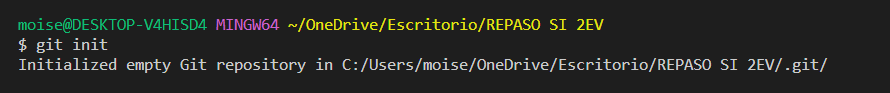
**CONTROL DE VERSIONES CON GIT**

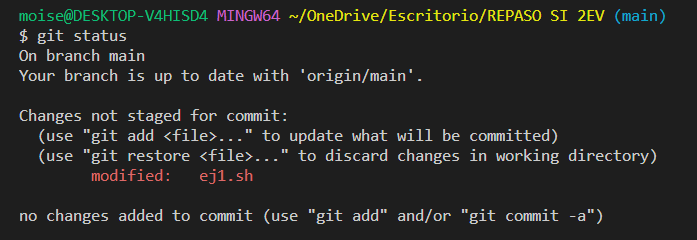
**CREA UN REPOSITORIO NUEVO:** *git init*



**COPIA DE REPOSITORIO:** *git clone <url-repositorio>*

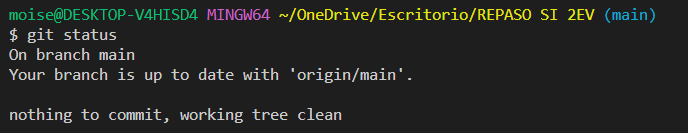
Con este comando bajaremos desde github a vs code el proyecto tal y como este para seguir trabajando.

**MOSTRAR EL ESTADO DE UN REPOSITORIO:** *git status*

****

Aqui he puesto algo nuevo en el ej1.sh y como no he commiteado con un *git status*

puedes ver si has hecho algo para guardar los cambios o no. sino saldria esto:

****

Está todo en orden.

**SHELL SCRIPTS**

**La primera línea del script tiene que tener #!/bin/bash**

**Ejecutándolo con el comando bash o con el comando sh.**

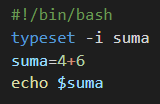
**Comando *touch*: crearemos un nuevo fichero de texto plano.**

**Variables:**

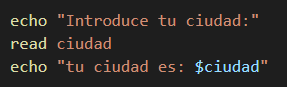
**se declaran así:** var1=5 **para llamarla con el $ antes del nombre echo** $var1**.**

**para borrar una variable:** unset var1**.**

**Para operar con números usamos** *typeset* -i **y la variable que queremos convertir a tipo numérico.**

****

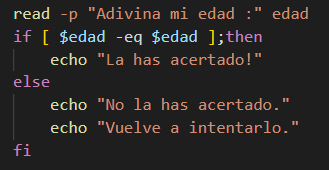
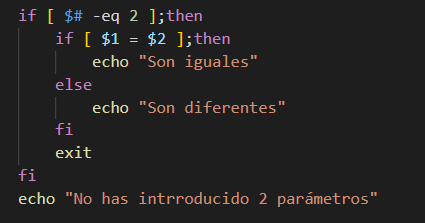
**El comando** *read* **nos permite crear una nueva variable con el valor introducido por la entrada estándar (teclado)**

****

**solicitar un valor, leerlo y almacenarlo en una variable en un solo comando, mediante** *read –p*

**

**Condicional if-then-else:**

**Funciona como otros lenguajes de programación solo que acaba en fi y tiene que tener espacios [ CONDICIÓN ]**

**Al comparar dos valores enteros en una [ CONDICIÓN ], utilizaremos los flags:**

*-eq (is equal to)*

****

*-gt (is greater than)*

****

*-ge (is greater than or equal to)*

****

*-lt (is less than)*

****

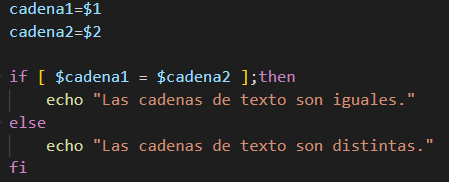
*-le (is less than or equal to)*

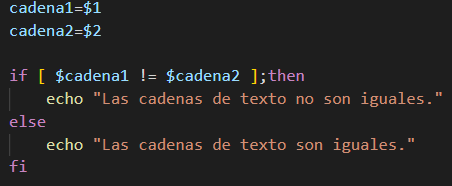
****

*-ne (is not equal to)*

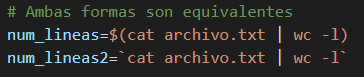
****

**Podemos comparar las cadenas de caracteres (strings) con** *=* **y** *!=*

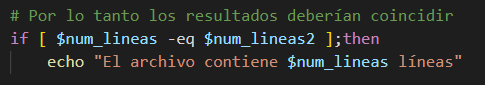
****



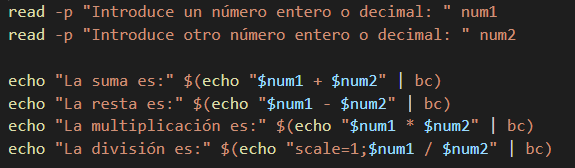
**si queremos almacenar su resultado, deberemos utilizar la nomenclatura $(COMANDO) o bien `COMANDO` (nótese la tilde invertida):**

