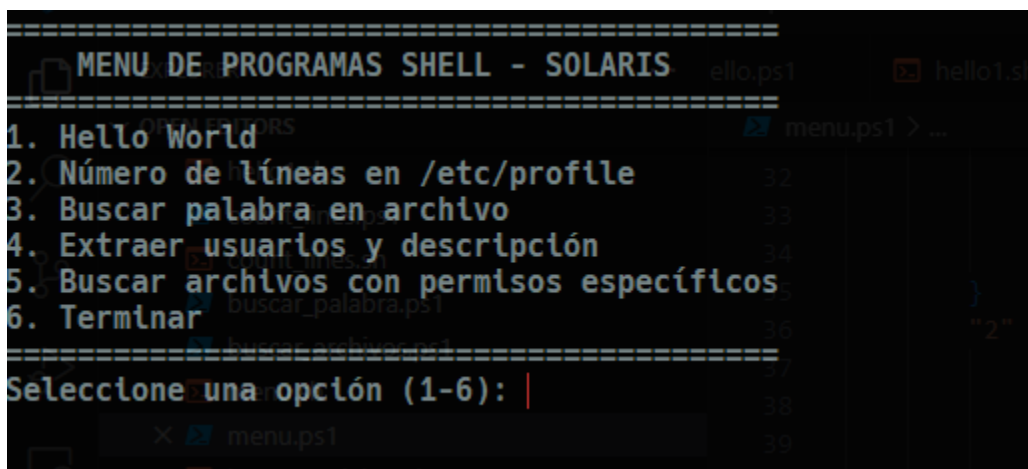
```
;;
```

```
esac
```

```
done
```

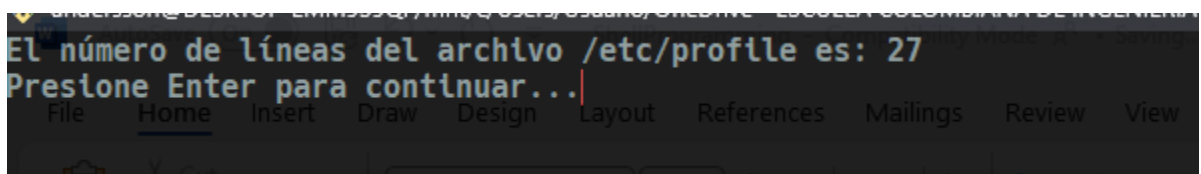
Se valida en Shellcheck:

Se hace ejecutable con **chmod +x menu.sh** y se ejecuta **./menu.sh**, generando el output correcto.



```
=====
MENU DE PROGRAMAS SHELL - SOLARIS
=====
1. Hello World
2. Número de líneas en /etc/profile
3. Buscar palabra en archivo
4. Extraer usuarios y descripción
5. Buscar archivos con permisos específicos
6. Terminar
=====
Seleccione una opción (1-6): |
```

Se prueba digamos la opción 2 y 4:



```
El número de líneas del archivo /etc/profile es: 27
Presione Enter para continuar...
```



```
=====
MENU DE PROGRAMAS SHELL - SOLARIS
=====
1. Hello World
2. Número de líneas en /etc/passwd
3. Buscar palabra en archivo
4. Extraer usuarios y descripción
5. Buscar archivos con permisos específicos
6. Terminar
=====
Seleccione una opción (1-6): 4
Ejecutando programa de extracción de usuarios...
Extrayendo nombres de usuario y descripciones...
La información ha sido guardada en usuariosS_descripcion.txt
Contenido del archivo de salida:
-----
=====
USUARIO | DESCRIPCIÓN
=====
root | root
daemon | daemon
bin | bin
sys | sys
sync | sync
games | games
man | man
lp | lp
mail | mail
news | news
uucp | uucp
proxy | proxy
www-data | www-data
backup | backup
list | Mailing List Manager
irc | ircd
nobody | nobody
systemd-network | systemd Network Management
systemd-timesync | systemd Time Synchronization
dhcpcd | DHCP Client Daemon,,,
systemd-resolve | systemd Resolver
polkitd | User for polkitd
andersson | ,,,
usuario_admin | Administrador del Sistema
usuario_dev | Desarrollador Web
usuario_dba | Analista de Base de Datos
usuario_soporte | Soporte Técnico
usuario_test | Estudiante de Prueba
Presione Enter para continuar...|
=====
```

Shell Programming - Nombre: Andersson David Sánchez Méndez ACSO-2 – 1L

7. Otros programas. Para cada una de las siguientes situaciones, escriba un programa Shell (y revíselo en ShellCheck) que las resuelva:

Lea de la línea de comandos un nombre y un directorio de búsqueda e indique si es un archivo, un subdirectorio u otra cosa en el directorio especificado.

