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:	**Fernando Boettner**
Jefe de trabajos prácticos:	**Ramiro de Lizarralde**
Ayudantes:	**Alejandro Rodriguez, Fernando Piubel**

Año: **2025 – Segundo cuatrimestre**

Actividad Práctica de Laboratorio

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:
    - APL/
        - 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: Análisis de resultados de encuestas de satisfacción de clientes

Objetivos de aprendizaje: Manejo de archivos de texto, procesamiento de datos tabulares, manejo de parámetros y salida por pantalla.

Se requiere un script para analizar los resultados de encuestas de satisfacción de clientes de un servicio de atención al cliente. Los datos se registran diariamente en archivos de texto, con cada encuesta en una línea.

El archivo de registro tiene un formato de campos fijos, donde la posición de cada campo indica su significado, y los campos están separados por un pipe (|). El nombre del archivo tendrá la fecha de registro de las encuestas.

Formato: ID_ENCUESTA|FECHA|CANAL|TIEMPO_RESPUESTA|NOTA_SATISFACCION
Campos:

- **ID_ENCUESTA:** numérico
- **FECHA:** texto (yyyy-mm-dd hh:mm:ss)
- **CANAL:** texto (Teléfono, Email, Chat)
- **TIEMPO_RESPUESTA:** numérico (en minutos)
- **NOTA_SATISFACCION:** numérico (de 1 a 5)

Ejemplo de archivo de entrada (2025-07-01.txt)

101|2025-07-01 10:22:33|Telefono|5.5|4
102|2025-07-01 12:23:11|Email|120|5
103|2025-07-01 22:34:43|Chat|2.1|3
104|2025-06-30 23:11:10|Telefono|7.8|2

Se requiere un script que procese todos los archivos de encuestas en un directorio, calcule el **tiempo de respuesta promedio y la nota de satisfacción promedio** por canal de atención y por día. El resultado debe ser un archivo o una impresión en pantalla, ambas en formato JSON.

Ejemplo de salida JSON

```
{
  "2025-06-30": {
    "Telefono": {
      "tiempo_respuesta_promedio": 7.8,
      "nota_satisfaccion_promedio": 2
    },
  },
  "2025-07-01": {
    "Telefono": {
      "tiempo_respuesta_promedio": 7.8,
      "nota_satisfaccion_promedio": 2
    },
    "Email": {
      "tiempo_respuesta_promedio": 120,
      "nota_satisfaccion_promedio": 5
    },
    "Chat": {
      "tiempo_respuesta_promedio": 2.1,
      "nota_satisfaccion_promedio": 3
    }
```

```
    }
}
```

Consideraciones:

1. Tener en cuenta que, por diferentes motivos, un archivo puede tener registros con fechas que no sean la misma que la indicada en su nombre.

Parámetros:

Parámetro bash	Parámetro PowerShell	Descripción
-d / --directorio	-directorio	Ruta del directorio con los archivos de encuestas a procesar
-a / --archivo	-archivo	Ruta completa del archivo JSON de salida. No se puede usar con -p / -pantalla
-p / --pantalla	-pantalla	Muestra la salida por pantalla. No se puede usar con -a / -archivo.

Ejercicio 2: Análisis de rutas en un mapa de transporte
Objetivos de aprendizaje: arrays y matrices.

Desarrollar un script para analizar rutas en una red de transporte público. La información de la red se representa como una matriz de adyacencia donde los valores representan el tiempo de viaje entre estaciones. El script debe ser capaz de determinar si una estación es un "hub" (estación con más conexiones) o encontrar el camino más corto en tiempo entre todas las estaciones. En caso de haber más de un camino, mostrará todos aquellos que cumplan con la condición. La salida se guardará en un archivo informe.nombreArchivoEntrada en el mismo directorio del archivo original.

El camino más corto se calculará usando el algoritmo de Dijkstra. Este algoritmo encuentra la ruta con menor peso (en este caso, tiempo de viaje) entre dos nodos en un grafo. Se debe implementar la lógica de este algoritmo para resolver el problema.

Ejemplo de archivo de entrada (mapa_transporte.txt)

```
0|10|0|5
10|0|4|0
0|4|0|8
5|0|8|0
```

En este ejemplo, la matriz representa la conexión entre 4 estaciones (1, 2, 3, 4). Un 0 indica que no hay conexión directa o qué es la misma estación. Los otros valores son el tiempo en minutos.

Ejemplo de salida del informe (varía según el parámetro recibido)

```
## Informe de análisis de red de transporte
**Hub de la red:** Estación 2 (4 conexiones)
**Camino más corto: entre Estación 1 y Estación 4:**
**Tiempo total:** 9 minutos
**Ruta:** 1 -> 2 -> 3 -> 4
```

Consideraciones:

1. El script debe validar que el archivo de entrada sea una matriz cuadrada y simétrica con valores numéricos
2. Los valores de la matriz serán enteros o decimales.

Parámetros:

Parámetro bash	Parámetro PowerShell	Descripción
-m / --matriz	-matriz	Ruta del archivo de la matriz de adyacencia.
-h / --hub	-hub	Determina qué estación es el "hub" de la red. No se puede usar junto con -c / -camino.
-c / --camino	-camino	Encuentra el camino más corto en tiempo. No se puede usar junto a -h / -hub.
-s / --separador	-separador	Carácter para utilizarse como separador de columnas.

