Solución al examen del TEMA 6 BASH

EJERCICIO 1

(3 puntos)

Crear un script que nos diga que abono transporte nos corresponde teniendo en cuenta la edad que tenemos. Para ello, le proporcionaremos la edad como parámetro y automáticamente mostrará un mensaje indicando el tipo de abono que nos corresponde. Existen los siguientes abonos:

- **Exentos de pago** los menores de 6 años
- **Tarifa infantil** para los niños con edades comprendidas entre los 7 y 14 años.
- **Tarifa joven** para las edades comprendidas entre los 15 y los 25 años.
- **Tarifa general** para las edades comprendidas entre los 26 y los 62 años.
- **Tarifa de jubilado/a** para los mayores de 62 años.

NOTAS:

- 0,5 puntos de la calificación se dedicarán a valorar buenas prácticas de programación: uso de comentarios, nombres de variables identificativos, indentación, estructura del código, etc...

SOLUCIÓN

```
#!/bin/bash

edad=$1

if [[ $edad -lt 6 ]]; then
echo "Eres exento de pago."
elif [[ $edad -ge 7 && $edad -le 14 ]]; then
echo "Tienes tarifa infantil."
elif [[ $edad -ge 15 && $edad -le 25 ]]; then
echo "Tienes tarifa joven."
elif [[ $edad -ge 26 && $edad -le 62 ]]; then
echo "Tienes tarifa general."
else
echo "Tienes tarifa de jubilado/a."
fi
```

EXPLICACIÓN

- El primer comando #!/bin/bash indica que el script debe ser interpretado por el shell de bash.
- Luego, definimos la variable edad y le asignamos el valor que se pasa como primer parámetro al script ($1).

- El if verifica la edad para determinar qué tipo de abono corresponde, utilizando la sintaxis de comparación de números en bash con los operadores -lt (menor que), -le (menor o igual que), -ge (mayor o igual que) y -gt (mayor que).
- Dependiendo de la edad, se muestra un mensaje en pantalla indicando el tipo de abono correspondiente utilizando el comando ec

EJERCICIO 2

(4 puntos)

Crear un script para la gestión de un libro de calificaciones. Debe aparecer un menú que nos permita añadir calificaciones, buscar la calificación de un alumno concreto y calcular la media de las calificaciones. No deben perderse los datos entre ejecuciones. El comportamiento de cada opción del script debe ser la siguiente:

- **Añadir calificación:** Pedirá el nombre del alumno y la calificación procediendo a guardar los datos.
- **Consultar calificación** Pedirá el nombre del alumno/a del cual se desea consultar la calificación y procederá a mostrar la calificación almacenada para dicho alumno/a.
- **Consultar media** Calculará la media de las calificaciones almacenadas y las mostrará por pantalla.

NOTAS:

- Se valorará positivamente el uso de funciones y el uso de las estructuras vistas para la creación de menús.
- 0,5 puntos de la calificación se dedicarán a valorar buenas prácticas de programación: uso de comentarios, nombres de variables identificativos, indentación, estructura del código, etc...

SOLUCIÓN - con fichero

```
#!/bin/bash

# Nombre del archivo donde se almacenarán las calificaciones
archivo_calificaciones="calificaciones.txt"

# Función para añadir una calificación
function añadir_calificacion {
read -p "Nombre del alumno/a: " nombre_alumno
read -p "Calificación: " calificacion

# Guardar los datos en el archivo de calificaciones
echo "$nombre_alumno:$calificacion" >> "$archivo_calificaciones"

echo "Calificación añadida correctamente."
}

# Función para consultar la calificación de un alumno/a
function consultar_calificacion {
read -p "Nombre del alumno/a: " nombre_alumno

# Buscar la calificación del alumno/a en el archivo de calificaciones
calificacion=$(grep "^$nombre_alumno:" "$archivo_calificaciones" | cut -d':' -
```

```
f2)

    if [[ -z $calificacion ]]; then
        echo "No se encontró la calificación del alumno/a $nombre_alumno."
    else
        echo "La calificación de $nombre_alumno es: $calificacion"
    fi
    }

    # Función para calcular la media de las calificaciones
    function calcular_media {
    # Leer las calificaciones del archivo y calcular la media
    media=$(awk -F':' '{ total += $2; count++ } END { print total/count }'
"$archivo_calificaciones")

    echo "La media de las calificaciones es: $media"
    }

    # Menú principal
    while true; do
    echo "1. Añadir calificación"
    echo "2. Consultar calificación"
    echo "3. Consultar media"
    echo "4. Salir"

    read opcion

    case $opcion in
        1)
        añadir_calificacion
        ;;
        2)
        consultar_calificacion
        ;;
        3)
        calcular_media
        ;;
        4)
        exit 0
        ;;
        *)
        echo "Opción inválida."
        ;;
    esac
    done
```

EXPLICACIÓN - con fichero

- El primer comando #!/bin/bash indica que el script debe ser interpretado por el shell de bash.
- Definimos la variable archivo_calificaciones que contiene el nombre del archivo donde se almacenarán las calificaciones.

