Universidad Nacional de La Matanza
Florencio Varela 1903 - San Justo - Buenos Aires - Argentina

Departamento de Ingeniería e Investigaciones Tecnológicas

Cátedra de Virtualización de Hardware (3654)

Jefe de Cátedra:	**Alexis Villamayor**
Docentes:	**Alexis Villamayor, Fernando Boettner**
Jefe de trabajos prácticos:	**Ramiro de Lizarralde**
Ayudantes:	**Alejandro Rodriguez, Fernando Piubel**

Año: **2024 – Primer Cuatrimestre**

Actividad Práctica de Laboratorio Nº 1
Bash y Powershell

Condiciones de entrega

- Se debe entregar por plataforma MIEL un archivo con formato ZIP o TAR (no se aceptan RAR u otros formatos de compresión/empaquetamiento de archivos), conteniendo la carátula que se publica en MIEL junto con los archivos de la resolución del trabajo.
- Se debe entregar el código fuente de cada uno de los ejercicios resueltos tanto en Bash como en Powershell. Si un ejercicio se resuelve en un único lenguaje se lo considerará incompleto y por lo tanto, desaprobado.
- Se deben entregar lotes de prueba válidos para los ejercicios que reciban archivos o directorios como parámetro.
- Los archivos de código deben tener un encabezado en el que se listen los integrantes del grupo.
- La entrega se realizará en formato zip, con un directorio para cada uno de los ejercicios, sin espacios en su nombre con la siguiente estructura:
    - Bash
        - Ejercicio1
        - Ejercicio2
        - Ejercicio3
        - Ejercicio4
        - Ejercicio5
    - PowerShell
        - Ejercicio1
        - Ejercicio2
        - Ejercicio3
        - Ejercicio4
        - Ejercicio5

Criterios de corrección y evaluación generales para todos los ejercicios

- La corrección se realizará sobre Ubuntu 22.04 LTS de Windows (WSL).
- Los scripts de bash muestran una ayuda con los parámetros “-h” y “--help”. Deben permitir el ingreso de parámetros en cualquier orden, y no por un orden fijo.
- Los scripts de Powershell deben mostrar una ayuda con el comando Get-Help. Ej: “Get-Help ./Ejercicio1.ps1”. Deben realizar la validación de parámetros en la sección params utilizando la funcionalidad nativa de Powershell.
- Cuando haya parámetros que reciban rutas de directorios o archivos se deben aceptar tanto rutas relativas como absolutas o que contengan espacios.
- No se debe permitir la ejecución del script si al menos un parámetro obligatorio no está presente.
- Si algún comando utilizado en el script da error, este se debe manejar correctamente: detener la ejecución del script (o salvar el error en caso de ser posible) y mostrar un mensaje informando el problema de una manera amigable con el usuario, pensando que el usuario **no** tiene conocimientos informáticos.
- Si se generan archivos temporales de trabajo se deben crear en el directorio temporal /tmp; y se deben eliminar al finalizar el script, tanto en forma exitosa como por error, para no dejar archivos basura. (Ver trap en bash / try-catch-finally en powershell)
- Deseable:
    - Utilización de funciones en el código para resolver los ejercicios.

Ejercicio 1

Objetivos de aprendizaje: manejo de archivos CSV, manejo de parámetros y salida por pantalla.

Una institución educativa almacena los resultados de los finales que se toman al final de cada cuatrimestre en archivos CSV (uno por mesa de examen y donde el nombre del archivo es el código de la materia) con el siguiente formato:

```
DNI-alumno,nota-ej-1,nota-ej-2,...,nota-ej-N
10100100,b,r,...,m
20200200,r,b,...,b
```

Una vez finalizadas todas las mesas de examen, se corre un proceso que se encarga de realizar un resumen de las notas de cada alumno para luego poder publicarlo en un sitio web. Para calcular la nota de cada final, se debe tener en cuenta que:

- Cada ejercicio tiene el mismo peso en la nota final. Para calcularlo se puede usar la siguiente fórmula: 10 / CantidadEjercicios.
- Un ejercicio bien (B) vale el ejercicio entero.
- Un ejercicio regular (R) vale medio ejercicio.
- Un ejercicio mal (M) no suma puntos a la nota final.
- Cada materia puede tener diferente cantidad de ejercicios.

