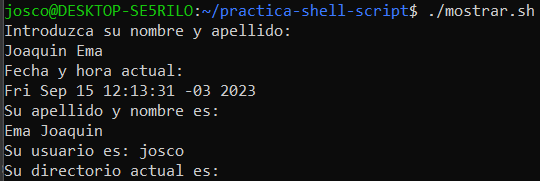
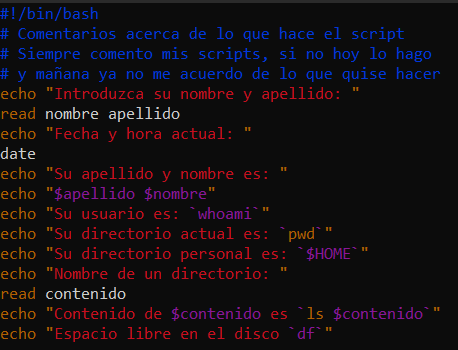
mkdir practica-shell-script -v

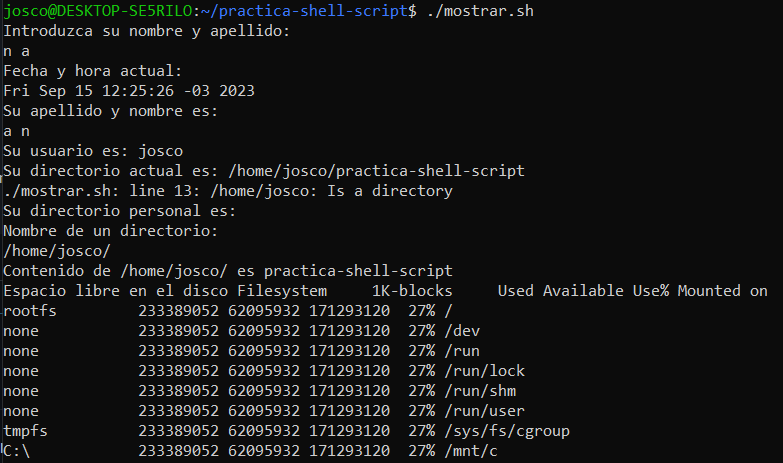
cd practica-shell-script

touch mostrar.sh

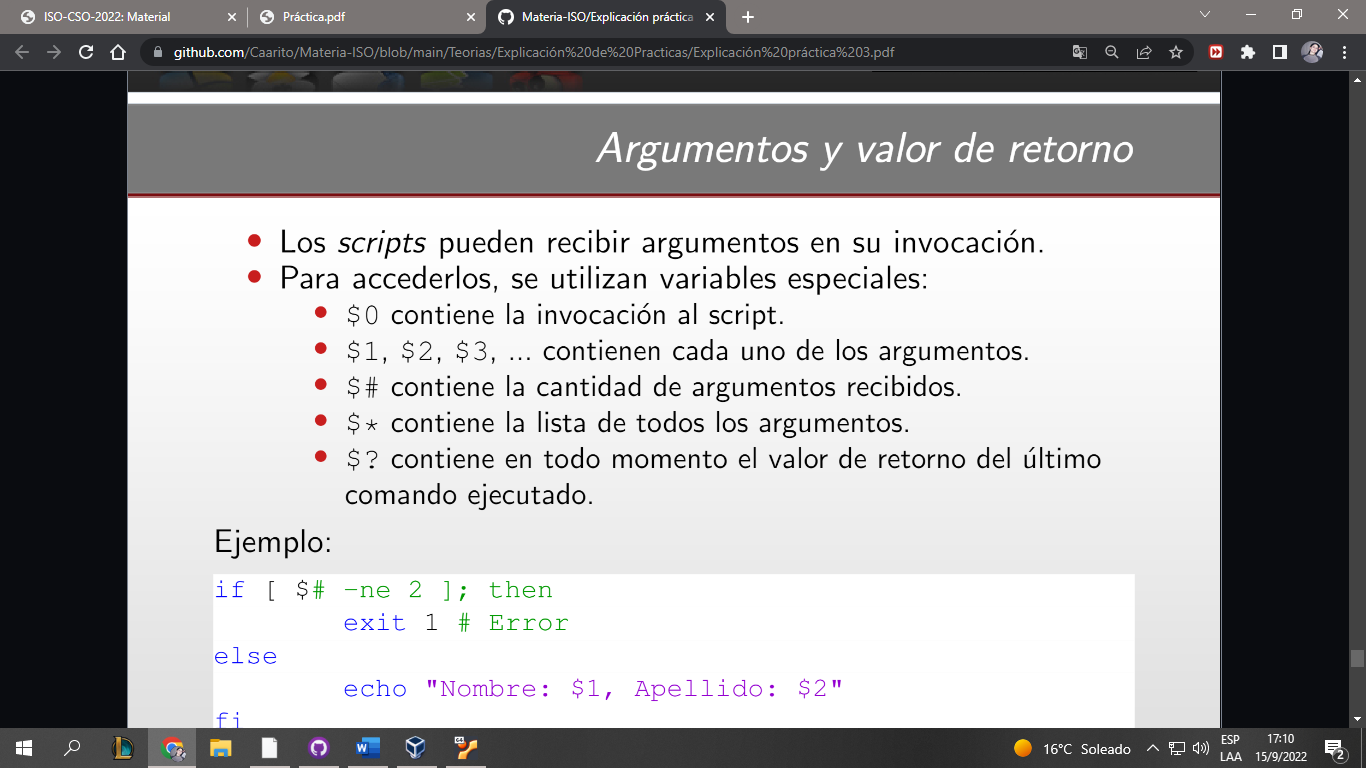
vim mostrar.sh # i = modo inserción, Esc = volver al modo comando, :wq = cerrar y guardad

1. chmod u+x mostrar.sh -c -v
3. 
4. Son para poder ejecutar comando cuando se encuentran dentro de un string









1. El comando exit finaliza un script. Incluso, puede arrojar un valor de terminación, el cual estará disponible para su proceso padre.

