Universidad de Guadalajara

Centro Universitario De Ciencias Exactas E Ingenieritas CUCEI

Ingeniería Informática

Actividad 14 – Scripts para optimizar

Juan Antonio Ramírez Aguilar

Código: 212482507

Mtra. Becerra Velázquez Violeta del Rocío

Seminario de Solución de Problemas de Uso, Adaptación, Explotación de Sistemas Operativos

Indicé

I. Introducción 1

II. Scripts 1

Lenguajes de programación para Scripts 2

Ejemplos de Scripts 3

III. Infografía de los Scripts 4

Link 4

IV. Elaboración de un BATCH 5

V. Links del Proyecto 8

VI. Conclusión 9

VII. Referencias 10

Tabla de Ilustraciones

Imagen 1. Ejemplo de Script. 1

I. Introducción

Un script es un fragmento de código utilizado en programación para desarrollar funciones específicas dentro de aplicaciones informáticas o herramientas web. Constituye una parte esencial del software, ya que puede representar todo el código de una aplicación o una función particular. Los scripts son fundamentales en la estructura de las páginas web, permitiendo la interacción y el dinamismo en el entorno digital.

Imagen 1. Ejemplo de Script.

II. Scripts

La utilidad de los scripts abarca una amplia gama de funciones y aplicaciones. Mostramos a continuación algunas de ellas.

- **Interactividad web**: Los scripts permiten añadir interactividad a las páginas web, como menús desplegables, formularios dinámicos y validación de datos en tiempo real. Esto mejora la experiencia del usuario al hacer que la navegación sea más intuitiva y atractiva.
- **Automatización de tareas**: Pueden automatizar tareas repetitivas, como actualizar contenidos, procesar formularios o gestionar bases de datos, lo que ahorra tiempo y reduce la posibilidad de errores humanos.
- **Integración de contenidos multimedia**: Los scripts facilitan la inclusión y gestión de contenidos multimedia, como videos y audios, permitiendo su reproducción y control dentro de la página web.
- **Optimización del rendimiento**: Ayudan a optimizar la carga de las páginas web mediante la gestión eficiente de recursos, como la carga diferida de imágenes o la compresión de archivos.

- **Personalización y adaptabilidad**: Permiten personalizar la experiencia del usuario mediante la adaptación de contenidos y funcionalidades según las preferencias o el comportamiento del usuario.
- **Conexión con servidores**: Los scripts pueden interactuar con servidores para enviar y recibir datos, lo que es fundamental para aplicaciones web que requieren comunicación constante con bases de datos o servicios en la nube.
- **Desarrollo de aplicaciones web complejas**: Son la base para el desarrollo de aplicaciones web ricas y complejas, como plataformas de comercio electrónico, redes sociales y sistemas de gestión de contenido (CMS).

Lenguajes de programación para Scripts

Los lenguajes de programación para scripts son herramientas clave en el desarrollo de aplicaciones y la automatización de tareas. Estos lenguajes permiten a los desarrolladores crear funciones dinámicas y automatizar procesos tanto en entornos web como en sistemas operativos. A continuación, se presentan algunos de los lenguajes de scripting más utilizados y sus principales aplicaciones:

- **JavaScript**: Utilizado principalmente para el desarrollo web del lado del cliente. Es el lenguaje de scripting más popular para añadir interactividad a las páginas web.
- **Python**: Usado en desarrollo web, automatización, análisis de datos e inteligencia artificial. Destaca por su sintaxis clara y legible.
- **PHP**: Ideal para el desarrollo web del lado del servidor. Es comúnmente utilizado para generar páginas web dinámicas y gestionar bases de datos.
- **Ruby**: Conocido por su simplicidad y elegancia, se utiliza para el desarrollo web y automatización, especialmente con el framework Ruby on Rails.
- **Perl**: Adecuado para administración de sistemas, desarrollo web y procesamiento de texto. Es flexible y potente para escribir scripts de automatización.
- **Bash**: Usado en administración de sistemas y automatización en entornos Unix/Linux. Permite automatizar tareas administrativas y de mantenimiento.
- PowerShell: Diseñado para la administración de sistemas en entornos Windows, facilita la automatización y gestión de configuraciones.
- **Lua**: Utilizado en el desarrollo de videojuegos y aplicaciones embebidas. Es un lenguaje ligero y flexible, ideal para motores de scripting embebidos.