Desarrollar dos scripts, uno en bash y otro en PowerShell que resuma la información de todos los archivos CSV en un único archivo JSON con el siguiente formato:

```
{
"notas": [
  {
    "dni": "12345678",
    "notas": [
      { "materia": 1115, "nota": 8 },
      { "materia": 1116, "nota": 2 }
    ]
  },
  {
    "dni": "87654321",
    "notas": [
      { "materia": 1116, "nota": 9 },
      { "materia": 1118, "nota": 7 }
    ]
  }
] }
```

Consideraciones:

- Se asume que solamente hay un archivo de notas por materia.
- La nota calculada se trunca para no tener decimales.
- El formato del JSON tiene que ser válido.

Parámetros:

Parámetro bash	Parámetro Powershell	Descripción
-d / --directorio	-directorio	Ruta del directorio que contiene los archivos CSV a procesar.
-s / --salida	-salida	Ruta del archivo JSON de salida.
-p / --pantalla	-pantalla	Muestra la salida por pantalla, no genera el archivo JSON. Este parámetro no se puede usar a la vez que -s.

Ejercicio 2
Objetivos de aprendizaje: arrays y matrices.

Desarrollar dos scripts, uno en bash y otro en Powershell, que multipliquen dos matrices cargadas en archivos separados y muestre el resultado por pantalla.

Del resultado también se deberá informar:

- Orden de la matriz.
- Si es cuadrada.
- Si es fila.
- Si es columna.

Parámetros:

Parámetro bash	Parámetro Powershell	Descripción
-m1 / --matriz1	-matriz1	Ruta del archivo de la matriz 1.
-m2 / --matriz2	-matriz2	Ruta del archivo de la matriz 2.
-s / --separador	-separador	Carácter separador de valores. Opcional, por defecto es coma “,”.

Los archivos que contienen la matriz tienen el siguiente formato:

1,3,5
4,6,7
6,4,-3

Aclaraciones:

1. Tener en cuenta que los archivos pueden contener matrices inválidas, esto puede ser porque la cantidad de columnas por fila no coincidan o que contengan algún valor que no sea numérico o incluso ser un archivo vacío.

2. La operación de multiplicación es siempre (matriz 1) x (matriz 2). Realizar las validaciones correspondientes para asegurar que se pueda realizar la operación de multiplicación.
3. El carácter separador puede ser cualquier carácter salvo números y el símbolo menos (“-”) para no confundir con los números negativos (esto debe ser parte de las validaciones del script).

Ejercicio 3

Objetivos de aprendizaje: arrays asociativos, búsqueda de archivos, manejo de archivos, AWK

Desarrollar dos scripts, uno en bash y otro en Powershell, que analicen los archivos de texto en un directorio pasado por parámetro e informen un resumen final con:

- La cantidad de ocurrencias de palabras de X caracteres. Se deben incluir todos los largos de caracteres. Ejemplo:
  Palabras de 1 caracter: 5
  Palabras de 2 caracteres: 10
  ...
  Palabras de N caracteres: 32
- La palabra o palabras que más ocurrencias tuvo, puede ser más de una si la cantidad de ocurrencias es igual.
- La cantidad total de palabras.
- El promedio de palabras por archivo (cantidad de palabras sobre el total de archivos analizados)
- El carácter más repetido.

Consideraciones:

- El separador por defecto de palabras es el espacio. Sin embargo, se puede configurar utilizando el parámetro -s / --separador / -separador. Por ejemplo, si el carácter separador es “a”, entonces la cadena “virtualización” tiene 3 palabras: “virtu”, “liz” y “ción”.

Parámetros:

Parámetro bash	**Parámetro Powershell**	**Descripción**
-d / --directorio	-directorio	Ruta del directorio a analizar.
-x / --extension	-extension	Si está presente, indica la extensión de los archivos a analizar. Opcional.
-s / --separador	-separador	Un carácter separador de palabras. Opcional. Valor por defecto: “ “ (espacio). Ignora si es mayúscula o minúscula.
-o / --omitir	-omitir	Array de caracteres a buscar en las palabras del archivo. Si ese carácter se encuentra en la palabra, esta no debe ser contabilizada.

Ejercicio 4

Objetivos de aprendizaje: procesos demonios, monitoreo de directorios, herramientas de compresión y archivado

Desarrollar dos scripts, uno en bash y otro en Powershell, que realicen el monitoreo en segundo plano de los archivos de un directorio. Cada vez que se detecte un cambio, ya sea creación o modificación de un archivo (no se monitorea borrado), el script debe realizar la búsqueda de un patrón determinado en el contenido del archivo y generar un backup del archivo creado/modificado y registrar en un archivo de log con la fecha, hora, ruta del directorio monitoreado y un detalle de los resultados encontrados.