****

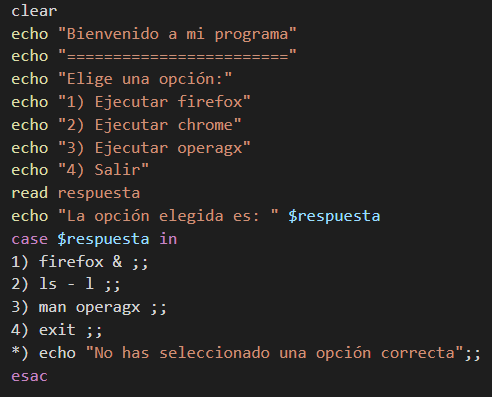
*cat* **sirve para mostrar un archivo de texto en la consola** *wc -l* **es para contar las líneas, palabras y caracteres que hay dentro de un archivo.**

****

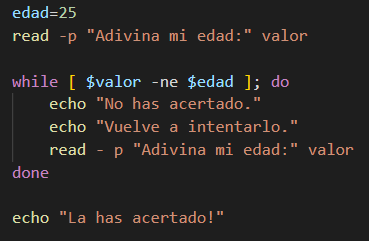
**Gracias al comando** *bc* **podemos hacer cualquier tipo de cálculo,** *ya sea con enteros o con decimales***, incluso especificando el número de decimales deseado mediante la variable** *scale***:**

****

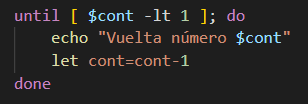
**Mediante un bloque** *case-esac* **podemos seleccionar entre varias alternativas de manera sencilla. Es equivalente al** *switch:*



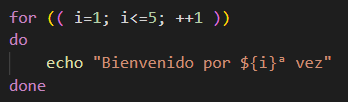
**Típico bucle condicional en el que ejecutaremos una serie de instrucciones mientras se cumpla la condición:**

****

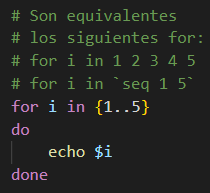
**Bucle condicional en el que ejecutaremos una serie de instrucciones hasta que se cumpla la condición**

****

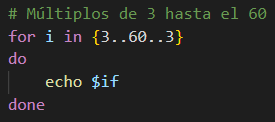
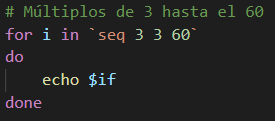
**Bucle condicional que, además de la condición, incluye la inicialización y el paso (normalmente un incremento o decremento)**

****

**Bucle incondicional, equivalente al *foreach* de otros lenguajes de programación, que sirve para recorrer una serie de valores**

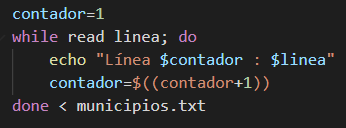


**Es posible recorrer el secuencial en pasos diferentes de 1, que es el paso por defecto, mediante** *`seq INICIO PASO FIN`* **o** *{INICIO..FIN..PASO} .*



**Hay diversas formas de procesar (recorrer) un fichero en shell script.**

**Podemos leerlo línea a línea proporcionando el fichero como entrada del bucle while y utilizando read en su condición**

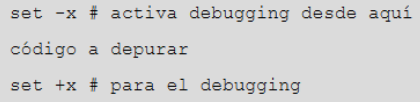


while read LINEA; do

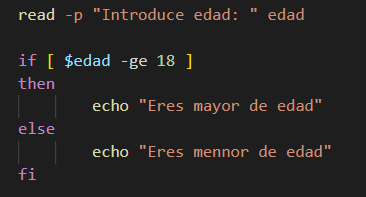
# hacer algo con la LINEA

done < FICHERO

**También podemos depurar una sección de código incluyendo:**

****

**Podemos indicar if [ CONDICIÓN ];then en la misma línea o bien omitir el ; y poner then en la línea siguiente. Esto también se aplica a while-do, until-do, etc.**

****

**Podemos emular un bloque if-then-else mediante los operadores && (and) y || (or), de la siguiente manera:**

