**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 b´asicos en Unix Shell
3. Aprender a escribir programas b´asicos en Windows PowerShell
4. Escriba un programa Shell que:

Limpie la pantalla

Imprima el mensaje “Hello World from Shell”.

Conéctese al ambiente ShellCheck (h[ttps://www.shellcheck.net/)](http://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.



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**:**

**A screen shot of a computer

AI-generated content may be incorrect.**

**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"

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.

**A screenshot of a computer

AI-generated content may be incorrect.**

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**:**

A black screen with white text

AI-generated content may be incorrect.

**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

# Print the message

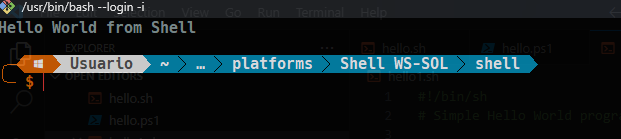
echo "Hello World from Shell"

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

A screenshot of a computer

AI-generated content may be incorrect.

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



1. Ejecuci´on autom´atica de una secuencia de comandos Escriba un programa Shell (y revíselo en ShellCheck) que:

Limpie la pantalla

Imprima el mensaje “El nu´mero de líneas del archivo /etc/profile es:” y el nu´mero 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

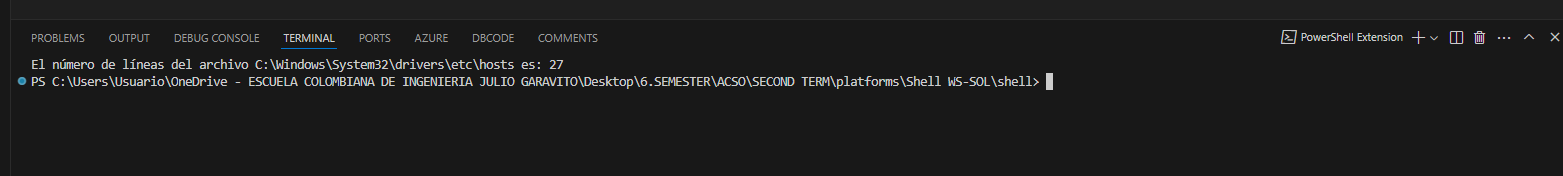
Write-Host "El número de líneas del archivo $filePath es: $lineCount"

Ahora se valida con Shellcheck:

A screenshot of a computer

AI-generated content may be incorrect.

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:

**A screenshot of a computer

AI-generated content may be incorrect.**

Se hace ejecutable con **chmod +x count\_lines.sh** y se ejecuta de la misma manera **./count\_lines.sh**, generando el output correcto.

**A screenshot of a computer

AI-generated content may be incorrect.**

1. 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 ejecuci´on sería del estilo *buscar palabra.sh palabra buscada archivo de bu´squeda*, ejemplo

*buscar palabra.sh casa /etc/passwd*

Imprima el resultado de la bu´squeda.

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

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\_busqueda>"

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 {

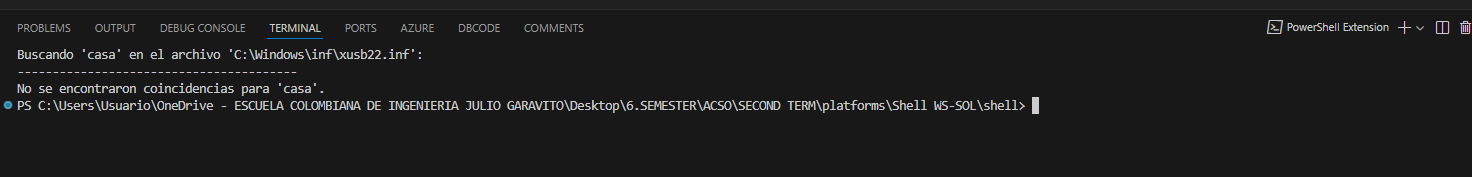
Write-Host "No se encontraron coincidencias para '$palabra'."