Ejemplos de Scripts

Los scripts son componentes esenciales en el desarrollo de aplicaciones y sitios web, proporcionando funcionalidad y dinamismo. A continuación, se presentan algunos de los scripts más conocidos que han transformado la forma en que interactuamos con la tecnología:

- **Google Analytics Tracking Script**: Utilizado para rastrear y analizar el tráfico de sitios web. Este script recoge datos sobre los visitantes, como el tiempo de permanencia y las páginas vistas, proporcionando información valiosa sobre el comportamiento del usuario.
- **jQuery**: Una popular biblioteca de JavaScript que simplifica la manipulación del DOM, la gestión de eventos y la creación de animaciones, permitiendo una interactividad mejorada en las páginas web.
- **Bootstrap**: Un framework de front-end que incluye scripts para diseñar sitios web responsivos. Facilita la creación de interfaces de usuario modernas y adaptativas.
- **WordPress Plugins**: Scripts como Contact Form 7 y Yoast SEO que añaden funcionalidades específicas a los sitios web de WordPress, como formularios de contacto y mejoras en SEO.
- **Node.js**: Plataforma utilizada para ejecutar JavaScript en el servidor, permitiendo el desarrollo de aplicaciones web dinámicas y servicios API en tiempo real.
- **Python Automation Scripts**: Herramientas como Beautiful Soup y Selenium para automatizar tareas repetitivas y realizar web scraping, recopilando datos de forma eficiente.
- **Shell Scripts (Bash)**: Utilizados en sistemas Unix/Linux para automatizar tareas administrativas, como copias de seguridad y gestión de archivos.
- **PowerShell Scripts**: Scripts diseñados para la administración de sistemas en entornos Windows, facilitando la automatización de tareas de mantenimiento y configuración.

III. Infografía de los Scripts

Link

https://create.piktochart.com/output/d7af0416baea-scripts-para-computadora

IV. Elaboración de un BATCH

Realizar un diagnóstico y mantenimiento completo en un sistema operativo Windows es una tarea que tradicionalmente requiere una cantidad considerable de tiempo y conocimientos técnicos. Un usuario o técnico tendría que navegar manualmente por múltiples herramientas y ventanas del sistema, que están dispersas y no presentan la información de manera consolidada.

La solución propuesta es la creación de un único script de PowerShell que automatiza todo el proceso de diagnóstico, limpieza y reporte. PowerShell es la interfaz de línea de comandos y lenguaje de scripting más potente integrado en Windows, diseñado específicamente para la automatización y la administración de sistemas.

Este script actúa como un técnico en un archivo, ejecutando una secuencia predefinida de más de 15 comandos de manera ordenada. La solución aborda los problemas de la siguiente manera:

- **Automatización**: Ejecuta todas las tareas de forma secuencial sin necesidad de intervención manual después de iniciarlo.
- **Centralización**: Utiliza una amplia gama de comandos de PowerShell que acceden a toda la información del sistema desde un solo lugar.
- **Mantenimiento Activo**: No solo recopila datos, sino que también realiza acciones de limpieza, como eliminar archivos temporales y vaciar la papelera de reciclaje.
- **Reporte Unificado**: Toda la información recopilada se formatea y se guarda en un único archivo de texto en el Escritorio del usuario, creando un registro claro y fácil de leer del estado del sistema en un momento específico.