Revisar si ha habido intentos de acceso ilegales al usuario root y mostrar fecha y hora de cada intento fallido y al final la cantidad total de ellos.

Debe estar tanto para Windows Core como Solaris.

POWERSHELL SCRIPT

Primero, se guarda el archivo **validar_tipos.ps1** y dentro de este, se agrega los comandos necesarios para cumplir con este objetivo, se hace lo mismo con **accesos_root.ps1** y se agregan los comandos, respectivamente.

```
# Script para verificar si un elemento es un archivo, un subdirectorio u otra cosa
```

```
# Uso: .\verificar_tipo.ps1 nombre_elemento directorio_búsqueda
```

```
# Verificar que se hayan proporcionado los dos parámetros
```

```
param(
```

```
    [Parameter(Mandatory=$true, Position=0)]
```

```
    [string]$Nombre,
```

```
    [Parameter(Mandatory=$true, Position=1)]
```

```
    [string]$Directorio
```

```
)
```

```
# Limpiar pantalla
```

```
Clear-Host
```

```
# Verificar si el directorio existe
```

```
if (-not (Test-Path -Path $Directorio -PathType Container)) {
```

```
    Write-Host "Error: El directorio '$Directorio' no existe." -ForegroundColor Red
```

```
    exit 1
```

```
}
```

```
# Construir la ruta completa del elemento
```

```
$RutaCompleta = Join-Path -Path $Directorio -ChildPath $Nombre
```

```
# Verificar el tipo de elemento
```

```
if (Test-Path -Path $RutaCompleta -PathType Leaf) {
```

```
    Write-Host "$Nombre' es un archivo en el directorio '$Directorio'." -ForegroundColor Green
```

```
}
```

```
elseif (Test-Path -Path $RutaCompleta -PathType Container) {
```

```
    Write-Host "$Nombre' es un subdirectorio dentro de '$Directorio'." -ForegroundColor Green
```

```
}
```

```
elseif (Test-Path -Path $RutaCompleta) {
```

```
    Write-Host "$Nombre' existe en '$Directorio' pero no es un archivo ni un subdirectorio (podría ser un enlace simbólico, etc.)." -ForegroundColor Yellow
```

```
}
```

```
else {
```

```
    Write-Host "$Nombre' no existe en el directorio '$Directorio'." -ForegroundColor Red
```

```
}
```

```
exit 0
```

```
# Script para revisar intentos fallidos de acceso al usuario administrador
```

```
# Este script busca eventos de seguridad relacionados con intentos fallidos de inicio de sesión
```

```
# Limpiar pantalla
```

```
Clear-Host
```

```
Write-Host "=====
```

```
Write-Host "  INTENTOS FALLIDOS DE ACCESO AL ADMINISTRADOR"
```

Shell Programming - Nombre: Andersson David Sánchez Méndez ACSO-2 – 1L

```
Write-Host "===== "

# Definir la cantidad de días hacia atrás para buscar (por defecto, 30 días)

$DiasAtras = 30

$FechaInicio = (Get-Date).AddDays(-$DiasAtras)

Write-Host "Buscando intentos fallidos en los últimos $DiasAtras días..." -ForegroundColor Yellow

# Intentar obtener eventos del registro de seguridad

# EventID 4625 = Fallo de inicio de sesión

try {

    $Events = Get-WinEvent -FilterHashTable @{

        LogName = 'Security'

        ID = 4625

        StartTime = $FechaInicio

    } -ErrorAction Stop | Where-Object {

        $_.Properties[5].Value -eq "Administrator" -or

        $_.Properties[5].Value -eq "Administrador" -or

        $_.Properties[1].Value -eq "S-1-5-21-*-500"

    }

    if ($Events.Count -gt 0) {

        Write-Host "Se encontraron $($Events.Count) intentos fallidos de acceso para el administrador:"

        -ForegroundColor Cyan

        # Mostrar cada intento con fecha y hora

        foreach ($Event in $Events) {

            $FechaHora = $Event.TimeCreated.ToString("yyyy-MM-dd HH:mm:ss")

            $DireccionIP = $Event.Properties[19].Value

            $Razon = $Event.Properties[8].Value
```

