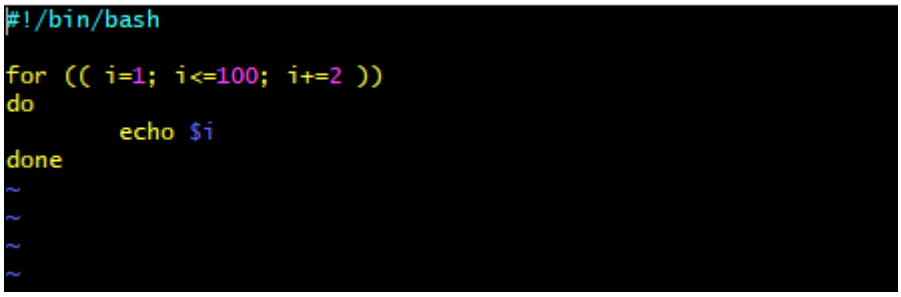
Ejercicio 1

MINGW64:/c/Users/paberrueco/Desktop/DATOS/UTAD/SSII/Und6/actividad



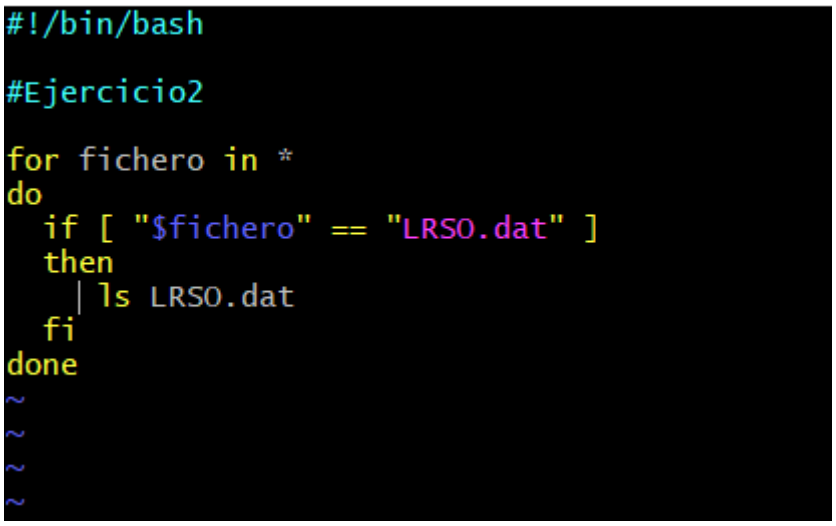
```
#!/bin/bash

for (( i=1; i<=100; i+=2 ))
do
        echo $i
done
~
~
~
~
```

El bucle for se utiliza para repetir un conjunto de comandos para cada valor de una variable determinada. En este caso, la variable $i se inicializa en 1 y se incrementa en 2 en cada iteración del bucle. Esto significa que el bucle se ejecutará para todos los números impares del 1 al 100.

Dentro del cuerpo del bucle, el comando echo $i se utiliza para imprimir el valor actual de la variable $i, que representa los números impares en cada iteración.

Ejercicio 2

MINGW64:/c/Users/paberrueco/Desktop/DATOS/UTAD/SS



```
#!/bin/bash

#Ejercicio2

for fichero in *
do
  if [ "$fichero" == "LRSO.dat" ]
  then
    | ls LRSO.dat
  fi
done
~
~
~
~
```