El script esta conformado de la siguiente forma:

Primero revisa si el script tiene los permisos necesarios.

```powershell
# --- 1. Verificacion de Permisos ---
Write-Host "Verificando permisos de Administrador..." -ForegroundColor Yellow
if (-NOT ([System.Security.Principal.WindowsPrincipal][System.Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole([System.Security.Principal.WindowsBuiltInRole]::Administrator)) {
    Write-Warning "Este script necesita ser ejecutado como Administrador. Por favor, haz clic derecho sobre el archivo y selecciona 'Ejecutar con PowerShell como Administrador'."
    # Pausa para que el usuario pueda leer el mensaje antes de cerrar.
    Start-Sleep -Seconds 10
    # Termina la ejecucion del script si no es administrador.
    Exit
}
```

Lo siguiente es que recopila la información del sistema.

```powershell
# --- 2. Preparacion del Script ---
# Limpia la pantalla de la consola.
Clear-Host
# Define la ruta del archivo de reporte que se creara en el Escritorio.
$reportePath = [System.Environment]::GetFolderPath('Desktop') + "\Reporte_Sistema_$(Get-Date -Format 'yyyy-MM-dd').txt"
Write-Host "El reporte se guardara en: $reportePath" -ForegroundColor Cyan

# --- 3. Recopilacion de Informacion del Sistema ---
Write-Host "Recopilando informacion del sistema..." -ForegroundColor Green
# Comando 1: Obtener informacion general del computador y guardarla en el reporte.
Get-ComputerInfo | Out-File -FilePath $reportePath
```

Obtenemos la información del sistema y el tiempo de sesión, después verificamos los discos duros.

```powershell
# Comando 2: Obtener informacion del sistema operativo y su tiempo de actividad.
$osInfo = Get-CimInstance -ClassName Win32_OperatingSystem
$uptime = (Get-Date) - $osInfo.LastBootUpTime
Add-Content -Path $reportePath -Value "`n--- Tiempo de Actividad ---"
Add-Content -Path $reportePath -Value "El sistema ha estado activo por: $($uptime.Days) dias, $($uptime.Hours) horas, $($uptime.Minutes) minutos."

# Comando 3: Verificar el espacio libre en los discos duros.
Add-Content -Path $reportePath -Value "`n--- Estado de los Discos ---"
Get-Volume | Format-Table -AutoSize | Out-String -Width 4096 | Add-Content -Path $reportePath
```

Obtenemos la información de red, la conexión a internet y servicios conectados.

```powershell
# Comando 4: Obtener informacion de los adaptadores de red.
Add-Content -Path $reportePath -Value "`n--- Adaptadores de Red ---"
Get-NetAdapter | Select-Object Name, InterfaceDescription, Status | Format-Table -AutoSize | Out-String -Width 4096 | Add-Content -Path $reportePath

# Comando 5: Probar la conexion a Internet.
Add-Content -Path $reportePath -Value "`n--- Prueba de Conexion a Internet ---"
Test-NetConnection -ComputerName "google.com" -InformationLevel "Quiet" | Out-String -Width 4096 | Add-Content -Path $reportePath

# --- 4. Analisis del Sistema ---
Write-Host "Analizando el estado del sistema..." -ForegroundColor Green
# Comando 6: Buscar servicios configurados como 'Automatico' que no se esten ejecutando.
Add-Content -Path $reportePath -Value "`n--- Servicios Automaticos Detenidos ---"
Get-Service | Where-Object { $_.StartType -eq 'Automatic' -and $_.Status -ne 'Running' } | Select-Object Name, DisplayName, Status | Format-Table -Aut
```

Se busca los 10 archivos mas grandes y los errores críticos.

```
# Comando 7: Encontrar los 10 archivos mas grandes en la unidad C: (puede tardar).
Write-Host "Buscando los 10 archivos mas grandes en C:\ (esto puede tardar unos minutos)..." -ForegroundColor Yellow
Add-Content -Path $reportePath -Value "`n--- Top 10 Archivos mas Grandes en C: ---"
Get-ChildItem -Path "C:\" -Recurse -File -ErrorAction SilentlyContinue | Sort-Object Length -Descending | Select-Object -First 10

# Comando 8: Obtener los últimos 5 errores criticos del Visor de Eventos del Sistema.
Add-Content -Path $reportePath -Value "`n--- Ultimos 5 Errores Criticos del Sistema ---"
Get-EventLog -LogName System -EntryType Error -Newest 5 | Format-List | Out-String -Width 4096 | Add-Content -Path $reportePath
```