Shell Programming - Nombre: Andersson David Sánchez Méndez ACSO-2 – 1L

```
Write-Host "Fecha/Hora: $FechaHora - IP: $DireccionIP - Razón: $Razon" -ForegroundColor
White

}

Write-Host "=====

Write-Host "Total de intentos fallidos de acceso para el administrador: $($Events.Count)" -
ForegroundColor Green

Write-Host "=====

}

else {

    Write-Host "No se encontraron intentos fallidos de acceso para el administrador en los últimos
$DiasAtras días." -ForegroundColor Green

}

}

catch {

    Write-Host "Error al buscar en el registro de eventos: $_" -ForegroundColor Red

    Write-Host "Es posible que necesite ejecutar PowerShell como administrador para acceder al
registro de seguridad." -ForegroundColor Yellow

# Intentar buscar en el registro de aplicaciones como alternativa

try {

    Write-Host "Intentando buscar en otros registros..." -ForegroundColor Yellow

    $Events = Get-WinEvent -FilterHashTable @{

        LogName = 'Application'

        StartTime = $FechaInicio

    } -ErrorAction Stop | Where-Object { $_.Message -match "administrador|administrador" -and
$_ .Message -match "fail|fallo" }

    if ($Events.Count -gt 0) {

        Write-Host "Se encontraron $($Events.Count) posibles referencias a intentos fallidos de
acceso:" -ForegroundColor Cyan

        foreach ($Event in $Events) {
```

Shell Programming - Nombre: Andersson David Sánchez Méndez ACSO-2 – 1L

```
Write-Host "Fecha/Hora: $($Event.TimeCreated) - ID: $($Event.Id) - Mensaje:
$($Event.Message.Substring(0, [Math]::Min(100, $Event.Message.Length)))..." -ForegroundColor
White

    }

}

else {

    Write-Host "No se encontraron referencias a intentos fallidos en otros registros." -
ForegroundColor Yellow

    }

}

catch {

    Write-Host "No se pudo acceder a los registros alternativos: $_" -ForegroundColor Red

}

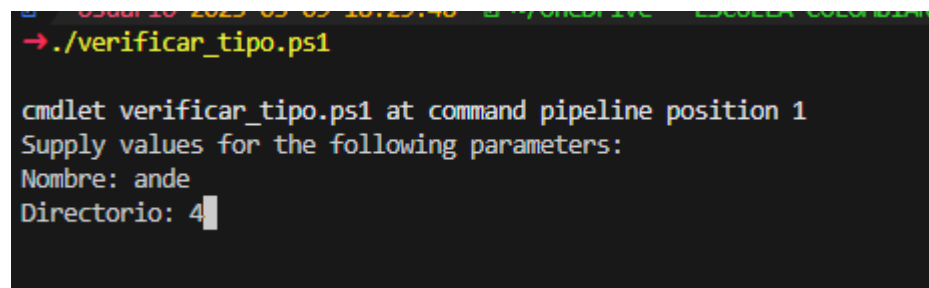
}

exit 0
```

Se valida en Shellcheck:

Asegurarse de la política de ejecución para ejecutar scripts: **Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser.**

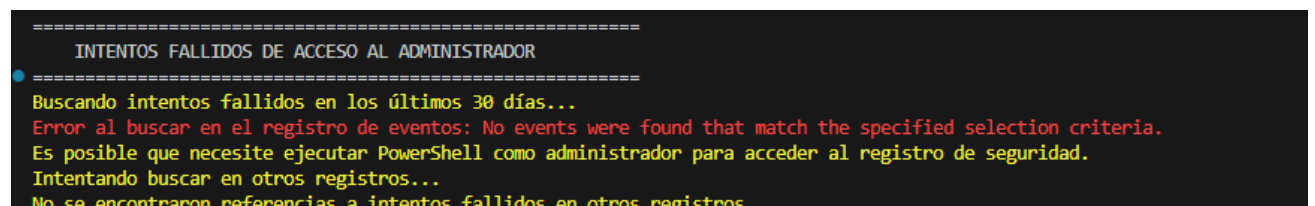
Se ejecuta con **./verificar_tipo.ps1** o **./accesos_root.ps1** generando el output correcto.



```
PS C:\Users\ande> ./verificar_tipo.ps1

cmdlet verificar_tipo.ps1 at command pipeline position 1
Supply values for the following parameters:
Nombre: ande
Directorio: 4
```

