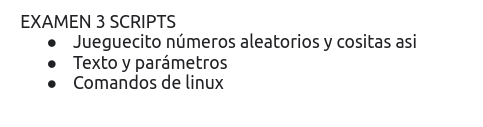
Admin.Sistemas

09/09/2022

Sistema operativo

* GUI
* CLI
  + Comandos y scripts
    - Linux→sh, **bash**, ksh...también pueden utilizar lenguajes de programación directamente en consola debido a que ya vienen instalado. Algunos ejemplos son Python, Ruby o PHP
    - Windows→ CMD (.bat y .cmd), wsh, **vbscript** (VisualBasicScript) y PowerShell



**#!/bin/bash**

**#!/usr/bin/python**

**chmod u+x fichero.sh**

# 

# 12/09/2022

**Organigrama:** representación gráfica de la estructura de una empresa o cualquier otra organización, que incluye las estructuras departamentales y, en algunos casos, las personas que las dirigen, hacen un esquema sobre las relaciones jerárquicas y competenciales de vigor.

**Ordinograma:** esquema gráfico que representa la secuencia de las operaciones y procesos que en la utilización de un ordenador forman un programa.

***nano ej01.sh***

**#!/bin/bash** (siempre al principio de cualquier script)

#Comentario

echo “Hola ASIR2” (muestra mensaje por pantalla)

**nano**

Ctrl+O->Guardar el contenido de fichero

Ctrl+X->Salir del fichero

bash +nombre del fichero–>ejecuta el fichero.

**chmod** u+x “fichero” para cambiar los permisos

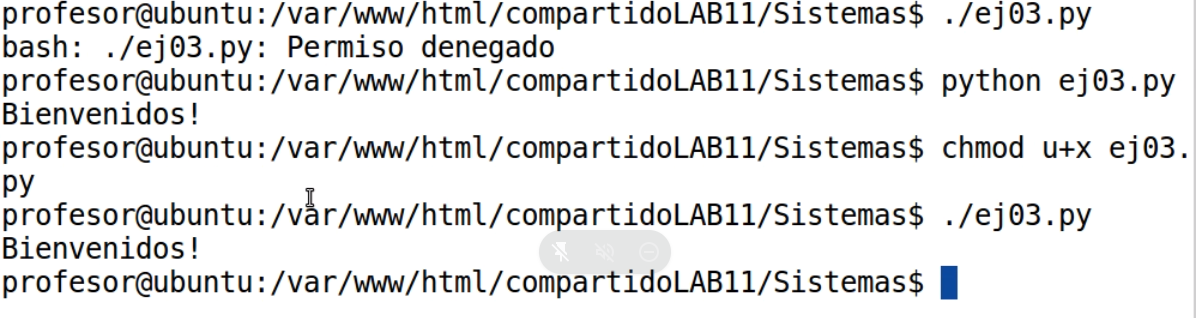
Con poner la ruta donde se encuentra el fichero también se ejecuta este gracias a los permisos dados en el anterior paso.

**#!/usr/bin/python-**>ruta de ficheros ejecutables con python

El comando **which** muestra dónde está la ruta.

which bash-> ruta de bash

El comando **file** nos muestra de que tipo es el fichero

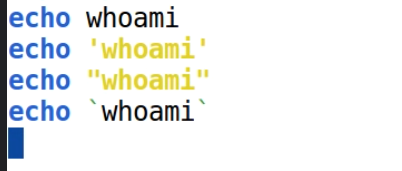


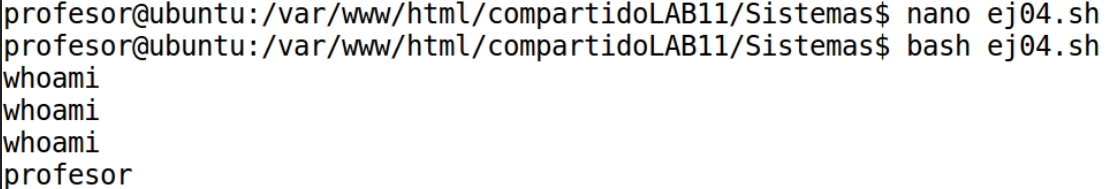
**read -p** —>Recibe y muestra los datos del fichero

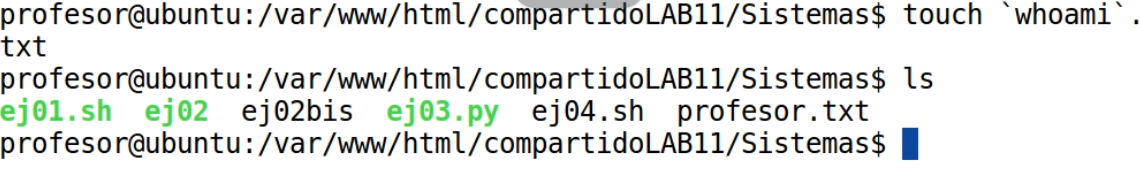
[Comando Read. Peticiones al usuario con scripting de Linux - gpsos.es](https://www.gpsos.es/2019/07/comando-read-interactua-con-el-usuario-en-bash/)

**echo -n**→Te muestra todo el contenido en la misma línea.

; para separar los comandos







# 13/09/2022

Las comillas invertidas primero ejecutan el propio comando y luego el resto.





**if** ¿decisión?

Parte Sí

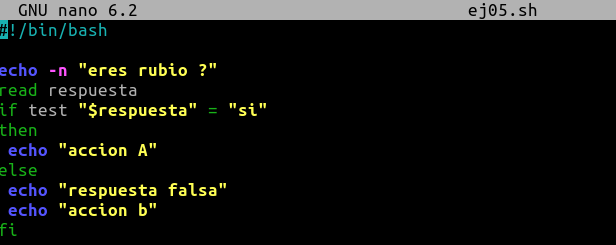
Acción A

Parte No

Acción B

Fin

***nano ej05.sh:***



—-----------------------------------------------------

#!/bin/bash

echo -n "Eres rubio (si/no)?"

read respuesta

if test $respuesta = "si"

then

echo "Accion A"

else

echo "Respuesta falsa"

echo "Accion B"

fi

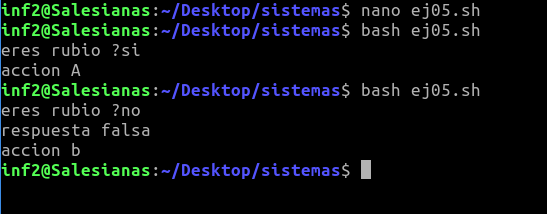
—-----------------------------------------------------

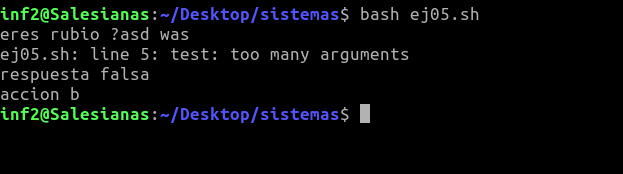
Ejecutar: bash ej05.sh

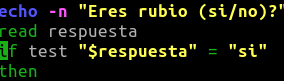
Linux diferencia mayúsculas de minúsculas ( si pongo SI va a la acción B)