Ejercicio 3: Conteo de eventos en logs de sistemas

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

Desarrollar un script que analice todos los archivos de logs (archivos con extensión .log) en un directorio para contar la ocurrencia de eventos específicos. Los eventos a buscar se proporcionarán como una lista de palabras clave.

Ejemplo de archivo de entrada (system.log)

```
Aug 23 10:00:01 server.local kernel: [256.789] USB device plugged in.
Aug 23 10:00:05 server.local sshd[1234]: Invalid user from 192.168.1.1.
Aug 23 10:00:10 server.local sudo[5678]: Command not found.
Aug 23 10:00:15 server.local kernel: [258.123] USB device unplugged.
Aug 23 10:00:20 server.local sshd[1234]: Invalid user from 192.168.1.2.
```

En este ejemplo, las palabras clave podrían ser "USB" y "Invalid".

Ejemplo de salida

USB: 2
Invalid: 2

Consideraciones:

1. El script debe usar AWK para procesar la información en Bash.
2. Las palabras clave deben ser de tipo array en PowerShell. En bash estarán separadas por comas. Por ejemplo: “usb,invalid”.
3. Las búsquedas deben ser **case-insensitive**.

Parámetros:

Parámetro bash	Parámetro PowerShell	Descripción
-d / --directorio	-directorio	Ruta del directorio de logs a analizar.
-p / --palabras	-palabras	Lista de palabras clave a contabilizar.

Ejercicio 4: Análisis de seguridad de código en repositorios Git

Objetivos de aprendizaje: Procesos demonios, manejo de archivos de configuración, búsqueda y reemplazo de texto.

Se necesita un script demonio para monitorear un repositorio Git y **detectar credenciales o datos sensibles** que se hayan subido por error. El demonio debe leer un archivo de configuración que contiene una lista de **palabras clave o patrones regex** a buscar (por ejemplo, password, API_KEY, "API_KEY = "). Cada vez que se detecte una nueva modificación en la rama principal del repositorio, el demonio debe escanear los archivos modificados. Si encuentra alguna coincidencia, debe registrar una alerta en un archivo de log con el nombre del archivo, el patrón encontrado y la fecha. El script debe ejecutarse en segundo plano, liberando la terminal.

Consideraciones:

1. La solución debe ser un único script que pueda ser ejecutado y detenido posteriormente.
2. No se puede ejecutar más de un proceso demonio para el mismo repositorio.
3. El monitoreo debe ser activado por los cambios identificados en los archivos del directorio.
4. La lista de patrones a buscar debe ser configurable en un archivo externo.
5. El script debe poder ser detenido con un flag.

Ejemplo de archivo de configuración (patrones.conf)

```
password
API_KEY
secret
regex:^.*API_KEY\s*=\s*['"].*['"].*$
```

Ejemplo de entrada del demonio:

$ ./audit.sh -r /home/user/myrepo -c ./patrones.conf -a 10
> ./audit.ps1 -repo /home/user/myrepo -configuracion ./patrones.conf -alerta 10

Ejemplo de salida en el archivo de log:

```
[2025-08-23 11:30:00] Alerta: patrón 'API_KEY' encontrado en el archivo
'config.js'.
```

Parámetro bash	Parámetro PowerShell	Descripción
-r / --repo	-repo	Ruta del repositorio Git a monitorear.
-c / --configuracion	-configuracion	Ruta del archivo de configuración que contiene la lista de patrones a buscar.
-l / --log	-log	Ruta del archivo de logs que contiene la lista de eventos identificados.
-k / --kill	-kill	Flag para detener el demonio. Solo se usa junto con -r / -repo y debe validar que exista un demonio en ejecución.

Ejercicio 5: Buscador de información de países
Objetivos de aprendizaje: Conexión con APIs y web services, manejo de archivos y objetos JSON, caché de información

Se necesita un script para consultar información de países utilizando una API pública. El script permitirá buscar países por nombre y una vez que obtiene la información de un país, se debe guardar en un archivo de caché para evitar futuras consultas a la API. Los detalles relevantes de cada país deben mostrarse por pantalla con el formato mencionado más adelante.
Los resultados guardados en el archivo caché deberán tener un TTL (time to live) que indica durante cuánto tiempo es válido ese resultado. Pasado ese tiempo deberá consultar nuevamente a la API para actualizar los valores en caché de ese elemento.

Documentación de la API:
La API de **REST Countries** no requiere registro. La consulta se realiza por nombre de país. **URL de la API:** https://restcountries.com/v3.1/name/{nombre}

Ejemplo de llamada cURL:
curl "https://restcountries.com/v3.1/name/spain"

Ejemplo de salida esperada (por cada país encontrado):

```
País: Spain
Capital: Madrid
Región: Europe
Población: 47615034
Moneda: Euro (EUR)
```

Consideraciones:

1. Los nombres de los países pueden ser múltiples y deben ser de tipo array en PowerShell.
2. El archivo de caché debe persistir las consultas durante un tiempo determinado (TTL).

Parámetros:

Parámetro bash	Parámetro PowerShell	Descripción
-n / --nombre	-nombre	Nombre/s de los países a buscar.
-t / --ttl	-ttl	Tiempo en segundos que se guardaran los resultados en caché