Error: El directorio '4' no existe.



```
=====
INTENTOS FALLIDOS DE ACCESO AL ADMINISTRADOR
=====
Buscando intentos fallidos en los últimos 30 días...
Error al buscar en el registro de eventos: No events were found that match the specified selection criteria.
Es posible que necesite ejecutar PowerShell como administrador para acceder al registro de seguridad.
Intentando buscar en otros registros...
No se encontraron referencias a intentos fallidos en otros registros.
```

UNIX/SOLARIS SHELL SCRIPT

Primero, se guarda el archivo **verificar_tipo.sh** y dentro de este, se agrega los comandos necesarios para cumplir con este objetivo, se hace lo mismo con **accesos_root.sh** y se agregan los comandos, respectivamente.

```
#!/bin/bash

# Script para verificar si un elemento es un archivo, un subdirectorio u otra cosa
# Uso: ./verificar_tipo.sh nombre_elemento directorio_busqueda

# Verificar que se hayan proporcionado los dos parámetros
if [ $# -ne 2 ]; then
    echo "Error: Debe proporcionar dos parámetros."
    echo "Uso: $0 nombre_elemento directorio_busqueda"
    exit 1
fi

nombre="$1"
directorio="$2"

# Verificar si el directorio existe
if [ ! -d "$directorio" ]; then
    echo "Error: El directorio '$directorio' no existe."
    exit 1
fi

# Construir la ruta completa del elemento
ruta_completa="$directorio/$nombre"

# Verificar el tipo de elemento
if [ -f "$ruta_completa" ]; then
    echo "'$nombre' es un archivo en el directorio '$directorio'."
elif [ -d "$ruta_completa" ]; then
    echo "'$nombre' es un subdirectorio dentro de '$directorio'."
elif [ -e "$ruta_completa" ]; then
    echo "'$nombre' existe en '$directorio' pero no es un archivo ni un subdirectorio (podría ser un enlace simbólico, socket, etc.)."
else
```

Shell Programming - Nombre: Andersson David Sánchez Méndez ACSO-2 – 1L

```
    echo "'$nombre' no existe en el directorio '$directorio'."
fi

exit 0

#!/bin/bash

# Script para revisar intentos fallidos de acceso al usuario root
# Este script busca en los logs de autenticación y muestra fecha y hora de cada intento

# Limpiar pantalla
clear

echo "=====
echo "  INTENTOS FALLIDOS DE ACCESO AL USUARIO ROOT"
echo "=====

# En sistemas Solaris, los logs pueden estar en diferentes ubicaciones
# Verificamos las ubicaciones más comunes
log_files=("/var/log/auth.log" "/var/adm/messages" "/var/log/syslog")
contador=0

for log_file in "${log_files[@]}; do
    if [ -f "$log_file" ]; then
        echo "Buscando en $log_file..."

        # Buscar patrones típicos de intentos fallidos de acceso para root
        # El comando awk se utiliza para extraer fecha y hora junto con el mensaje
        intentos=$(grep -i "Failed password" "$log_file" | grep -i "root" | awk '{print $1,$2,$3,$0}')

        if [ -n "$intentos" ]; then
            # Mostrar cada intento con fecha y hora
            echo "FECHA Y HORA          MENSAJE"
            echo "-----"
            echo "$intentos" | while read -r linea; do
```