**cat** para mostrar contenido del archivo.

**test** devuelve verdadero o falso

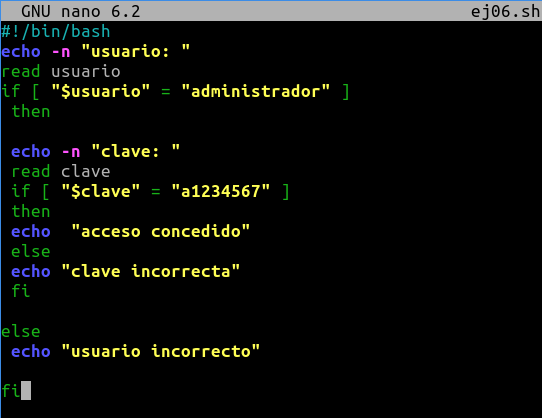






Si ponemos la variable entre comillas agrupará la respuesta que le des en un bloque conjunto aunque esta tenga espacios.

La ayuda del if es la ayuda de test (para ver lo de mayor, menor y mas condic.): **man test**

***nano ej06.sh:***

#!/bin/bash

echo -n “Usuario : “

red usuario

#opción 1 if test “$usuario” = “administrador”

#opción 2 if [ “usuario” = “administrador” ]

then

echo -n “Clave: “

red clave

if [ “$clave” = “ “a1234567” ]

then

echo “Acceso concedido”

esle

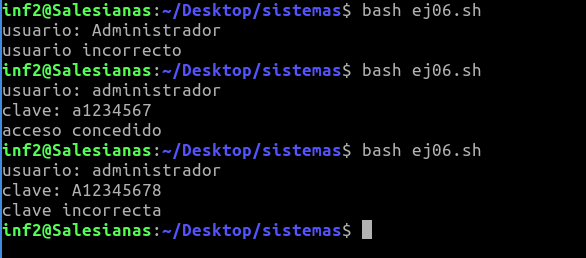
echo “Clave incorrecta”

else

echo “Usuario incorrecto”

fi

Ejecutar: bash ej06.sh



*Anotación: dejar espacios antes y después de los corchetes*

# 16/09/2022

***nano ej07.sh***

#!/bin/bash

echo "Te invito a mi fiesta de cumpleaños"

echo -n "Vas a asistir (s/n)?"

read asistencia

if [ $asistencia = "s" -o $asistencia = "S" ]

then

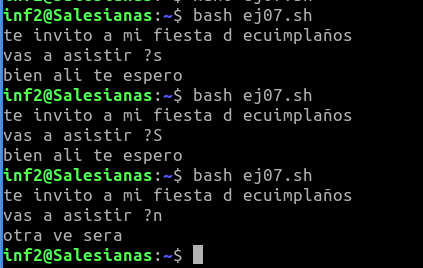
echo "bien,allí te espero"

elif [ $asistencia = “n” -o $asistencia = “ N” ]

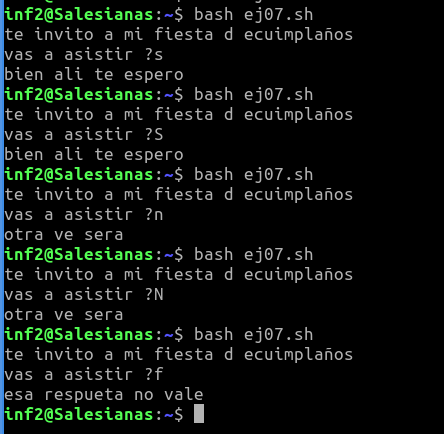
then

echo "Otra vez será"

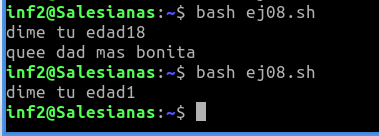
else Salida:

echo “Respuesta incorrecta”

fi



***nano ej08.sh***



*con* **-eq** *sí que acepta que metas por ejemplo en este caso 018, un 0 delante. (comparación numérica)*

*con “=” tienen que ser los caracteres exactos (comparación de caracteres)*

**COMPARACIONES NUMÉRICAS**

-eq->equal

-ne-> not equal

-gt->greater than

-ge->greater equal

-lt->less than

-le->less equal

**COMPARACIONES TEXTO**

=

!=

-n-> dice verdadero si tiene algo escrito

-z-> dice verdadero si está vacío

**COMPARACIONES FICHERO**

-r ->permiso lectura (se puede leer)

-w ->permiso escritura

-x ->si tiene el permiso de ejecución

-f ->fichero

-d ->directorio

-e ->existe

-s ->size: te dice si tiene tamaño

**NEGACIÓN**

!

**CONJUNCIÓN**

-a

**DISYUNCIÓN**

-o



***nano ej09.sh***

#!/bin/bash

read -p "Dime tu nombre: " nombre

if [ -z $nombre ]

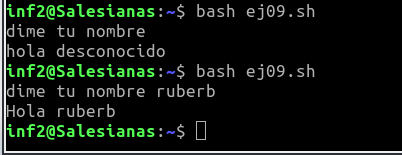
then

echo "Hola desconocido"

else

echo "Hola $nombre"

fi



***nano ej10.sh***

read -p "Introduce un fichero: " fichero

if [ -e $fichero ]

then

if [ -r $fichero ]

then

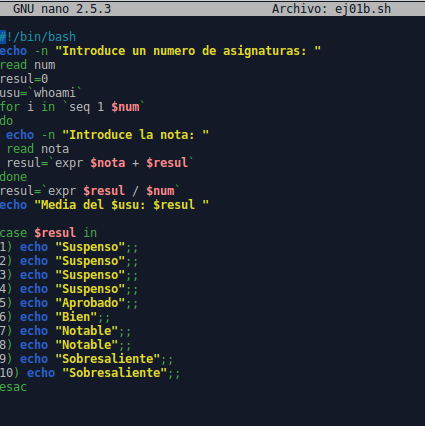
echo "El $fichero existe y se puede leer"

else

echo "No puedo leer el fichero $fichero"

fi

else

echo "Ese fichero no existe"

fi

*ES COMO UN SWITCH = CASE*

*nano ej11.sh*

#!/bin/bash

echo "que dia de la semana es hoy"

read dia

**case** $dia in Lunes) echo "Horario"

echo "Sistemas"

echo "BBDD"

echo "..." ;;

Martes) echo "Martes" ;;

Miercoles) echo "Miercoles" ;;

Jueves) echo "Jueves" ;;

Viernes | viernes) echo "Fiesta!";;

\*) echo "No has puesto un dia de la semana"