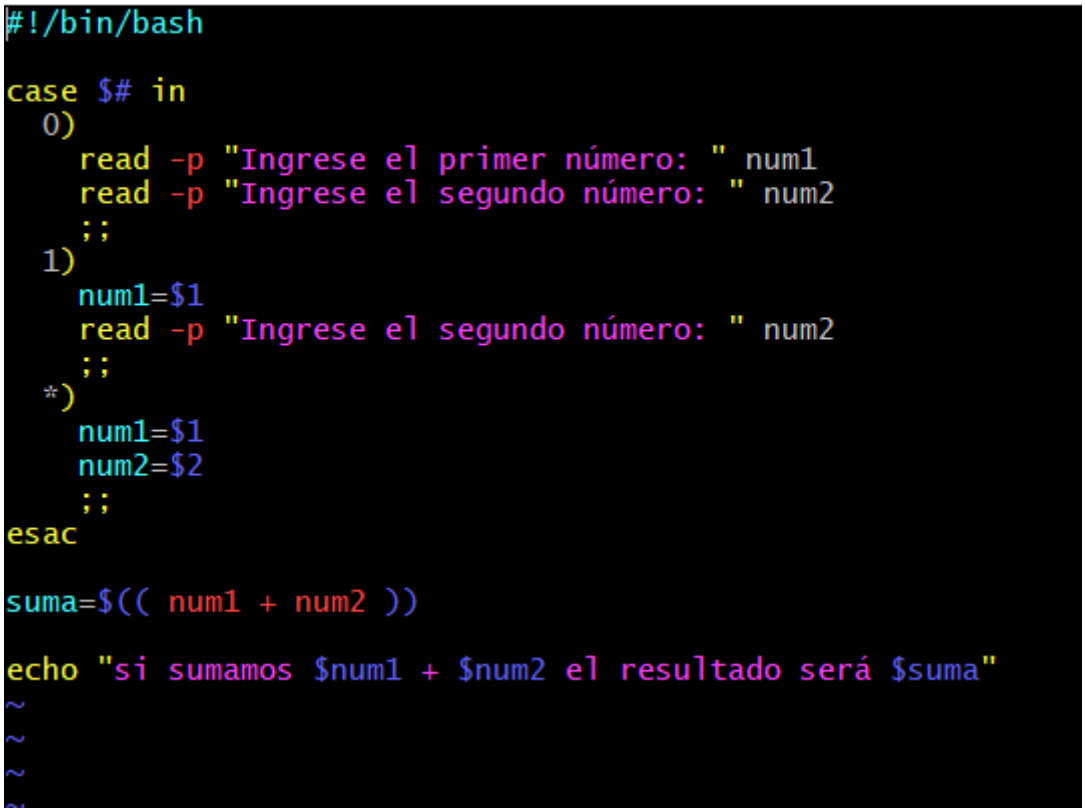
Este script busca un archivo llamado LRSO.dat en el directorio actual y lo muestra utilizando el comando “ls”

El script utiliza un bucle “for” que itera a través de todos los archivos y directorios en el directorio actual. La variable fichero se utiliza para almacenar el nombre de cada archivo o directorio a medida que se procesa.

Dentro del bucle, se utiliza una estructura de control if para verificar si el nombre del archivo actual es igual a LRSO.dat. Si es así, se ejecuta el comando ls LRSO.dat para mostrar información sobre el archivo.

Ejercicio 3

MINGW64:/c/Users/paberrueco/Desktop/DATOS/UTAD/SSII/Und6/actividad



```
#!/bin/bash

case $# in
  0)
    read -p "Ingrese el primer número: " num1
    read -p "Ingrese el segundo número: " num2
    ;;
  1)
    num1=$1
    read -p "Ingrese el segundo número: " num2
    ;;
  *)
    num1=$1
    num2=$2
    ;;
esac

suma=$(( num1 + num2 ))

echo "si sumamos $num1 + $num2 el resultado será $suma"
~
~
~
```

El script espera recibir uno o dos argumentos de línea de comando, que representan dos números enteros que se sumarán. El script realiza las siguientes acciones:

El comando case evalúa el número de argumentos que se pasan al script ($#).

Si no se pasan argumentos (case 0), el script solicita los números al usuario utilizando el comando read.

Si se pasa un argumento (case 1), el script asigna ese valor a la variable num1 y solicita el segundo número al usuario con el comando read.

Si se pasan dos argumentos o más (case *), el script asigna el primer argumento a la variable num1 y el segundo argumento a la variable num2.

El script realiza la suma de num1 y num2 y lo asigna a la variable suma.