Aquí empieza el mantenimiento de la computadora. Borra archivos de la carpeta de reciclaje y comprueba archivos de sistema.

```
# --- 5. Tareas de Mantenimiento ---
Write-Host "Realizando tareas de mantenimiento..." -ForegroundColor Green
# Comando 9: Limpiar la carpeta de archivos temporales del usuario.
Write-Host "Limpiando archivos temporales..." -ForegroundColor Yellow
Remove-Item -Path "$env:TEMP\*" -Recurse -Force -ErrorAction SilentlyContinue
Add-Content -Path $reportePath -Value "`n--- Limpieza de Archivos Temporales ---`nLimpieza completada con exito."

# Comando 10: Vaciar la Papelera de Reciclaje (sin pedir confirmacion).
Write-Host "Vaciando la Papelera de Reciclaje..." -ForegroundColor Yellow
Clear-RecycleBin -Force -ErrorAction SilentlyContinue
Add-Content -Path $reportePath -Value "`n--- Vaciado de la Papelera de Reciclaje ---`nVaciado completado con exito."

# Comando 11: Ejecutar el Comprobador de Archivos de Sistema (SFC).
Write-Host "Ejecutando SFC /scannow para verificar la integridad de los archivos de sistema..." -ForegroundColor Yellow
sfc.exe /scannow
```

Ya por último busca actualizaciones de Windows, busca programas instalados y enlista los procesos que consumen mas CPU. Para finalizar comprueba el estado del firewall y escribe todo el reporte en el escritorio.

```
# Comando 12: Iniciar una búsqueda de actualizaciones de Windows.
Write-Host "Abriendo la configuracion de Windows Update..." -ForegroundColor Yellow
Start-Process ms-settings:windowsupdate

# Comando 13: Obtener lista de programas instalados.
Add-Content -Path $reportePath -Value "`n--- Programas Instalados ---"
Get-ItemProperty HKLM:\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\* | Select-Object DisplayName,

# Comando 14: Listar los 10 procesos que mas CPU consumen.
Add-Content -Path $reportePath -Value "`n--- Procesos con Mayor Consumo de CPU ---"
Get-Process | Sort-Object CPU -Descending | Select-Object -First 10 | Format-Table -AutoSize | Out-String -Width 4096

# Comando 15: Comprobar el estado del firewall.
Add-Content -Path $reportePath -Value "`n--- Estado del Firewall ---"
Get-NetFirewallProfile | Format-Table -AutoSize | Out-String -Width 4096 | Add-Content -Path $reportePath

# --- 6. Finalizacion ---
Write-Host "`n¡Proceso completado con exito!" -ForegroundColor Green
Write-Host "Revisa el archivo 'Reporte_Sistema.txt' en tu Escritorio para ver los resultados." -ForegroundColor Cyan
# Comando 16 (extra): Abrir el reporte automaticamente.
Invoke-Item $reportePath
```

V. Links del Proyecto

Link al video de demostración:

https://drive.google.com/file/d/1ICurvvbLl27SAo9ovxv3VannVAd71ZFV/view?usp=sharing

VI. Conclusión

Los scripts son parte fundamental del sistema operativo. Se usan en todos lados y por todas las aplicaciones que tenemos. Es importante que como ingenieros aprendamos a manejar estos scripts a nuestro favor.

Existen varios tipos de scripts, pueden desde apagar cosas, hasta automatizar la creación de carpetas y archivos. A diferencia del código, los scripts suelen trabajar directamente con el sistema operativo, lo cual permite una gran personalización. Esto nos permite la creación de scripts enfocados en diferentes cosas.

VII. Referencias

- *Qué es Script*. (n.d.). Arimetrics.com. Retrieved October 28, 2025, from https://www.arimetrics.com/glosario-digital/script