**esac**

**case** es para diferentes opciones

# 19/09/2022

**Explicación test**

$?->código de salida del último comando ejecutado, si es 0 funciona (verdadero) y si es 1 u otro número falla ( falso)

**BUCLES**

**while -> mientras que…**

**until -> hasta que…**

***nano ej12.sh***

#!/bin/bash

#situación inicial

flexiones=0

#bucle y condición de finalización

while [ $flexiones -lt 20 ]

do

#acciones

echo "Haciendo flexion..."

sleep 0.5 #pausa

#avance

flexiones=`**expr** $flexiones + 1`

done #cuando la condición ya es falsa salta aquí

*“no se pone* ***$*** *en la variable cuando me da igual que valor lleva, es cuando quiero guardarle algo”*

***expr ->*** *resuelve operaciones matemáticas, tienes que dejar espacios entre todos los caracteres, y no te da decimales. (para* ***multiplicar*** *añade* ***\\**** *y el módulo* ***% es el resto de la división entera****)*

**expr length** sol-> longitud de la palabra sol

↪3

**expr index** sol l -> me dice la posición de un carácter

↪3

**expr substr** cocodrilo 1 4-> coge un trozo de la cadena

↪coco

***nano ej13.sh***

#!/bin/bash

echo -n "Dime un palabra: "

read palabra

longitud=`expr length "$palabra"`

posicion=1

while [ $posicion -le $longitud ]

do

letra=`expr substr "$palabra" $posicion 1`

echo "$letra"

sleep 1

posicion=`expr $posicion + 1`

done

26/09/2022

**for**

***nano ej14.sh***

#!/bin/bash

num=1

while [ $num -le 5 ]

do

echo "$num"

num=`ex $num + 1`

done

echo ""

num=1

until [ $num -gt 5 ]

do

echo "$num"

num=`ex $num + 1`

done

***nano ej15.sh***

#!/bin/bash

#for variable in conjunto

#do

#hacer algo por cada elemento del conjunto expresado en varible

#done

**for** numero in 1 2 3 4 5

do

echo "$numero"

done

**for** dia in Lunes Martes "Miercoles Jueves" 1234 #entre comillas un elemento

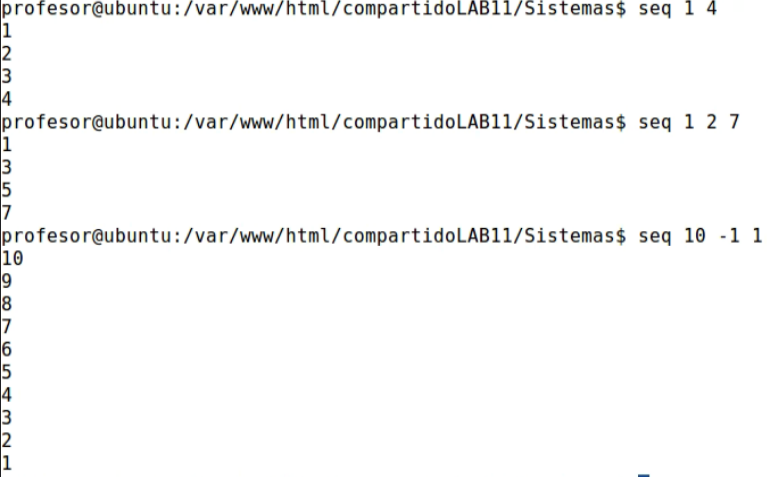
do

echo "$dia"

done

**Entre comillas invertidas <`comando`>**

**seq**



***nano ej16.sh***

#!/bin/bash

for numero in seq 0 0.5 3 #lo entiende como un elemento

do

echo "$numero"

done

echo ""

for numero in `seq 0 0.5 3` #por eso es importante comillas invertidas

do

sleep $numero #problema que la salida del seq es con coma por teclado español

#y sleep funciona con punto

echo "$numero"

done

***nano ej17dias.sh***

#!/bin/bash

echo "Lunes"

echo "Martes"

echo "Miercoles"

echo "Jueves"

echo "Viernes"

echo "Sabado"

echo "DOmingo"

***nano ej17.sh***

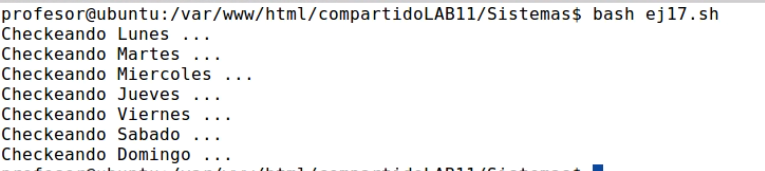
#!/bin/bash

for dia in `bash ej17dias.sh`

do

echo "Checkeando $dia ..."

done



**whoami**

**groups**

**cut**

***nano ej18.sh***

#!/bin/bash

usuario= `whoami`

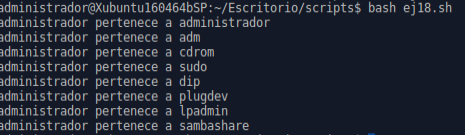
grupos=`groups $usuario | cut -d ":" -f 2` #para filtrar solo esa parte

for item in $grupos #puede tener muchos elementos esta variable

do

echo "$usuario pertenece a $item"

done

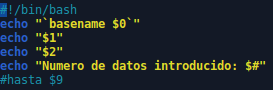


Nºs aleatorios=> variable **$RANDOM**

alea=`expr $RANDOM % 10 (RANGO) + 1 (INICIAL)

3/10/2022 - Parámetros

***nano ej20.sh***

#!/bin/bash

echo "`basename $0`"

echo "$1"

echo "$2"

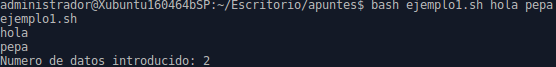
echo "Numero de datos introducido: $#"

#hasta $9

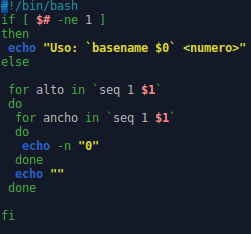
Guarda como en variables lo que introduces al ejecutar

**$0** nos dice el nombre del archivo pero puede incluir la ruta por eso usas **basename** que se queda solo con el nombre

**$#** nos dice el nº de variables introducidas



***nano ej21.sh***



#!/bin/bash

if [ $# -ne 1 ]

then

echo "Uso: `basename $0` <numero>"

else

for alto in `seq 1 $1`

do

for ancho in `seq 1 $1`

do

echo -n "0"

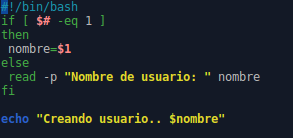
done

echo ""

done

fi

***nano ej22.sh***

#!/bin/bash

if [ $# -eq 1 ]

then

nombre=$1

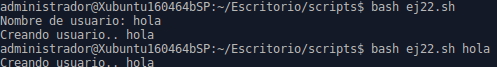
else

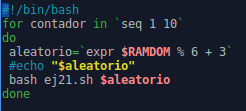
read -p "Nombre de usuario: " $nombre

fi

echo "Creando usuario.. $nombre"

