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 - Segundo 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.
- Los archivos con el código de cada ejercicio y sus lotes de prueba se deben ubicar en un directorio con la siguiente estructura:
    - APL1/
        - bash/
            - ejercicio1
            - ejercicio2
            - ejercicio3
            - ejercicio4
            - ejercicio5
        - powershell/
            - ejercicio1
            - ejercicio2
            - ejercicio3
            - ejercicio4
            - ejercicio5

Criterios de corrección y evaluación generales para todos los ejercicios

- 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

Se quiere automatizar la validación de jugadas de las distintas agencias de lotería de la ciudad. Para ello se requiere crear un script que pueda procesar todas las jugadas generadas en la semana y validar si existen ganadores o quienes obtuvieron premios secundarios (4 o 3 aciertos).
Las diversas agencias de lotería recibirán las jugadas de los participantes, quienes indican los 5 números para su jugada (valores del 0 al 99) y luego enviarán estos datos para su procesamiento con el siguiente formato:

- 1 archivo por agencia.
- El nombre del archivo indicará la agencia.
- Cada archivo contiene las jugadas recolectadas durante la semana en formato CSV.
    - Número de jugada (id)
    - Los 5 números de la jugada

Por ejemplo:

1.csv

```
1,33,42,3,55,99
2,1,3,4,67,54
3,44,55,66,77,88
```

Para facilitar las pruebas y validación del script, los 5 números ganadores estarán disponibles en otro archivo, también en formato CSV, con el cuál se realizarán las validaciones y generación de resultados.
Al finalizar el procesamiento, se indicará por pantalla en formato JSON las jugadas con 5, 4 y 3 aciertos.

```
{
  "5_aciertos": [
    {
      "agencia": "1",
      "jugada": "3"
    }
  ],
  "4_aciertos": [
    {
      "agencia": "2",
      "jugada": "3"
    },
    {
      "agencia": "3",
      "jugada": "3"
    }
  ],
  "3_aciertos": [
    {
      "agencia": "2",
      "jugada": "1"
    },
    {
```

```
      "agencia": "2",
      "jugada": "34"
    },
    {
      "agencia": "1",
      "jugada": "33"
    }
  ]
}
```


Consideraciones:

1. El formato del JSON tiene que ser válido.
2. Los resultados se pueden mostrar tanto por pantalla como en un archivo, pero solo se puede generar 1 de las 2 formas en una ejecución.
3. Los archivos no tienen encabezado.

Parámetros:

Parámetro bash	Parámetro PowerShell	Descripción
-d / --directorio	-directorio	Ruta del directorio que contiene los archivos CSV a procesar.
-a / --archivo	-archivo	Ruta del archivo JSON de salida (no es un directorio, es la ruta completa incluyendo el nombre del archivo). Este parámetro no se puede usar a la vez que –p o -pantalla.
-p / --pantalla	-pantalla	Muestra la salida por pantalla, no genera el archivo JSON. Este parámetro no se puede usar a la vez que –a o -archivo.

Ejercicio 2

Objetivos de aprendizaje: arrays y matrices.

Desarrollar un script que permita realizar producto escalar y trasposición de matrices. La entrada de la matriz al script se realizará mediante un archivo de texto plano. La salida se guardará en otro archivo que se llamará “salida.nombreArchivoEntrada” y se generará en el mismo directorio donde se encuentre el archivo de la matriz que se está procesando (puede ser cualquier directorio, considerar que no necesariamente es donde se ejecuta el script).

El formato de los archivos de matrices debe ser el siguiente:

```
0|1|2
1|1|1
-3|-3|-1
```

Consideraciones:

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. 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).
3. Los valores de la matriz serán numéricos. Esto incluye valores decimales y negativos.

Parámetros:

Parámetro bash	**Parámetro PowerShell**	**Descripción**
-m / --matriz	-matriz	Ruta del archivo de la matriz.
-p / --producto	-producto	Valor entero para utilizarse en el producto escalar. No se puede usar junto con -t o -trasponer
-t / --trasponer	-trasponer	Indica que se debe realizar la operación de trasposición sobre la matriz. (no recibe valor adicional, solo el parámetro) No se puede usar junto a –p o –producto.
-s / --separador	-separador	Carácter para utilizarse como separador de columnas.

Ejercicio 3

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

Desarrollar un script que identifique los archivos duplicados en un directorio (incluyendo los subdirectorios). Para esto, se considerará que un archivo está duplicado, si su nombre y tamaño son iguales, sin importar su contenido.

La salida del script debe ser un listado solo con los nombres de los archivos duplicados y en qué path fueron encontrados, por ejemplo:

ejercicio3.sh
/home/user/apl
/home/user/apl/final
/home/user/apl/final/final

Consideraciones:

1. En bash el procesamiento de la información se realizará mediante AWK. Para esto pueden utilizar el resultado del comando ls donde podrán obtener tanto el nombre como el tamaño de los archivos.

Parámetros:

Parámetro bash	**Parámetro PowerShell**	**Descripción**

-d / --directorio	-directorio	Ruta del directorio a analizar.

Ejercicio 4

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

Evaluar si el archivo creado es igual a otro y en caso de serlo, generar un backup con esos archivos.

Con el script anterior se detectó que en el FS hay muchos archivos duplicados, por lo que se quiere realizar algo para evitarlo. Para esto, le piden crear un proceso que monitoree un directorio (incluyendo los subdirectorios) y valide si se está generando un archivo duplicado (bajo el mismo criterio del Ejercicio 3) y en caso de ocurrir se genere un log y se archive en un archivo comprimido.

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.

Consideraciones:

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	Ruta del directorio a monitorear
-s / --salida	-salida	Ruta del directorio en donde se van a crear los backups.
-k / --kill	-kill	Flag (switch) que se utiliza para indicar que el script debe detener el demonio previamente iniciado. Este parámetro solo se puede usar junto con –d o -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 al mundo de Star Wars. El script permitirá buscar información de los personajes y películas por su id a través de la api swapi.tech y pueden enviarse más de 1 id para personajes o películas en la ejecución del script e incluso solicitar la búsqueda por ambos parámetros.
Una vez obtenida la información, se generará a modo de caché, para evitar volver a consultarlo a la api, y se mostrará por pantalla la información básica de él o los personajes o películas con el siguiente formato:

Personajes: (uno por cada resultado encontrado)
Id: 1
Name: Luke Skywalker
Gender: male
Height: 172
Mass: 77
Birth Year: 19BBY

Películas: (uno por cada resultado encontrado)
Title: A New Hope
Episode id: 4
Release date: 1977-05-25
Opening crawl: It is a period of civil war...

La información de cómo obtener los datos se puede consultar en el siguiente link:
https://www.swapi.tech/documentation

Por ejemplo:
https://www.swapi.tech/api/people/1
https://www.swapi.tech/api/films/1

En caso de ingresar un id inválido, se deberá informar al cliente un mensaje acorde. Mismo en caso de que la api retorne un error.

Ejemplo de uso:
Bash:
$ ./swapi.sh --people "1,2" --film "1,2"
O
$ ./swapi.sh -p "1,2" -f "1,2"

PS:
> ./swapi.ps1 -people 1,2 –film 1,2

Consideraciones:

1. 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.

Parámetros:

Parámetro bash	**Parámetro PowerShell**	**Descripción**
-p / --people	-people	Id o ids de los personajes a buscar.
-f / --film	-film	Id o ids de las películas a buscar.