}

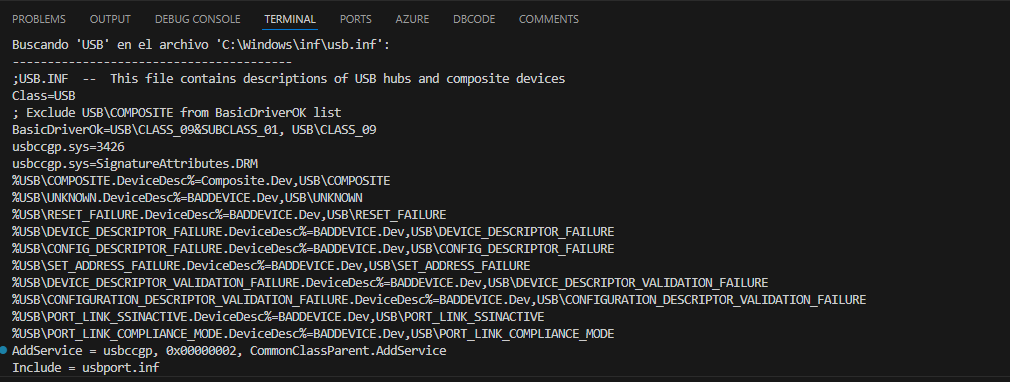
Ahora se valida con Shellcheck:

****

Se ejecuta con **.\buscar\_palabra.ps1 casa C:\Windows\inf\xusb22.inf**, generando el output correcto.

****

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**

****

**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\_busqueda>"

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")

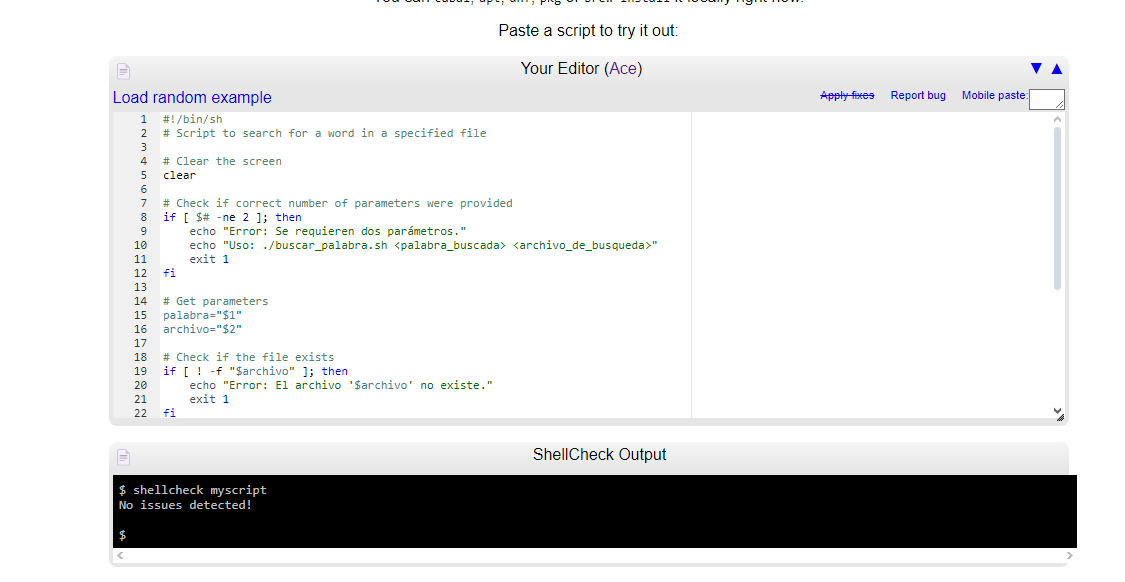
echo "Se encontraron $coincidencias coincidencias."

else

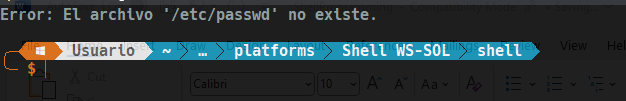
echo "No se encontraron coincidencias para '$palabra'."