Lo que hace:



***nano ej23.sh***

#!/bin/bash

for contador in `seq 1 10`

do

aleatorio=`expr **$RAMDOM** % 6 + 3`

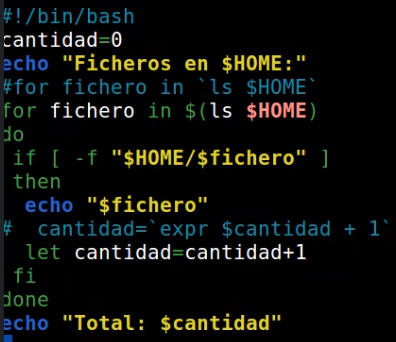
#echo "$aleatorio"

bash ej21.sh $aleatorio

done

14/10/2022 - Ejecuciones de comandos

y recursos adicionales



***nano ej25.sh***

#!/bin/bash

cantidad=0

echo "Ficheros en $HOME:"

#for fichero in `ls $HOME`

#for fichero in $(ls $HOME)

do

if [ -f "$HOME/$fichero" ]

then

echo "$fichero"

# cantidad=`expr $cantidad + 1`

let cantidad=cantidad+1

fi

done

echo "Total: $cantidad"

**let** => **expr**

* **let variable** = operaciones **numéricas**

**$(comando)**=> sirve para ejecutar comandos =>igual que `comillas invertidas`

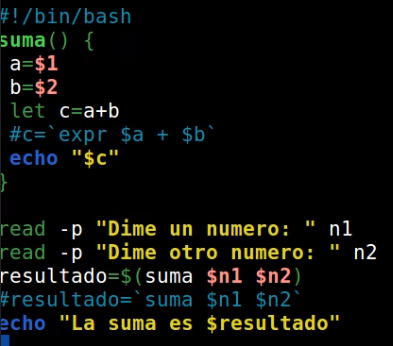
—----------------------------------------------------------------------------------------------------------------------------------

**Funciones (1º se define, para poder ejecutar después):**

suma() {

a=$1

b=$2

let c=a+b

echo “$c”

}

***ej26.sh***

#!/bin/bash

suma() {

a=$1

b=$2

let c=a+b

#c=`expr $a + $b`

echo "$c"

}

read -p "Dime un numero: " n1

read -p "Dime otro numero: " n2

resultado=$(suma $n1 $n2) **=>LLAMAR A LA FUNCIÓN**

#resultado=`suma $n1 $n2` **=>LLAMAR A LA FUNCIÓN**

echo "La suma es $resultado"

—----------------------------------------------------------------------------------------------------------------------------------

**Arrays**

array1=(“elemento1” “elemento2” “elemento3”...)

${#array1[@]}

# cuenta todos los elementos

@ todos los elementos

***ej27.sh***

#!/bin/bash

suma=0

materias=("Sistemas" "Seguridad" "Redes" "Bases de Datos")

longitud=${#materias[@]}

let limite=longitud-1

for i in $(seq 0 $limite)

do

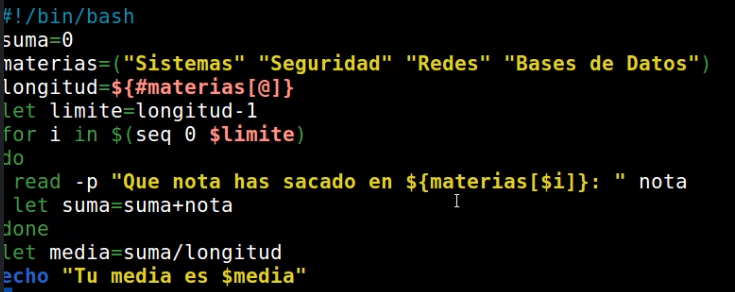
read -p "Que nota has sacado en ${materias[$i]}: " nota

let suma=suma+nota

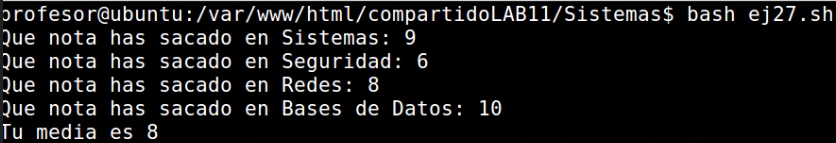
done

let media=suma/longitud

echo "Tu media es $media"



*¡Cuidado! porque los arrays empiezan en 0 por eso le restamos 1 y empezamos la seq en 0*

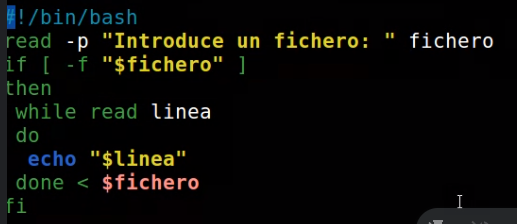


—----------------------------------------------------------------------------------------------------------------------------------

**while comando**

**do**

**done < variable/documento o > variable/documento**



***ej28.sh***

Tener todas las líneas del fichero:

#!/bin/bash

read -p "Introduce un fichero: " fichero

if [ -f "$fichero" ]

then

while read linea

do

echo "$linea"

done < $fichero

fi

***ej28.1.sh leer un documento poco a poco***

#!/bin/bash

read -p "Introduce un fichero: " fichero

if [ -f "$fichero" ]

then

while read linea

do

#echo "$linea"

longitud=$(expr length "$linea") **=> me dice el nº de carácteres**

for posicion in $(seq 1 $longitud)

do

caracter=$(expr substr "$linea" $posicion 1) **=>coger uno a uno todos los carácteres de la linea, los muestro con intervalo de 0.5**

echo -n "$caracter"

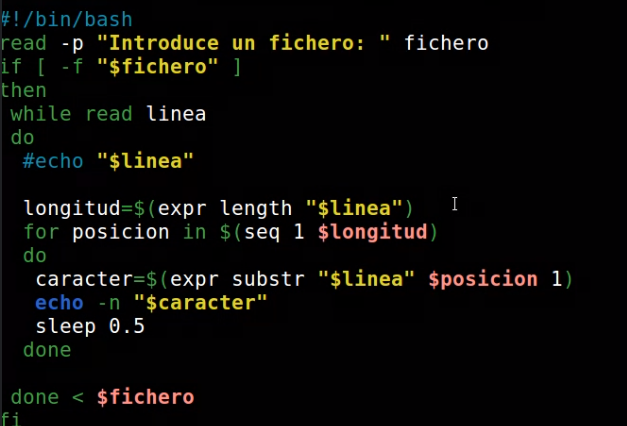
sleep 0.5

done

echo “” **=> salto de línea**

done < $fichero

fi



—----------------------------------------------------------------------------------------------------------------------------------