Por último, el script imprime un mensaje que indica los números que se sumaron y el resultado de la suma.

Por lo tanto, este script permite al usuario especificar uno o dos números para sumar y produce un resultado que indica los números y el resultado de la suma.

Ejercicio 4

MINGW64:/c/Users/paberrueco/Desktop/DATOS/UTAD/SSII/Und6/actividad

```
#!/bin/bash

while true
do
    # Mostramos el menú de selección de operación
    echo "Seleccione una operación:"
    echo "1 ) Potenciación:"
    echo "2 ) Conversión de grados centígrados a fahrenheit:"
    echo ""

    # Leemos la opción del usuario
    read opcion

    # Evaluamos la opción seleccionada por el usuario
    case $opcion in
        1)
            # Pedimos la base y el exponente al usuario
            echo "Introduce la base:"
            read base

            echo "Introduce el exponente:"
            read exponente

            # Comprobamos si los parámetros son numéricos
            if [[ ! "$base" =~ ^[0-9]+$ || ! "$exponente" =~ ^[0-9]+$ ]]; then
                echo "Error: los parámetros deben ser numéricos"
                echo ""
                continue
            fi

            # Realizamos la operación de potenciación con un bucle for
            resultado=1
            for ((i=1; i<=exponente; i++))
            do
                resultado=$((resultado * base))
            done

            echo "$base elevado a $exponente es igual a $resultado"
            echo ""
            ;;
        2)
            # Pedimos los grados centígrados al usuario
            echo "Introduce los grados centígrados:"
            read temperatura

            # Comprobamos si el parámetro es numérico
            if [[ ! "$temperatura" =~ ^[0-9]+$ ]]; then
                echo "Error: el parámetro debe ser numérico"
                echo ""
                continue
            fi

            # Realizamos la conversión a fahrenheit
            fahrenheit=$((18 * gradosC / 10 + 32))

            echo "$temperatura grados centígrados son $fahrenheit grados fahrenheit"
            echo ""
            ;;
        *)
            echo "Error: opción no válida"
            echo ""
            ;;
    esac
done
```

ejercicio4.sh [unix] (23:40 12/03/2023)

El script está dentro de un bucle infinito, lo que significa que el menú se mostrará repetidamente hasta que el usuario decida salir. Dentro del bucle, el script muestra el menú de opciones y lee la selección del usuario usando el comando read. A continuación, el script utiliza un comando case para evaluar la selección del usuario y llevar a cabo la operación correspondiente.

Si el usuario elige la opción de potenciación, el script pide al usuario que introduzca la base y el exponente. Luego, el script utiliza un bucle for para calcular la potencia y muestra el resultado al usuario.

Si el usuario elige la opción de conversión de grados centígrados a Fahrenheit, el script pide al usuario que introduzca la temperatura en grados centígrados. Luego, el script realiza la conversión a grados Fahrenheit y muestra el resultado al usuario.

Si el usuario elige una opción inválida, el script muestra un mensaje de error y vuelve a mostrar el menú de selección de operaciones.

El bucle infinito se ejecutará hasta que el usuario decida salir del programa manualmente mediante un comando de interrupción (por ejemplo, presionando Ctrl+C).

Ejercicio 5

MINGW64:/c/Users/paberrueco/Desktop/DATOS/UTAD/SSII/Und6/actividad

```
#!/bin/bash

if [ $# -ne 1 ]; then
  echo "Error debe introducir un paramatro detras de $0"
  exit 1
fi

filename=$1

if [ -e "$filename" ]; then
  echo "El fichero $filename ya existe en el directorio actual"
else
  echo "El fichero $filename no existe en el directorio actual. Se creará y se introducirán 10 líneas con números del 1 al 10."
  for i in {1..10}; do
    echo $i >> $filename
  done
fi
~
~
~
~
```

Este script verifica si se proporcionó un argumento en la línea de comando. Si se proporciona un argumento, el script verifica si el archivo con el nombre proporcionado existe en el directorio actual. Si existe, el script muestra un mensaje indicando que el archivo ya existe. Si no existe, el script crea el archivo y escribe 10 líneas con números del 1 al 10.