Fi

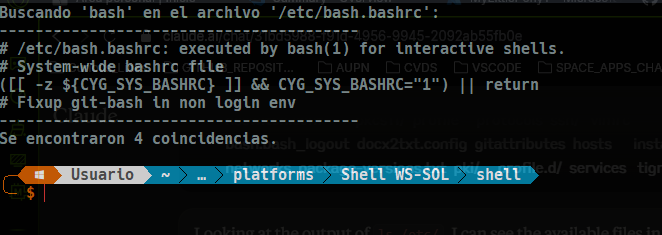
Ahora se valida con Shellcheck:



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



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.**



1. Uso de repeticiones y almacenamiento de la respuesta de ejecuci´on de un comando en un archivo

Cree 5 usuarios en su sistema. Incluya descripci´on 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 descripci´on de los mismos
  + Deje el resultado de la ejecuci´on 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"

)

# 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

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

# 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)

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Y se mira el archivo generado .txt



A black screen with white text

AI-generated content may be incorrect.

**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

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"

# 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:"

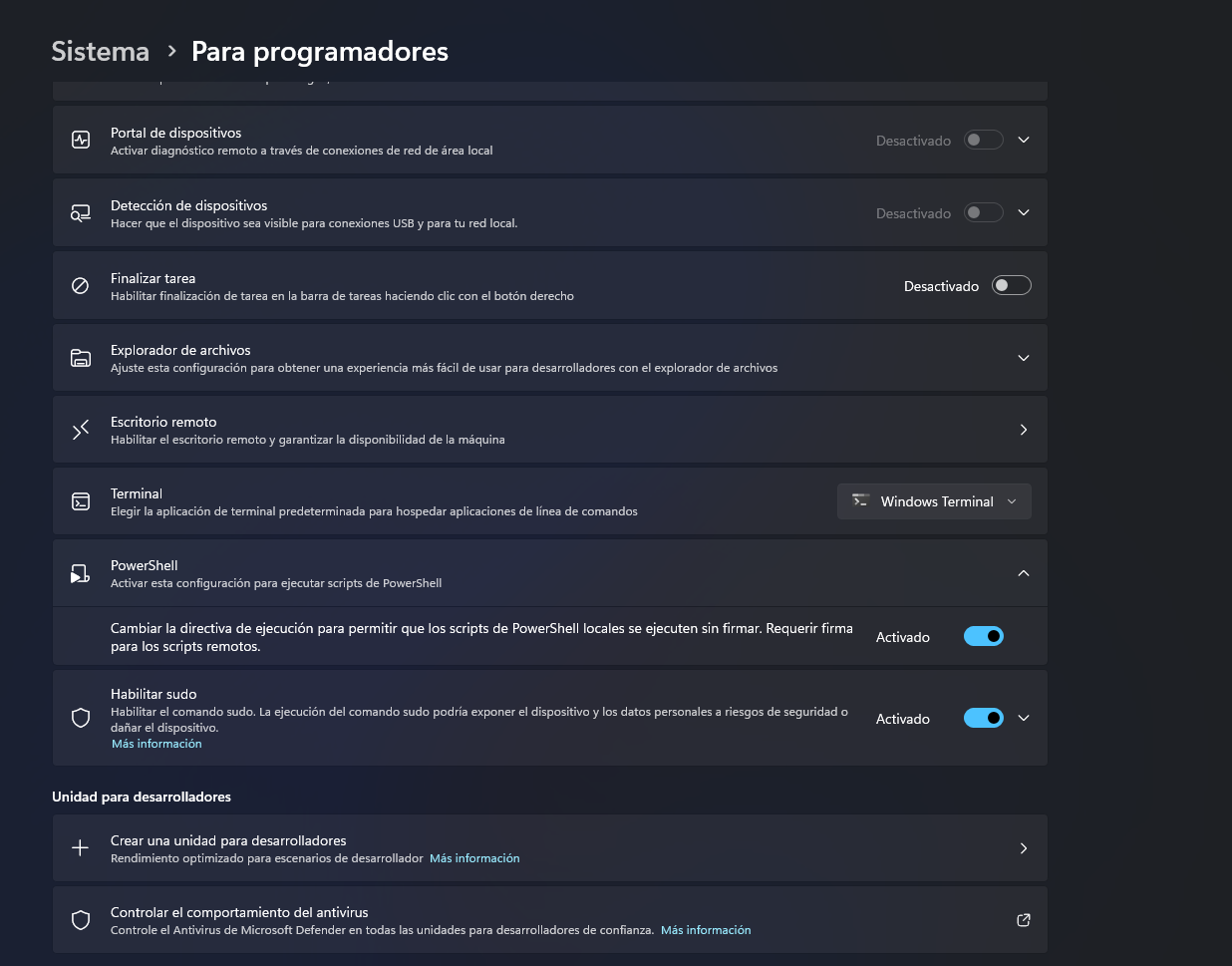
echo "--------------------------------"