- Definimos tres funciones:
  - añadir_calificacion: pide el nombre del alumno/a y la calificación, y los guarda en el archivo de calificaciones.
  - consultar_calificacion: pide el nombre del alumno/a y busca su calificación en el archivo de calificaciones.
  - calcular_media: lee las calificaciones del archivo y calcula la media.
- En el menú principal, se muestra un menú con las opciones disponibles y se lee la opción elegida por el usuario.
- Dependiendo de la opción elegida, se llama a la función correspondiente.

SOLUCIÓN - arrays asociativo

```
#!/bin/bash

# Inicializar la variable para almacenar las calificaciones
declare -A calificaciones

# Función para añadir una calificación
function añadir_calificacion {
read -p "Nombre del alumno/a: " nombre_alumno
read -p "Calificación: " calificacion

# Guardar la calificación en el array
calificaciones[$nombre_alumno]=$calificacion

echo "Calificación añadida correctamente."
}

# Función para consultar la calificación de un alumno/a
function consultar_calificacion {
read -p "Nombre del alumno/a: " nombre_alumno

# Buscar la calificación del alumno/a en el array
calificacion=${calificaciones[$nombre_alumno]}

if [[ -z $calificacion ]]; then
    echo "No se encontró la calificación del alumno/a $nombre_alumno."
else
    echo "La calificación de $nombre_alumno es: $calificacion"
fi
}

# Función para calcular la media de las calificaciones
function calcular_media {
# Calcular la media de las calificaciones
total=0
count=0
for calificacion in "${calificaciones[@]}"; do
    total=$((total + calificacion))
    count=$((count + 1))
```

```
done

media=$((total/count))

echo "La media de las calificaciones es: $media"
}

# Menú principal
while true; do
echo "1. Añadir calificación"
echo "2. Consultar calificación"
echo "3. Consultar media"
echo "4. Salir"

read opcion

case $opcion in
    1)
    añadir_calificacion
    ;;
    2)
    consultar_calificacion
    ;;
    3)
    calcular_media
    ;;
    4)
    exit 0
    ;;
    *)
    echo "Opción inválida."
    ;;
esac
done
```

EXPLICACIÓN - arrays asociativos

El script utiliza un array asociativo (declare -A) para almacenar las calificaciones y proporciona tres funciones principales: añadir_calificacion, consultar_calificacion y calcular_media.

La función añadir_calificacion permite al usuario ingresar el nombre del estudiante y su calificación, que luego se almacena en el array asociativo calificaciones utilizando el nombre del estudiante como clave. La función consultar_calificacion permite al usuario ingresar el nombre del estudiante y muestra la calificación correspondiente si está disponible en el array asociativo. La función calcular_media calcula y muestra la media de todas las calificaciones almacenadas en el array asociativo.

El programa utiliza un bucle while para mostrar un menú de opciones al usuario (añadir calificación, consultar calificación, consultar media o salir) y utiliza un case statement para ejecutar la opción elegida por el usuario. Si el usuario ingresa una opción inválida, se muestra un mensaje de error.

En general, este script es una forma simple de mantener un registro de las calificaciones de los estudiantes utilizando arrays asociativos en shell. Sin embargo, no es muy escalable y no maneja casos de error o excepciones, por lo que podría mejorar su robustez si se utiliza en una situación más crítica.

EJERCICIO 3

(3 puntos)

Crear un script para la creación automática de usuarios proporcionados en un fichero. El fichero contendrá los nombres de los usuarios a crear y su ruta se proporcionará como parámetro. Antes de la creación de un usuario se comprobará si el usuario ya existe, de ser así, se indicará con un mensaje y no se realizará la creación del usuario. Sin embargo, se comprobará si tiene como shell /bin/bash y si no es así se modificará y se pondrá como shell por defecto /bin/bash. Los usuarios deben crearse con el directorio home por defecto y con la shell /bin/bash.

NOTAS:

- Ejemplo para la creación de un usuario denominado “ejemplo” con shell /bin/bash y con directorio por defecto /home/ejemplo:

```
useradd -m -s /bin/bash ejemplo
```

- Los datos de los usuarios del sistema se guardan en el archivo /etc/passwd.
- 0,5 puntos de la calificación se dedicarán a valorar buenas prácticas de programación: uso de comentarios, nombres de variables identificativos, indentación, estructura del código, etc...

SOLUCIÓN

```
#!/bin/bash

# Comprobar si se proporciona un argumento
if [ -z "$1" ]; then
echo "Se debe proporcionar la ruta del fichero con los nombres de usuarios."
exit 1
fi

# Comprobar si el fichero existe
if [ ! -f "$1" ]; then
echo "El fichero $1 no existe."
exit 1
fi

# Iterar sobre los nombres de usuario en el fichero
while IFS= read -r username; do
# Comprobar si el usuario ya existe
if id -u "$username" >/dev/null 2>&1; then
    echo "El usuario $username ya existe."
```

```
    else
        # Crear el usuario con la shell /bin/bash y directorio home por defecto
        useradd -m -s /bin/bash "$username"
        echo "Se ha creado el usuario $username."

        # Comprobar si la shell es /bin/bash y, en caso contrario, modificarla
        if [ "$(getent passwd "$username" | cut -d':' -f7)" != "/bin/bash" ]; then
        chsh -s /bin/bash "$username"
        echo "Se ha cambiado la shell del usuario $username a /bin/bash."
        fi
    fi
    done < "$1"
```

¿Cómo ejecutar el script?

```
./crear_usuarios.sh /ruta/al/fichero.txt
```