La línea if [$# -ne 1]; then verifica si el número de argumentos pasados en la línea de comando es diferente de 1. Si es así, muestra un mensaje de error y sale del script con un código de error 1.

La variable filename se establece en el primer argumento proporcionado en la línea de comando.

La línea if [-e "$filename"]; then verifica si el archivo con el nombre proporcionado en la variable filename existe en el directorio actual. Si el archivo existe, el script muestra un mensaje indicando que el archivo ya existe.

Si el archivo no existe, el script muestra un mensaje indicando que el archivo será creado y escribe 10 líneas con números del 1 al 10 en el archivo.

El bucle for se utiliza para escribir las 10 líneas en el archivo. Cada iteración del bucle escribe un número del 1 al 10 en una nueva línea del archivo. La sintaxis {1..10} genera una secuencia de números del 1 al 10 que se utiliza en el bucle.

Ejercicio6

MINGW64:/c/Users/paberrueco/Desktop/DATOS/UTAD/SSII/Und6/acti

```
#!/bin/bash

for ((num=1; num<=100; num++))
do
    # Si el número es menor o igual a 1, no es primo
    if [[ $num -le 1 ]]
    then
        echo "$num no es primo"
    else
        # Verificar si el número es primo
        primo=true
        for ((i=2; i<num; i++))
        do
            if [[ $((num%i)) -eq 0 ]]
            then
                primo=false
                break
            fi
        done

        # Imprimir el resultado
        if [[ $primo == true ]]
        then
            echo "$num es primo"
        else
            echo "$num no es primo"
        fi
    fi
done
~
~
~
```

Este script muestra los números del 1 al 100 e indica cuáles son primos y cuáles no. La lógica es la siguiente:

Se utiliza un bucle for para iterar desde 1 hasta 100, asignando cada número a la variable "num".

Se verifica si "num" es menor o igual a 1. En ese caso, se imprime que no es primo.

Si "num" es mayor que 1, se verifica si es primo. Se establece una variable "primo" en true y se utiliza otro bucle for para iterar desde 2 hasta "num"-1. Dentro de este bucle for, se verifica si "num" es divisible por "i". Si es así, se establece "primo" en false y se rompe el bucle for interno con el comando "break".

Después de salir del bucle for interno, se imprime si "num" es primo o no, según el valor de la variable "primo". Si "primo" es true, se imprime que "num" es primo. Si es false, se imprime que no lo es.

Ejercicio7

MINGW64:/c/Users/paberrueco/Desktop/DATOS/UTAD/SSII/Und6/actividad

```
#!/bin/bash

# Verificar si se ha proporcionado un parámetro
if [[ $# -ne 1 ]]
then
    # Mostrar calendario del mes actual
    cal
elif [[ $1 == "c" || $1 == "corta" ]]
then
    fecha=$(date +"%d/%m/%Y")
    echo "$fecha"
elif [[ $1 == "l" || $1 == "larga" ]]
then
    dia=$(date +"%d")
    mes=$(date +"%m")
    ano=$(date +"%Y")
    echo "Hoy es el día '$dia' del mes '$mes' del año '$ano'."
else
    echo "Opción incorrecta."
    exit 2
fi
~
~
~
~
~
~
```

Este script verifica si se proporciona un parámetro y, dependiendo del parámetro, realiza una de tres operaciones:

Si no se proporciona ningún parámetro, se muestra el calendario del mes actual utilizando el comando cal.

Si el parámetro es "c" o "corta", se muestra la fecha actual en formato corto utilizando el comando date con la opción +"%d/%m/%Y".

Si el parámetro es "l" o "larga", se muestra la fecha actual en formato largo utilizando el comando date con las opciones +"%d", +"%m", y +"%Y" para obtener el día, mes y año, respectivamente.

Si se proporciona un parámetro que no es válido, el script muestra un mensaje de error y sale con un código de salida 2.