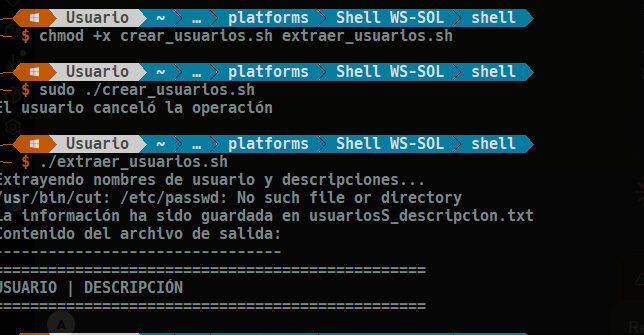
/usr/bin/cat "$ARCHIVO\_SALIDA"

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.



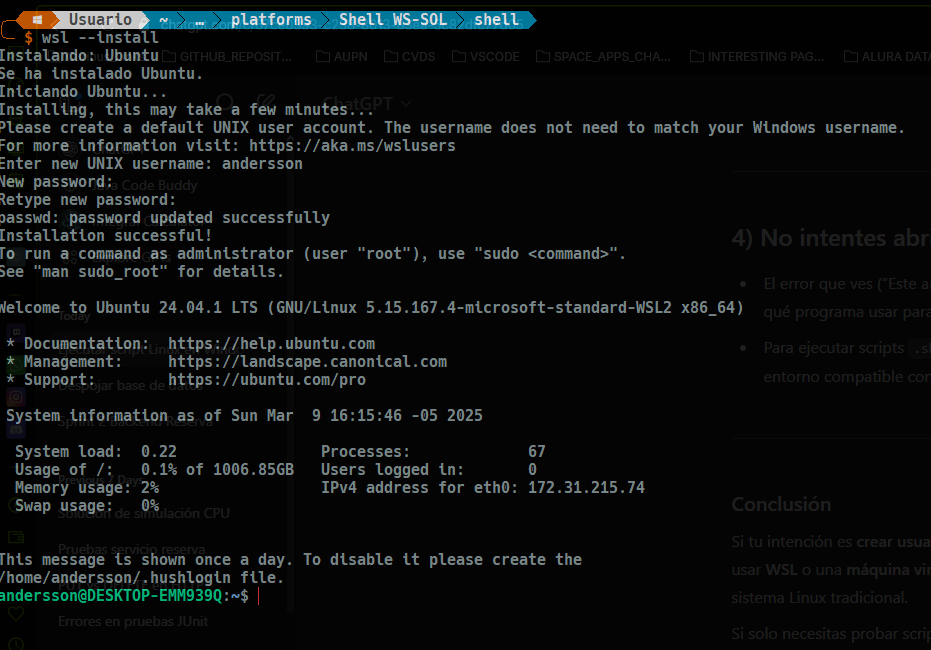


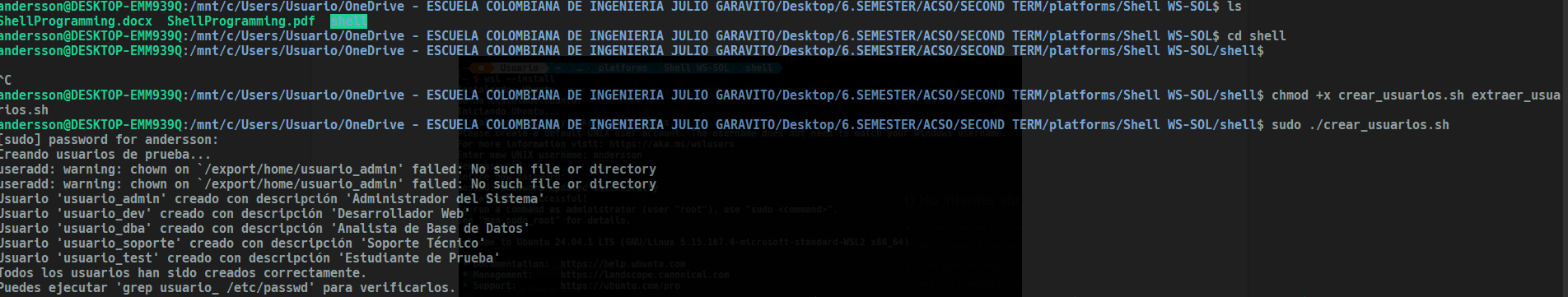


A black background with white lines

AI-generated content may be incorrect.

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.





A screenshot of a computer

AI-generated content may be incorrect.

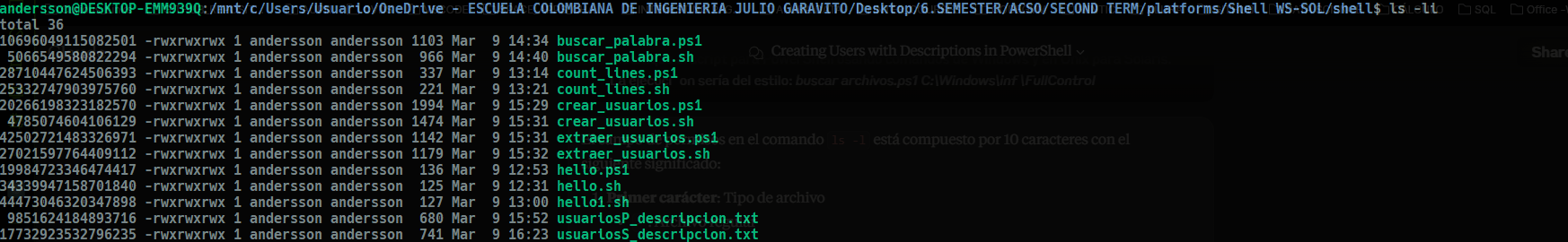
Y en el .txt:

A screenshot of a computer program

AI-generated content may be incorrect.

1. Manejo de condicionales

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



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 ejecuci´on 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 ejecuci´on 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.

A black screen with white text

AI-generated content may be incorrect.