El backup es un archivo comprimido con el archivo identificado. Para el script de bash, el archivo es de extensión ".tar.gz". En el caso de Powershell, la extensión es ".zip". El nombre de los archivos de backup debe tener el siguiente formato: "yyyyMMdd-HHmmss", donde:

- yyyy: año con 4 dígitos.
- MM: mes con 2 dígitos.
- dd: día con 2 dígitos.
- HH: hora con dos dígitos en formato 24 horas.
- mm: minutos con 2 dígitos.
- ss: segundos con 2 dígitos.

Aclaraciones:

1. El script debe quedar ejecutando por sí solo en segundo plano, el usuario no debe necesitar ejecutar ningún comando adicional a la llamada del propio script para que quede ejecutando como demonio en segundo plano. La solución debe utilizar un único archivo script (por cada tecnología), no se aceptan soluciones con dos o más scripts que trabajen en conjunto.
2. El script debe poder ejecutarse nuevamente para finalizar el demonio ya iniciado. Debe validar que esté en ejecución sobre el directorio correspondiente.
3. No se debe poder ejecutar más de 1 proceso demonio para un determinado directorio al mismo tiempo.
4. El monitoreo del directorio se debe hacer utilizando inotify-tools en bash y FileSystemWatcher en Powershell.

Ejemplo de uso:
$ ./demonio.sh -d ../monitor --salida ../salida
> ./demonio.ps1 -directorio ../monitor -salida ../salida

$ ./demonio.sh -d ../monitor --kill
> ./demonio.ps1 -directorio ../monitor -kill

Parámetro bash	Parámetro Powershell	Descripción
-d / --directorio {directorio}	-directorio	Ruta del directorio a monitorear

-s / --salida {directorio}	-salida	Ruta del directorio en donde se van a crear los backups.
-p / --patron {patrón}	-patron	Patrón a buscar una vez detectado un cambio en los archivos monitoreados.
-k / --kill	-kill	Flag que se utiliza para indicar que el script debe detener el demonio previamente iniciado. Este parámetro solo se puede usar junto con -d/--directorio/-directorio.

Ejercicio 5

Objetivos de aprendizaje: conexión con APIs y web services, manejo de archivos y objetos json, cache de información

Se necesita implementar un script que facilite la consulta de información relacionada a la serie Rick and Morty. El script permitirá buscar información de los personajes por su id o su nombre a través de la api https://rickandmortyapi.com/ y pueden enviarse más de 1 id o nombre en la ejecución del script e incluso solicitar la búsqueda por ambos parámetros. Tener en cuenta que al buscar por nombre el resultado será una lista a diferencia de la búsqueda por ID.
Una vez obtenida la información, se generará un archivo con la información obtenida, para evitar volver a consultarlo a la api, y se mostrará por pantalla la información básica de él/los personaje/s con el siguiente formato.
La información de cómo obtener los datos se puede consultar en el siguiente link:
https://rickandmortyapi.com/documentation/#character

Por ejemplo:
https://rickandmortyapi.com/api/character/?name=rick
https://rickandmortyapi.com/api/character/1
https://rickandmortyapi.com/api/character/1,2,3

Character info: (uno por cada resultado encontrado)
Id: 1
Name: Rick Sanchez
Status: Alive
Species: Human
Gender: Male
Origin: Earth (C-137)
Location: Citadel of Ricks

En caso de ingresar un id inválido o un nombre que no traiga resultados, se deberá informar al cliente un mensaje acorde. Mismo en caso de que la api retorne un error.

Ejemplo de uso:

$ ./rickandmorty.sh --id “1,2” --nombre “rick, morty”
> ./rickandmorty.ps1 -id 1,2 -nombre rick, morty

Parámetro bash	**Parámetro Powershell**	**Descripción**
-i / --id	-id	Id o ids de los personajes a buscar.
-n / --nombre	-nombre	Nombre o nombre de los personajes a buscar.

Los parámetros al utilizar Powershell deben ser de tipo array, es decir, no es correcto que sea un string y luego utilizar una función como “.split(‘,’)” para obtener los valores correspondientes.