Shell Programming - Nombre: Andersson David Sánchez Méndez ACSO-2 – 1L

```
# Extraer fecha y hora (generalmente están en los primeros campos)
fecha_hora=$(echo "$linea" | awk '{print $1" "$2" "$3}')
mensaje=$(echo "$linea")
echo "[$fecha_hora] - $mensaje"
done

# Contar el número de intentos encontrados
num_intentos=$(echo "$intentos" | wc -l)
contador=$((contador + num_intentos))
fi
fi
done

# Si no se encontraron logs en las ubicaciones estándar, buscar en /var/log
if [ $contador -eq 0 ]; then
    echo "Buscando en archivos de log adicionales..."

    # Buscar en todos los archivos de log
    for log_file in /var/log/*; do
        if [ -f "$log_file" ]; then
            intentos=$(grep -i "Failed password" "$log_file" | grep -i "root" | awk '{print $1,$2,$3,$0}')

            if [ -n "$intentos" ]; then
                echo "En $log_file:"
                echo "FECHA Y HORA          MENSAJE"
                echo "-----"
                echo "$intentos" | while read -r linea; do
                    # Extraer fecha y hora
                    fecha_hora=$(echo "$linea" | awk '{print $1" "$2" "$3}')
                    mensaje=$(echo "$linea")
                    echo "[$fecha_hora] - $mensaje"
                done

                # Contar el número de intentos encontrados
                num_intentos=$(echo "$intentos" | wc -l)
                contador=$((contador + num_intentos))
            fi
        fi
    done
fi
```

Shell Programming - Nombre: Andersson David Sánchez Méndez ACSO-2 – 1L

```
fi

fi

done

fi

echo "=====
echo "Total de intentos fallidos de acceso para root: $contador"
echo "=====

exit 0
```

Se valida en Shellcheck:

Se hace ejecutable con `chmod +x verificar_tipo.sh accesos_root.sh` y se ejecuta con `./verificar_tipo.sh archivo.txt /mnt/c/Users/Usuario` o `./accesos_root.sh`, generando el output correcto.

```
andersson@DESKTOP-EMM939Q: /mnt/c/Users/Usuario/OneDrive - ESCUELA COLOMBIANA DE INGENIERIA JULIO GARAVITO/Desktop/6. SEMESTER/ACSO/SECOND TERM/platforms/Shell WS-SOL/shells ./verificar_tipo.sh archivo.txt /mnt/c/
Users/Usuario
archivo.txt: no existe en el directorio '/mnt/c/Users/Usuario'.
```

```
andersson@DESKTOP-EMM939Q: /mnt/c/Users/Usuario/OneDrive - ESCUELA COLOMBIANA DE INGENIERIA JULIO GARAVITO/Desktop/6. SEMESTER/ACSO/SECOND TERM/platforms/Shell WS-SOL/shells ./verificar_tipo.sh archivo.txt /mnt/c/
Users/Usuario
archivo.txt: no existe en el directorio '/mnt/c/Users/Usuario'.
```

```
=====
INTENTOS FALLIDOS DE ACCESO AL USUARIO ROOT
=====
Buscando en /var/log/auth.log...
Buscando en /var/log/syslog...
Buscando en archivos de log adicionales...
grep: /var/log/btmp: Permission denied
=====
Total de intentos fallidos de acceso para root: 0
=====
andersson@DESKTOP-EMM939Q: /mnt/c/Users/Usuario/OneDrive - ESCUELA COLOMBIANA DE INGENIERIA JULIO GARAVITO/Desktop/6. SEMESTER/ACSO/SECOND TERM/platforms/Shell WS-SOL/shells ./verificar_tipo.sh archivo.txt /mnt/c/
Users/Usuario
archivo.txt: no existe en el directorio '/mnt/c/Users/Usuario'.
```

2. El directorio donde quieres buscar ese elemento

Para ejecutar el script correctamente, debes incluir estos

Por ejemplo:

8. Con base al menú creado anteriormente, crear otro archivo donde se agreguen los dos últimos archivos, **verificar_tipo.sh/ps1** y **accesos/root.sh/ps1** como dos nuevas opciones.