Todo comando posee un estado de terminación (exit status). Un comando que se ejecutó satisfactoriamente arrojará un valor de 0, mientras que un comando que no se procesó de manera correcta, finalizará con un número entre 1 y 255.



* Se puede utilizar
  + suma
  + resta
  + multiplicación (/\*)
  + división (/).
* Y devuelve solo la parte entera.:
  + expr 10 + 10 -> 20
  + expr 14 % 9 -> 5

También se puede utilizar para:

* Cálculo de la longitud de la cadena:
  + expr length “esto es un test” >
  + devuelve la cantidad de caracteres: 15
* Cadena de rastreo:
  + expr substr “esto es un test” 3 5 >
  + te devuelve el texto desde la pos 3 y 5 lugares: to es
* La cadena numérica primera posición de rastreo personaje que aparece:
  + expr index “esto es un test” s
  + te devuelve el índice de la letra pasado por parámetro en la cadena (la primera aparición).

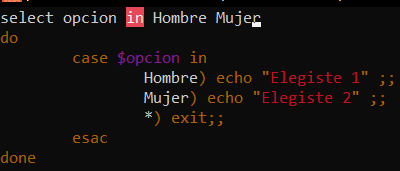


|  |
| --- |
| #!/bin/bash  var1=10  var2=5  if [ -e 7.sh ]; then echo "existe"; else echo "no existe"; fi  if (( $var1 % $var2 == 0 )); then echo "Es Par"; else echo "Es Impar"; fi  if (( $var1 > $var2 )); then echo "Es Mayor"; else echo "Es Menor"; fi  if [ -r 7.sh -a -w 7.sh -a -x 7.sh ]; then echo "Todos los permisos"  else echo "no tiene todos los permisos"; fi  if [ -r 7.sh -o -w 7.sh -o -x 7.sh ]; then echo "algun permiso"  else echo "no tiene todos los permisos"; fi |

| **Parámetro** | **Valor** |
| --- | --- |
| -r | Si es de Lectura |
| -w | Si es de escritura |
| -x | Si es de ejecución |
| -a | and |
| -o | or |
| -e | Si existe |
| -d | Si es un directorio |
| -f | Si es un archivo |
| -n | No es cadena vacia |

| **Operador** | **Con Strings** | **Con números** | **Operaciones** |
| --- | --- | --- | --- |
| Igualdad | [ $nombre = "Maria" ] | [ $edad -eq 20 ] | $(( 2 == 2 )) |
| Desigualdad | [ $nombre != "Maria" ] | [ $edad -ne 20 ] | $(( 3 != 4 )) |
| Mayor | [ A > Z ] | [ 5 -gt 20 ] | $(( 3 > 1 )) |
| Mayor o Igual | [ A >= Z ] | [ 5 -ge 20 ] | $( 2 >= 1 )) |
| Menor | [ A < Z ] | [ 5 -lt 20 ] | $(( 1 < 10 )) |
| Menor o Igual | [ A <= Z ] | [ 5 -le 20 ] | $(( 3 <= 5 )) |







* break [n] Corta la ejecución de n niveles de loops.
* continue [n] salta a la siguiente iteración del enésimo loop que contiene esta instrucción



bash soporta strings y arrays

* Los nombres son case sensitive
* Para crear una variable:
  + NOMBRE="pepe" SIN espacios alrededor del =
* Para accederla se usa $:
  + echo $NOMBRE

Para evitar ambigüedades se pueden usar llaves:

* Esto no accede a $NOMBRE
  + echo $NOMBREesto\_no\_es\_parte\_de\_la\_variable
* Esto sí
  + echo ${NOMBRE}esto\_no\_es\_parte\_de\_la\_variable

Los nombres de las variables pueden contener mayúsculas, minúsculas, números y el símbolo (underscore), pero no pueden empezar con un número.

No es fuertemente tipado, y no requiere una declaración explícita de tipo variables.

| **Bash** | **Resultado** |
| --- | --- |
| arreglo=() | Definir Arreglo vacio |
| arreglo[0]="valor1" | Asignación |
| arreglo=("valor1", "valor2") | Definir con valores |
| read -a arreglo | leer desde teclado tiene que estar definido arriba |
| ${arreglo[0]} | retorna "valor1" |
| ${arreglo[\*]} | Retorna todos los elementos del array |
| ${arreglo[@]} | Retorna todos los elementos del array |
| ${#arreglo[@]} | Retorna 2, si contiene valor1 y valor2 |

Si, se pueden definir funciones dentro de un script.

* Las funciones permiten modularizar el comportamiento de los scripts.
* Se pueden declarar de 2 formas:

|  |
| --- |
| function nombre { block }  nombre() { block } |

* Con la sentencia return se retorna un valor entre 0 y 255
* El valor de retorno se puede evaluar mediante la variable $?
* Reciben argumentos en las variables $1, $2, etc

Para definir variables locales podemos usar la palabra clave **g**