**Bucle infinito**

#!/bin/bash

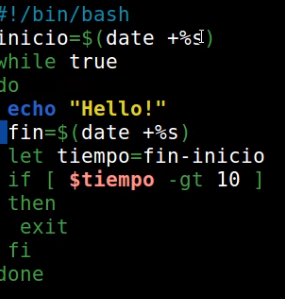
**while** true **#algo que sea verdadero**

do

echo "Hello!"

done

—----------------------------------------------------------------------------------------------------------------------------------**exit->** sale del script (puede ir a compañado de un numero indicando si ha tenido éxito(**exit 0**) o no(**exit** **1**))

**break**-> salta una ejecución

***ej29.sh***

#!/bin/bash

inicio=$(date +%s)

while true

do

echo "Hello!"

fin=$(date +%s)

let tiempo=fin-inicio

if [ $tiempo -gt 10 ]

then

exit

fi

done

**date +%s** => cuántos segundos han pasado desde que se creó linux: 1/1/1970

***ej29.1.sh***

#!/bin/bash

inicio=$(date +%s)

while true

do

while true

do

echo "Hello!"

fin=$(date +%s)

let tiempo=fin-inicio

if [ $tiempo -gt 2 ]

then

break

fi

done

echo "Goodbye!"

fin=$(date +%s)

let tiempo=fin-inicio

if [ $tiempo -gt 5 ]

then

exit

fi

done

**$?** código de salida de un comando (en el ejer 0 o 1)

***ej30.sh***

#!/bin/bash

read -p "Dime un nombre para crear una carpeta: " carpeta

mkdir $HOME/$carpeta 2> /dev/null

if [ $? -eq 0 ]

then

echo "Carpeta creada correctamente"

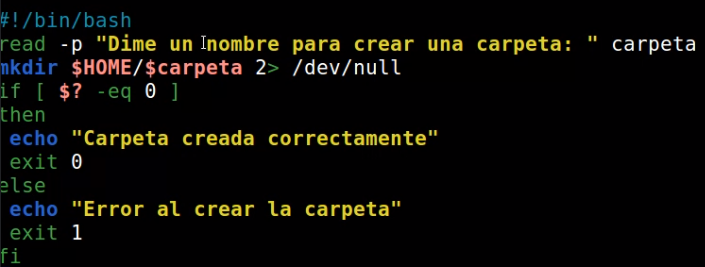
exit 0

else

echo "Error al crear la carpeta"

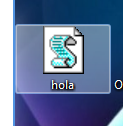
exit 1

fi



Windows

Los archivos se guarda como extensión **.vbs y se ejecuta al abrir con Microsoft Windows Based Script Host**



Siempre en la primera línea hay que poner **option explicit**

**Objeto->wscript**

**wscript.echo->** muestra un texto

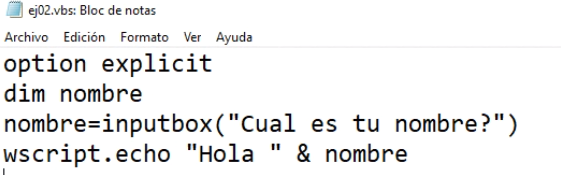
**dim ->**declarar variables con coma. Ej: dim nombre, apellido

**inputbox->**recoge un dato y lo guarda en una variable

**&->** concatenar

**‘comentario->** comentarios

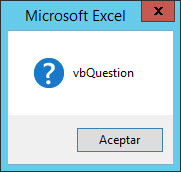
**“text a salir”**





**msgbox->**Muestra un mensaje y guarda que botón ha sido seleccionado. También deja poner distintas configuraciones de iconos **(vbOkOnly)** y botones.



No respeta mayúsculas, y minúsculas, pero sí algunos espacios.



ICONOS:

**vbInformation**->información

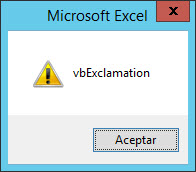
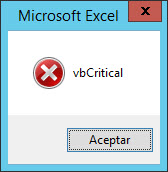
**vbQuestion**->pregunta

**vbExclamation**->warning

**vbCritical**->error crítico

BOTONES:

**vbOkOnly -**>solo aceptar

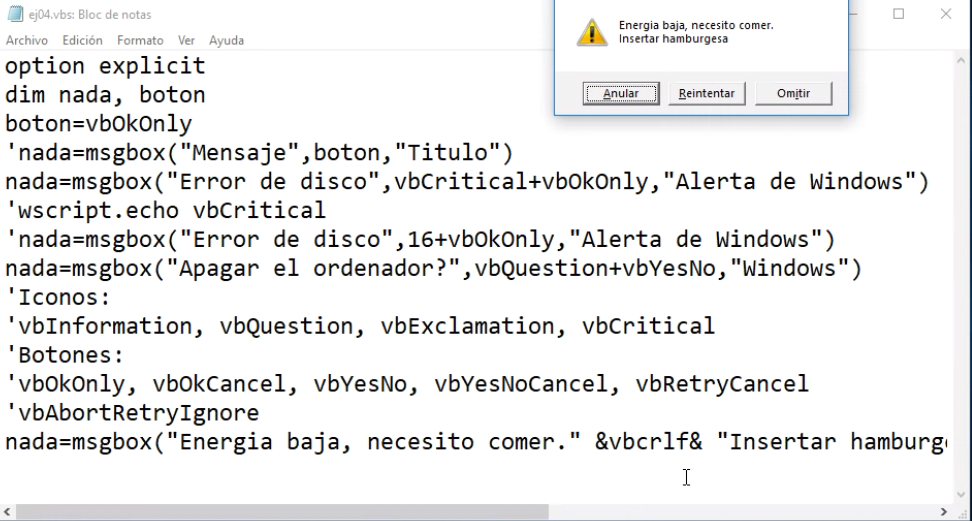
**vbOkCancel** -> aceptar y cancelar

**vbYesNo** -> si y no

**vbYesNoCancel** -> si, no, cancelar

**vbRetryCancel**->reintentar o cancelar

**vbAbortRetryIgnore**-> abortar , reintentar, ignorar

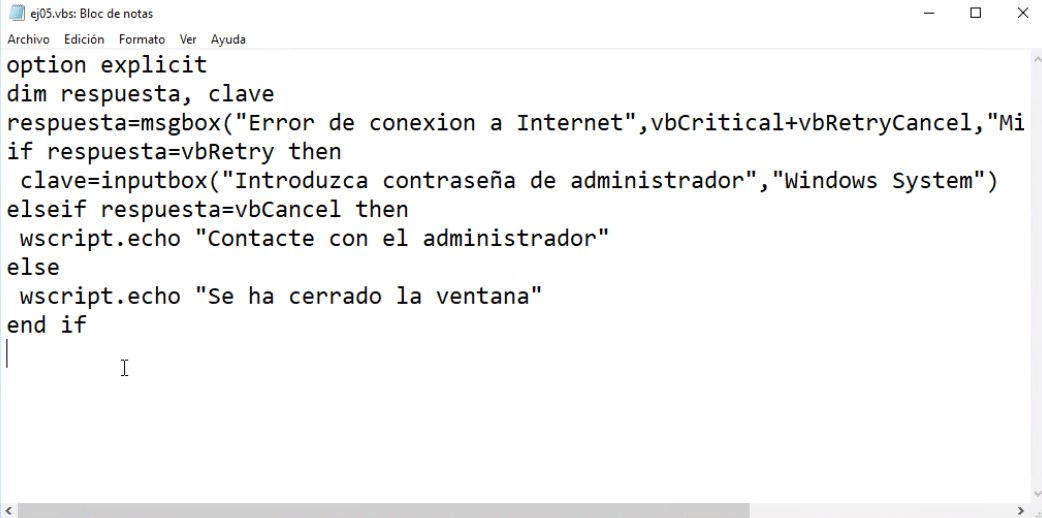


**&vbcrlf& ->** salto de línea en una pregunta

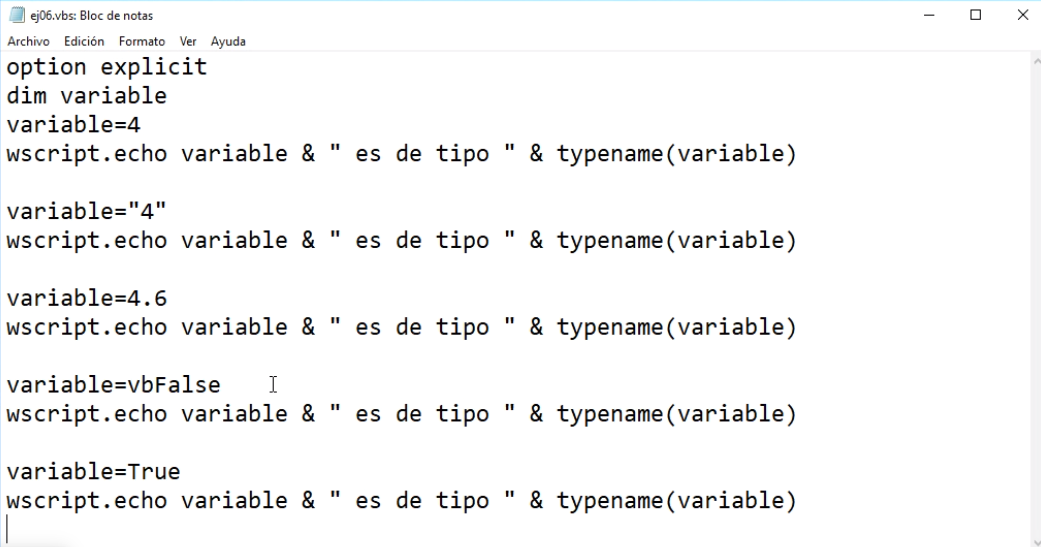


*Ejemplo: variable= msgbox(“Energía baja, necesito.Comer”* ***&vbclrf&*** *“Insertar hamburguesa”,vbExclamation+vbAbortRetryIgnore, “Windows 10”);*

***if,else,elseif***

**

**typename(variable)**-> nos dice que tipo de variable es.



* **“4”-> tipo string**
* **4-> tipo integer**
* **4.6-> tipo Double**
* **vbFalse->tipo Integer (valor 0 predefinido)**
* **True->Boolean**

**cint(variable)**->convierte la variable a Integer

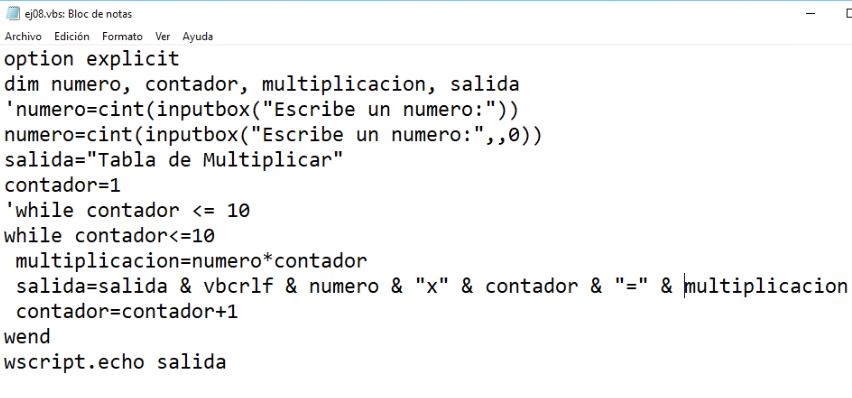
**cstri(variable)**->convierte la variable a String

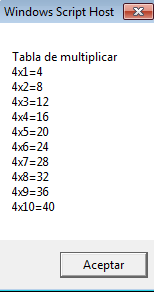
**cdbl(variable)**->convierte la variable a Double