POWERSHELL SCRIPT

```
# Menu program for Windows PowerShell
```

```
# This script displays a menu and executes the selected previous exercises
```

```
function Show-Menu {
```

```
    Clear-Host
```

```
    Write-Host "====="
```

```
    Write-Host "  MENU DE PROGRAMAS SHELL - WINDOWS"
```

```
    Write-Host "====="
```

```
    Write-Host "1. Hello World"
```

```
    Write-Host "2. Número de líneas en C:\Windows\System32\drivers\etc\hosts"
```

```
    Write-Host "3. Buscar palabra en archivo"
```

```
    Write-Host "4. Extraer usuarios y descripción"
```

```
    Write-Host "5. Buscar archivos con permisos específicos"
```

```
    Write-Host "6. Verificar tipo de archivo"
```

```
    Write-Host "7. Mostrar accesos administrativos"
```

```
    Write-Host "8. Terminar"
```

```
    Write-Host "====="
```

```
    Write-Host "Seleccione una opción (1-8): " -NoNewline
```

```
}
```

```
# Main program loop
```

```
$running = $true
```

```
while ($running) {
```

```
    # Display the menu
```

```
    Show-Menu
```

```
    # Read user input
```

```
    $option = Read-Host
```

```
    # Process the selected option using switch statement (PowerShell's equivalent to case)
```

```
    switch ($option) {
```

```
"1" {  
    Write-Host "Ejecutando programa Hello World..."  
    # Suponiendo que el script se llama hello_world.ps1  
    & ".\hello.ps1"  
    Read-Host "Presione Enter para continuar..."  
}  
"2" {  
    Write-Host "Ejecutando programa de conteo de líneas..."  
    # Suponiendo que el script se llama contar_lineas.ps1  
    & ".\count_lines.ps1"  
    Read-Host "Presione Enter para continuar..."  
}  
"3" {  
    Write-Host "Ejecutando programa de búsqueda de palabras..."  
    $palabra = Read-Host "Ingrese la palabra a buscar"  
    $archivo = Read-Host "Ingrese el archivo donde buscar"  
    # Suponiendo que el script se llama buscar_palabra.ps1  
    & ".\buscar_palabra.ps1" $palabra $archivo  
    Read-Host "Presione Enter para continuar..."  
}  
"4" {  
    Write-Host "Ejecutando programa de extracción de usuarios..."  
    # Suponiendo que el script se llama extraer_usuarios.ps1  
    & ".\extraer_usuarios.ps1"  
    Read-Host "Presione Enter para continuar..."  
}  
"5" {  
    Write-Host "Ejecutando programa de búsqueda de archivos por permisos..."  
    $directorio = Read-Host "Ingrese el directorio donde buscar"  
    $permisos = Read-Host "Ingrese los permisos a buscar (ejemplo: FullControl)"  
    # Suponiendo que el script se llama buscar_archivos.ps1  
    & ".\buscar_archivos.ps1" $directorio $permisos  
    Read-Host "Presione Enter para continuar..."  
}  
"6" {  
    Write-Host "Ejecutando programa para verificar tipo de archivo..."
```

Shell Programming - Nombre: Andersson David Sánchez Méndez ACSO-2 – 1L

```
$rutaArchivo = Read-Host "Ingrese la ruta del archivo a verificar"

# Suponiendo que el script se llama verificar_tipo.ps1
& ".\verificar_tipo.ps1" $rutaArchivo

Read-Host "Presione Enter para continuar..."
}

"7" {

    Write-Host "Ejecutando programa para mostrar accesos administrativos..."

    # Suponiendo que el script se llama accesos_root.ps1
    & ".\accesos_root.ps1"

    Read-Host "Presione Enter para continuar..."
}

"8" {

    Write-Host "Saliendo del programa. ¡Hasta luego!"

    $running = $false
}

default {

    Write-Host "Opción inválida. Por favor, seleccione una opción válida (1-8)."

    Read-Host "Presione Enter para continuar..."
}

}

}
```

Asegurarse de la política de ejecución para ejecutar scripts: **Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser**.

Se ejecuta con **./menú_completo.ps1**, generando el output correcto.

```
=====
MENU DE PROGRAMAS SHELL - WINDOWS
=====
1. Hello World
2. Número de líneas en C:\Windows\System32\drivers\etc\hosts
3. Buscar palabra en archivo
4. Extraer usuarios y descripción
5. Buscar archivos con permisos específicos
6. Verificar tipo de archivo
7. Mostrar accesos administrativos
8. Terminar
=====
Seleccione una opción (1-8):
```

Opción 7

```
=====
INTENTOS FALLIDOS DE ACCESO AL ADMINISTRADOR
=====
Buscando intentos fallidos en los últimos 30 días...
Error al buscar en el registro de eventos: No events were found that match the specified selection criteria.
Es posible que necesite ejecutar PowerShell como administrador para acceder al registro de seguridad.
Intentando buscar en otros registros...
No se encontraron referencias a intentos fallidos en otros registros.
Presione Enter para continuar...
```