**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"

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

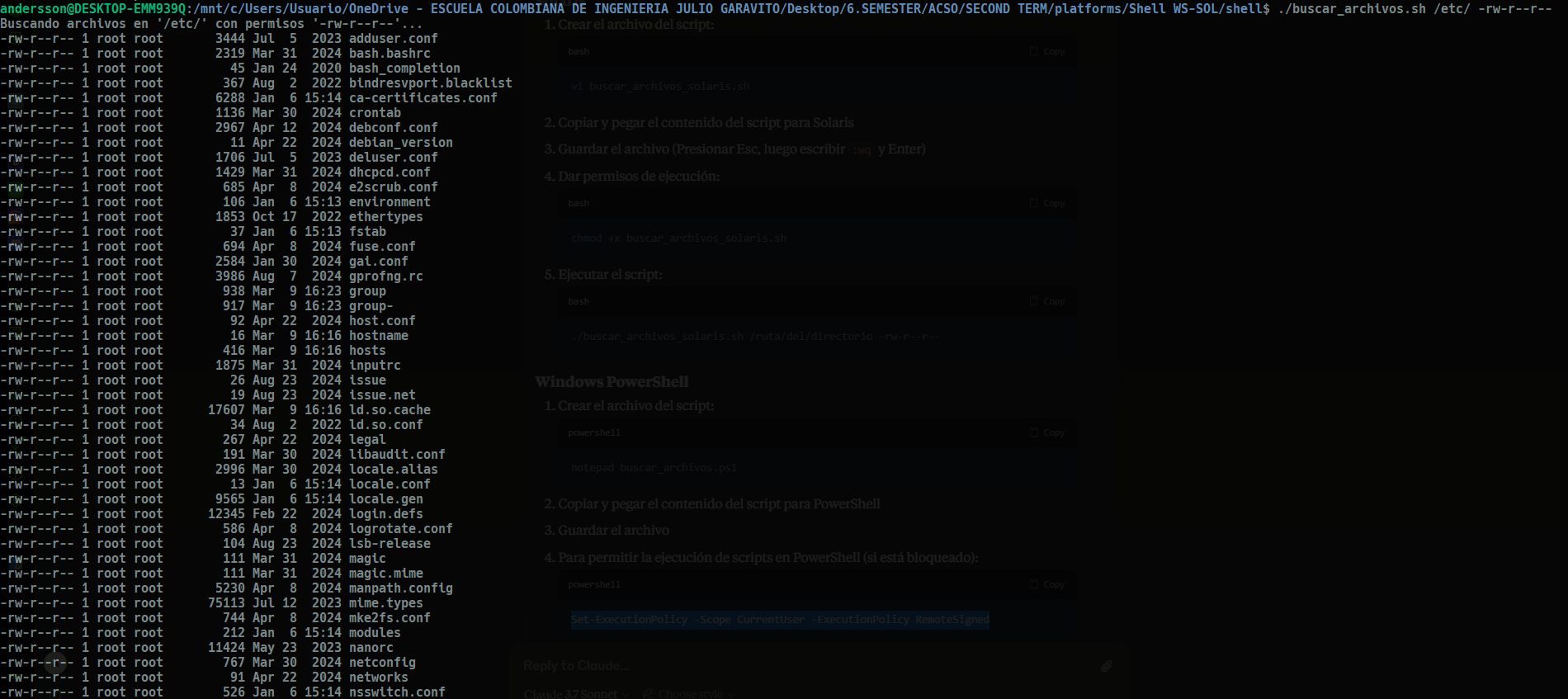
echo "$line"

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.

****

1. Menu´ en Shell usando la instrucci´on case.

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

Realice un menu´ usando Shell que permita escoger la ejecuci´on de uno de los Shell anteriores todas las veces que se quiera

Adicionalmente incluya una opci´on 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..."