**cbool(variable)**->convierte la variable a Boolean

**BUCLES:**

* **while (condición) – wend**

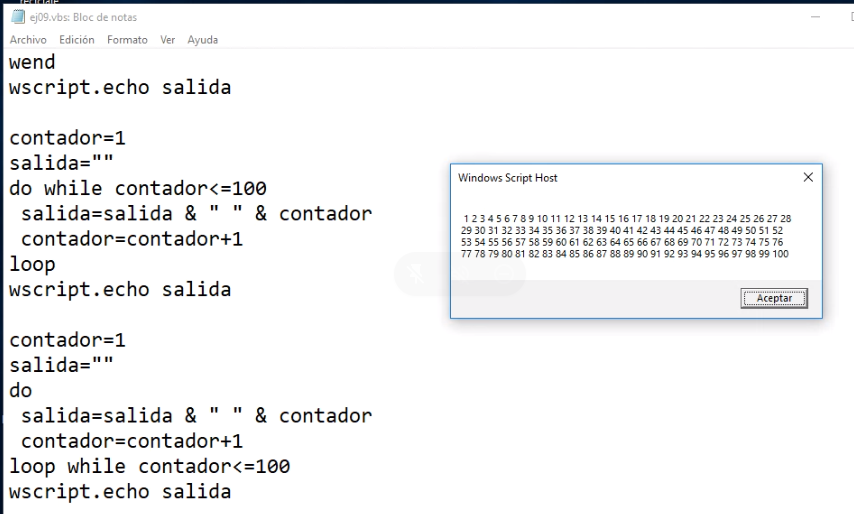
****

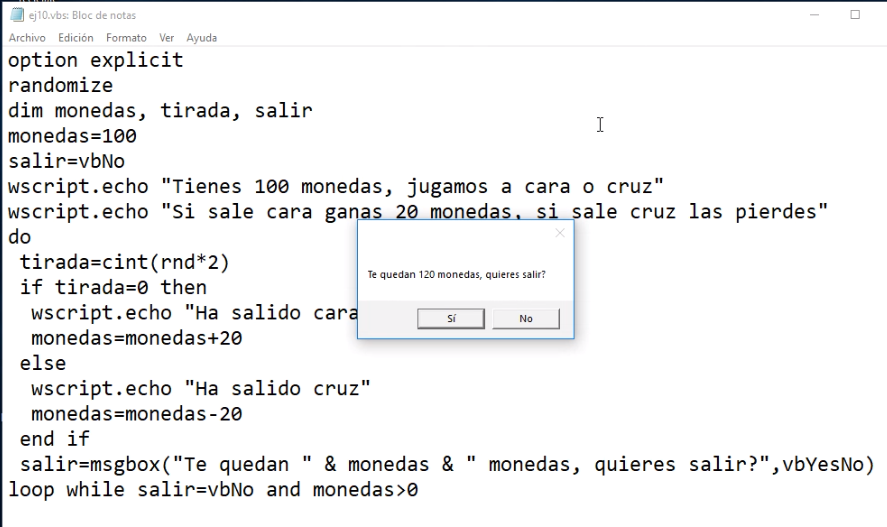
****

* **do while (condición) - loop ->EL QUE MÁS SE VA A USAR**

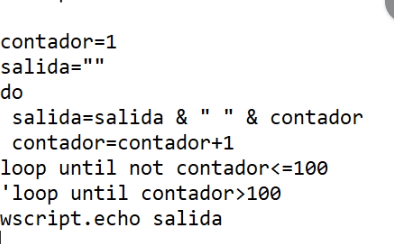
**captura abajo**

* **do - loop while (condición)->** diferencia con los dos anteriores es que se ejecuta mínimo una vez porque la condición está al final.

****

****

* **do - loop until [not] (condición)->** puedes poner el not para negar la condición.

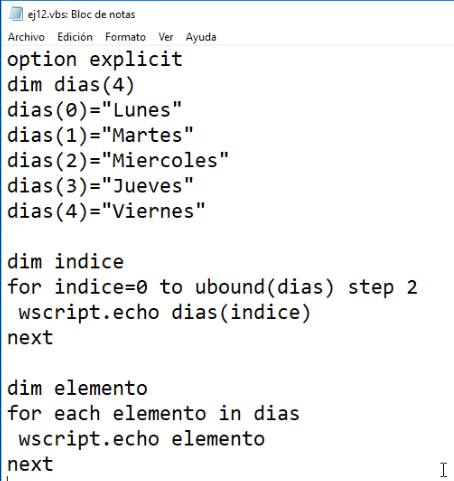
****

**ARRAYS / bucles for**

* **for - variable= (donde empieza)- to 999 step 5 (secuencia)**
* **for each - next**

dim dias (4) -> como 5 elementos

dias (0)=”Lunes”...

****

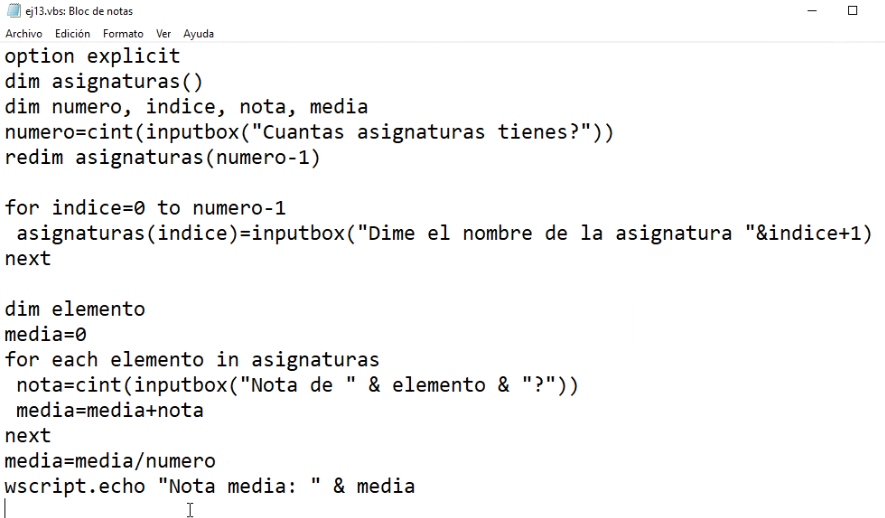
**ubound->** como el length de un array (mayor índice en este caso)-> en este caso 5, porque hay 5 elementos

Salida=

1er bucle= Lunes, Miércoles y Viernes

2º bucle= Salen todos los días

**Cuando no sabemos cuántos elementos queremos que tenga el array:**



En este caso dim asignaturas() es un array que aún no sabemos sus índices

**redim:** darle el valor de cuantos índices

**NUMEROS ALEATORIOS**

**rnd()**

-x defecto del 0 al 1, con decimales. EJ: 0.7055475

-Si lo vuelves a ejecutar, vuelve a salir el mismo valor, que arriba. Para reiniciar la semilla que emplea el ordenador porque no sabe generar números aleatorios, se pone una vez **randomize.**

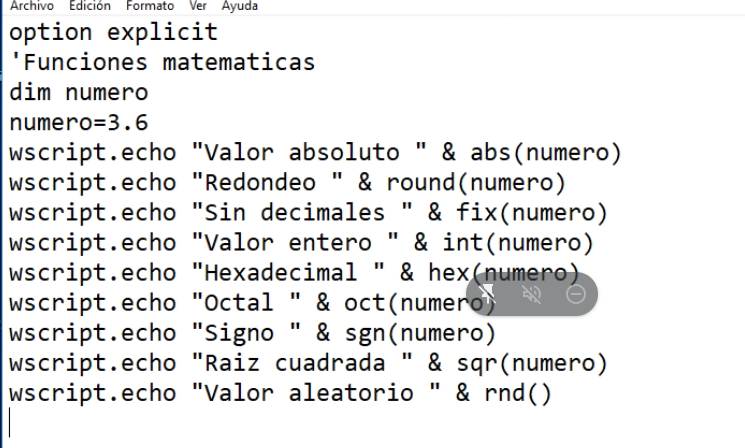
* si quieres números del 1 al 9: multiplicas por 10 y le quitas los decimales con **fix**