UNIX/SOLARIS SHELL SCRIPT

```
#!/bin/bash

# Menu program for Solaris Unix Shell

# This script displays a menu and executes the selected previous exercises

while true; do

    # Clear the screen

    clear

    # Display the menu

    echo "=====
MENU DE PROGRAMAS SHELL - SOLARIS"
echo "====="
```

Shell Programming - Nombre: Andersson David Sánchez Méndez ACSO-2 – 1L

```
echo "1. Hello World"
echo "2. Número de líneas en /etc/profile"
echo "3. Buscar palabra en archivo"
echo "4. Extraer usuarios y descripción"
echo "5. Buscar archivos con permisos específicos"
echo "6. Verificar tipo de archivo"
echo "7. Mostrar accesos root"
echo "8. Terminar"
echo "====="
echo -n "Seleccione una opción (1-8): "

# Read user input
read option

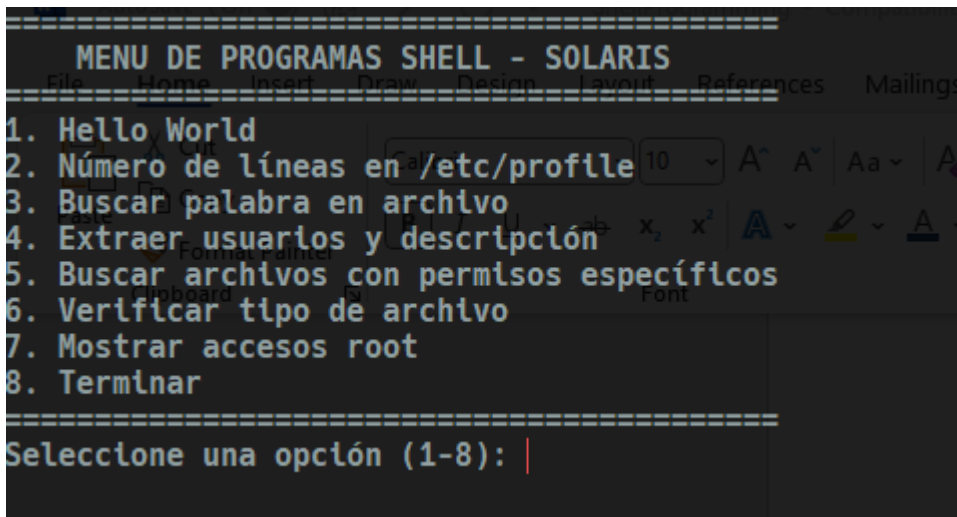
# Process the selected option using case statement
case $option in
    1)
        echo "Ejecutando programa Hello World..."
        # Suponiendo que el script se llama hello_world.sh
        ./hello.sh
        read -p "Presione Enter para continuar..."
        ;;
    2)
        echo "Ejecutando programa de conteo de líneas..."
        # Suponiendo que el script se llama contar_lineas.sh
        ./count_lines.sh
        read -p "Presione Enter para continuar..."
        ;;
    3)
        echo "Ejecutando programa de búsqueda de palabras..."
        echo -n "Ingrese la palabra a buscar: "
        read palabra
        echo -n "Ingrese el archivo donde buscar: "
        read archivo
        # Suponiendo que el script se llama buscar_palabra.sh
        ./buscar_palabra.sh "$palabra" "$archivo"
```

```
    read -p "Presione Enter para continuar..."
;;
4)
    echo "Ejecutando programa de extracción de usuarios..."
    # Suponiendo que el script se llama extraer_usuarios.sh
    ./extraer_usuarios.sh
    read -p "Presione Enter para continuar..."
    ;;
5)
    echo "Ejecutando programa de búsqueda de archivos por permisos..."
    echo -n "Ingrese el directorio donde buscar: "
    read directorio
    echo -n "Ingrese los permisos a buscar (formato -rw-r--r--): "
    read permisos
    # Suponiendo que el script se llama buscar_archivos.sh
    ./buscar_archivos.sh "$directorio" "$permisos"
    read -p "Presione Enter para continuar..."
    ;;
6)
    echo "Ejecutando programa para verificar tipo de archivo..."
    echo -n "Ingrese la ruta del archivo a verificar: "
    read ruta_archivo
    # Suponiendo que el script se llama verificar_tipo.sh
    ./verificar_tipo.sh "$ruta_archivo"
    read -p "Presione Enter para continuar..."
    ;;
7)
    echo "Ejecutando programa para mostrar accesos root..."
    # Suponiendo que el script se llama accesos_root.sh
    ./accesos_root.sh
    read -p "Presione Enter para continuar..."
    ;;
8)
    echo "Saliendo del programa. ¡Hasta luego!"
    exit 0
;;
```