}

"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:

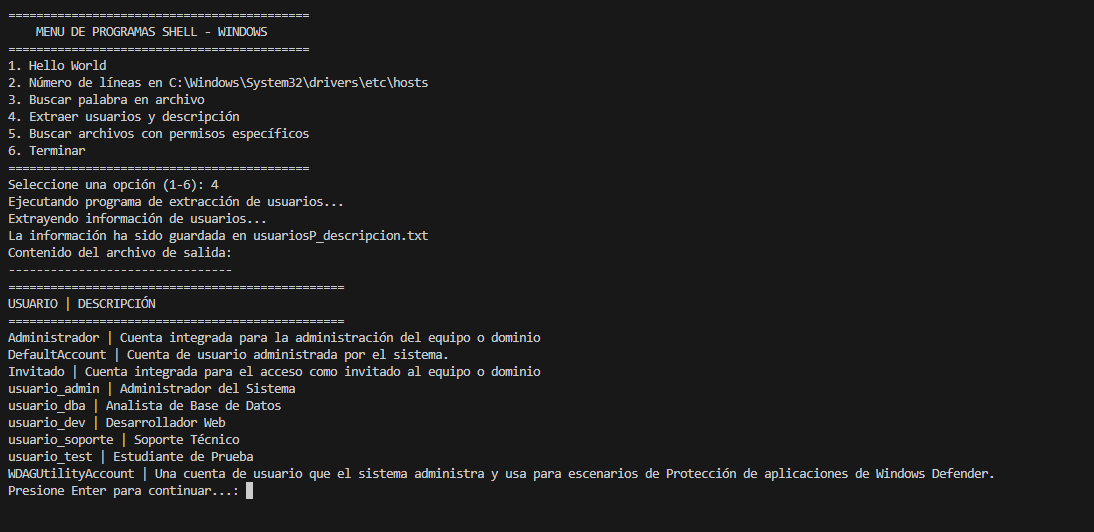
Para permitir la ejecución de scripts en pwsh si está bloqueado se usa: **Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser.**

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

A black screen with white text

AI-generated content may be incorrect.

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



**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

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..."

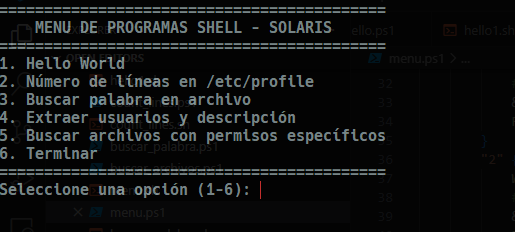
;;

esac

done

Se valida en Shellcheck:

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



Se prueba digamos la opción 2 y 4:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. 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 bu´squeda 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\_busqueda

# 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"

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

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 "administrator|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) {

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.

A screenshot of a computer

AI-generated content may be incorrect.



A screenshot of a computer error

AI-generated content may be incorrect.

**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

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

# 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

echo "=========================================================="

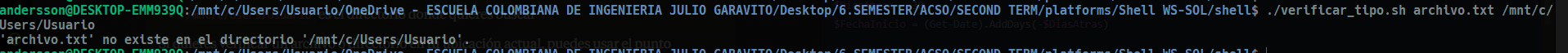
echo "Total de intentos fallidos de acceso para root: $contador"

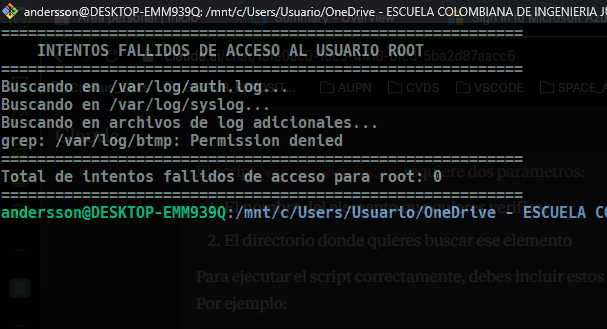
echo "=========================================================="

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.





1. 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..."

$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.

A screenshot of a computer program

AI-generated content may be incorrect.

Opción 7

A screenshot of a computer program

AI-generated content may be incorrect.

**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 "==========================================="

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. 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.

A screenshot of a computer program

AI-generated content may be incorrect.

Opción 6

A screenshot of a computer

AI-generated content may be incorrect.

**BIBLIOGRAFÍA**

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