**fix(rnd()\*6) -> 0 - 5**

**fix((rnd()\*6)+1)**

**FUNCIONES Y OBJETOS**

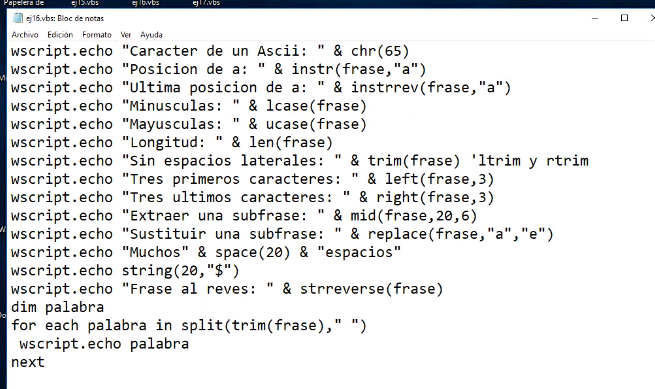
* **Matemáticas:**
  + abs(num)→valor absoluto
  + round(num)→redondeo
  + fix(num)→ sin decimales
  + int(num)→valor entero
  + hex(num)→hexadecimal
  + oct(num)→octal
  + sgn(num)→ signo, si es positivo o negativo
  + sqr(num)→raíz cuadrada
  + rnd(num)→valor aleatorio

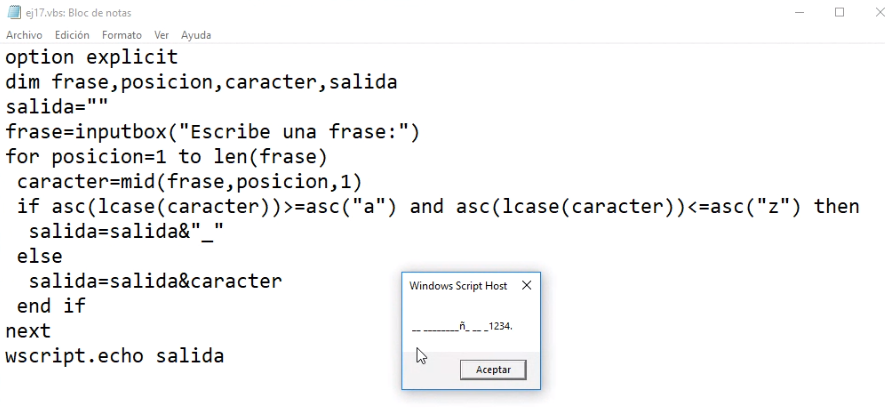
****

* **Texto**
  + asc(“a”)→La posición que tiene la a en ASCII
  + chr(65)→introduces número y te dice la posición del carácter en ASCII.
  + instr(frase,”a”)n→busca la letra de izquierda a derecha (posición num.)
  + instrrev(frase,”a”)→busca la letra de derecha a izquierda (posición num.)

es la última “a”

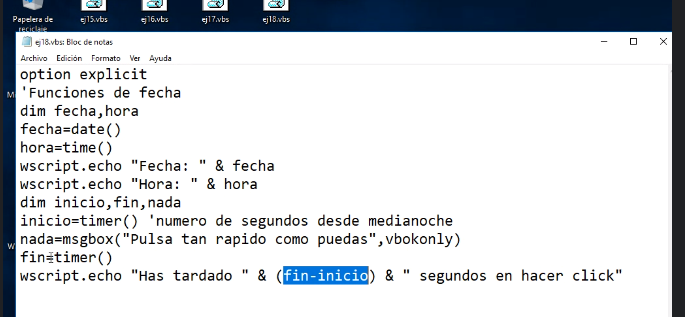
* + lcase(frase)→minúsculas
  + ucase(frase)→mayúsculas
  + len(frase)→longitud numérica
  + trim(frase)→elimina espacios laterales de una cadena
    - ltrim→x la izquierda
    - rtrim→x la derecha
  + left(frase,3)→3 primeros carácteres desde la izquierda
  + right(frase,3)→3 primeros carácteres desde la derecha
  + mid(frase,20,6)→ posición y cuantos caracteres quiero.
  + replace(frase,”a”,”e”)-> para sustituir a por e
  + space(20)-> te pone espacios
  + sting(20,”$”) te pone 20 caracteres de $
  + strreverse(frase) -> te pone la frases al revés
  + split(frase,” ”) mete en un array la frase cortando por espacios

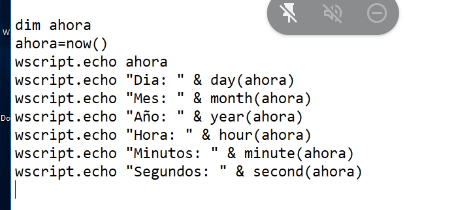
****

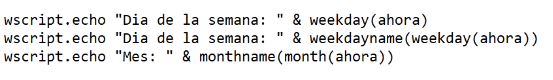


**Para leer una frase caracter a caracter; y comparar si está entre la a-z y si no está mostramos qué carácter es.**

* **Fechas**
  + date()→devuelve la fecha de hoy
  + datediff()-->restar fechas
  + time()→devuelve la hora
  + timer()→el número de segundos desde la medianoche
  + now()→ fecha y hora juntos
  + day()
  + moth()
  + year()
  + hour()
  + minute()
  + second()
  + weekday()→número del día de la
  + semana
  + weekdayname()→día de la semana (“Lunes”)
  + monthname(8)→ devuelve el nombre del mes
    - podemos juntarlos: monthname(month(ahora))→el nombre del mes, en este caso, 11.



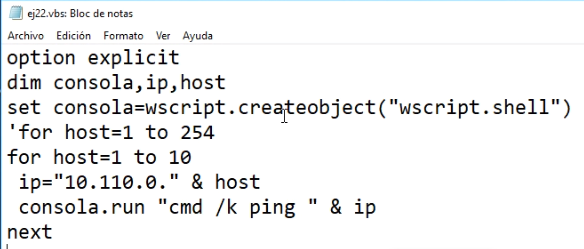




* **ARRAYS**
  + join→cambia de array a cadena ( lo contrario a split)
* **OBJETOS**
  + **para declarar un objeto tengo que poner *set* al principio**
  + wscript.scriptname→nombre del script
  + wscript.scriptfullname→lo mismo que wscript.scriptname pero incluye la ruta
  + wscript.sleep *1000*→esperar un número de segundos
  + wscript.quit→salir
  + wscript.createobject→se utiliza para crear un objeto
    - wscript.shell→terminal de comandos
      * typename(obj)
      * ***objeto*.run**→ ejecuta el objeto
      * ***objeto.*sendkeys** “^p” →(Ctrl+p)→escribir tal cual en la terminal,en este caso imprimir
        + %→lo mismo que Alt
      * ***objeto.*sendkeys** “ping 8.8.8.8 {ENTER}” →escribe comandos,en este caso ejecuta un ping



—--------------------------------------------------------------------------------------------------------------------



**FileSystemObject -> objeto para el manejo del sistema de ficheros**

**DOCUMENTO EN CLASSROOM**

* OBJETO SISTEMA: Scripting.FileSystemObject
  + FUNCIÓN/MÉTODO
    - objeto.getfolder(“”)->obtener la carpeta
  + DATO/OBJETOS
    - objeto.path-> ruta
    - objeto.files-> lista de objetos-> lista de los ficheros que contiene
    - objeto.name-> nombre del fichero
    - objeto.size-> tamaño