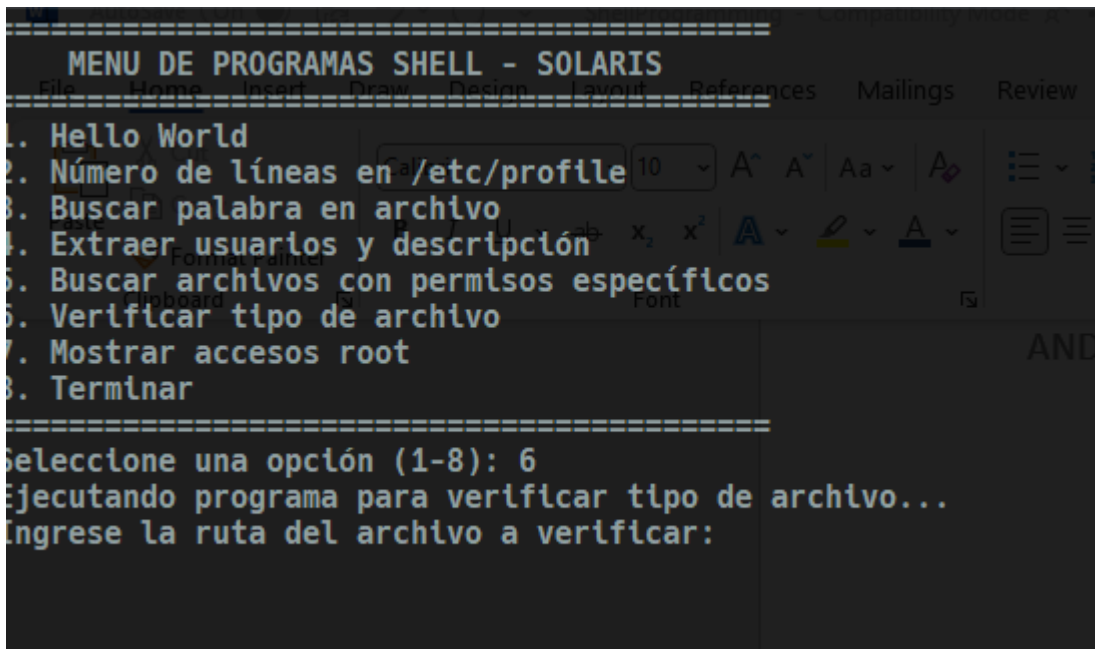
```
*)  
  
    echo "Opción inválida. Por favor, seleccione una opción válida (1-8)."  
  
    read -p "Presione Enter para continuar..."  
  
    ;;  
  
esac  
  
done
```

Se hace ejecutable con **chmod +x menu_completo.sh** y se ejecuta con **./menu_completo.sh**, generando el output correcto.



```
=====
MENU DE PROGRAMAS SHELL - SOLARIS
=====
1. Hello World
2. Número de líneas en /etc/profile
3. Buscar palabra en archivo
4. Extraer usuarios y descripción
5. Buscar archivos con permisos específicos
6. Verificar tipo de archivo
7. Mostrar accesos root
8. Terminar
=====
Seleccione una opción (1-8): |
```

Opción 6



```
=====
MENU DE PROGRAMAS SHELL - SOLARIS
=====
1. Hello World
2. Número de líneas en /etc/profile
3. Buscar palabra en archivo
4. Extraer usuarios y descripción
5. Buscar archivos con permisos específicos
6. Verificar tipo de archivo
7. Mostrar accesos root
8. Terminar
=====
Seleccione una opción (1-8): 6
Ejecutando programa para verificar tipo de archivo...
Ingrese la ruta del archivo a verificar:
```

BIBLIOGRAFÍA

- <https://github.com/PowerShell/PowerShell>
- <https://learn.microsoft.com/es-es/host-integration-server/core/powershell-module-cmdlets-and-commands>
- <https://www.geeksforgeeks.org/essential-linuxunix-commands/>
- <https://www.ibm.com/docs/en/zos/2.4.0?topic=reference-summary-zos-unix-shell-commands>