Bash puede ejecutar casi cualquier operación como si fuera un lenguaje de programación

Se pueden hacer condiciones variables ciclos métodos y todo esto nos sirve cuando mezclamos comandos, ejemplo un reporte de todos los archivos que tiene mas de 50gb y los ponemos en un archivo y lo mandamos por correo.

La diferencia entre bash y los copiladores es que esta el sistema operativo, bash no necesita librerías si no que se auxilia de los comandos internos del sistema operativo,

Cuando nosotros estemos trabajando scripts con bash le informas a el sistema operativos que utilizamos bash

#!/bin/bash 🡨-- Primeta línea simpre, que indica que BASH estamos utilizando

#Esto es un ejemplo 🡨----- esto indica un comentario

Echo “Hola, esto es un ejemplo”

Usar bin en la terminal

Vim ejercicio1.sh

Cuardar es con Esc

:wq

O

:w

Ejecutar el programa

li ejercicio1.sh

ver permiso

chmod +x ejercicio1.sh

Bah con parámetros

#!/bin/bash 🡨-- Primeta línea simpre, que indica que BASH estamos utilizando

#Esto es un ejemplo 🡨----- esto indica un comentario

echo $0 #primer parámetro

echo $1 # primer argurmento

ARCHIVO = “archivo\_salida”

echo “Hola, $(1) esto es un ejemplo” > $(ARCHIVO)

#del siguiente ejemplo

# ./ejercicio1 2022

#El parámetro 0 es ejercicio1

#El parámetro 1 es 2022

#solo de 0 a 9 no exceder recomendación, 9 parametro después del ejecutable.

#comando **ps** ejecuta lo que esta corriendo actualmente

#ps -ef | grep.sh le pasa parámetro para

Variables de entorno

$$ = Process ID

$USER = el usuario que esta ejecutando el proceso

$HOSTNAME = el host donde se ejecuta el proceso

$SECONDS = Tiempo que lleva ejecutando el proceso

Poder de bash

$$ = Process ID

$USER = el usuario que esta ejecutando el proceso

$HOSTNAME = el host donde se ejecuta el proceso

$SECONDS = Tiempo que lleva ejecutando el proceso

>significar crea un archivo

>> abrir el archivo y agregar el final

mkdir curso\_bash

cp ejercicio1.sh curso\_bash/

cd curso\_bash

exit 0

Operación armeticas

Let a+7+10

Echo $a

Exit 0

E

Operaciones arimeticas 2

Let a+$1+$2

Let b= $1-$2

Echo $a

Exit 0

Operaciones aritméticas 3

Expr c=$1+$2

Expr d=$1+$2

Echo $a

Echo $b

Echo $c

Echo $d

Exit 0

Let permite asignar a una variable el resultado

Expr que permite realizar una expresión y mostrarla en vivo

Echo $(DATE)

Dia 2

Echo $@ # muestra los parámetros que tengo en forma de una cadena

Debugear

echo $LINERO #nos muestra el error

echo $RANGE #

#!/bin/bash

Let multi= $1\*$2

echo $(multi)

exit 0

para ejecutar se le da permiso

chmod +x calculadora.sh

./calculadora.sh 8 7

Para casi todas las operaciones que se realicen no se debería dejar espacios en blancos

#imprimir fechas

INVESTIGAR QUE OPERADOTES SE PUEDE UTILIZAR EN LAS CODICIONES IF EN LINUX

VARIABLES DE ETORNO MAS COMUNES EN LINUX

USER: El usuario que inició sesión actualmente

PWD: El directorio actual de trabajo.

PATH: Una lista de directorios que el sistema comprobará cuando busque comandos. Cuando un usuario escriba un comando, el sistema comprobará los directorios en este orden para el ejecutable.

HOME: El directorio principal del usuario actual.

HOSTNAME: El nombre de host de la computadora en este momento.

Bash tiene un gran soporte de comparadores de todo tipo que nos permiten hacer comparaciones en los bucles y crear condiciones de todo tipo:

**Comparación de enteros (números)**

* **-eq**  
  es igual a

if [ "$a" -eq "$b" ]

* **-ne**  
  no es igual a / es distinto a

if [ "$a" -ne "$b" ]

* **-gt**  
  es mayor que

if [ "$a" -gt "$b" ]

* **-ge**  
  es mayor que o igual a

if [ "$a" -ge "$b" ]

* **-lt**  
  es menor que

if [ "$a" -lt "$b" ]

* **-le**  
  es menor que o igual a

if [ "$a" -le "$b" ]

* **<**  
  es menor que (dentro de doble paréntesis)

(("$a" < "$b"))

* **<=**  
  es menor que o igual a (dentro de doble paréntesis)

(("$a" <= "$b"))

* **>**  
  es mayor que (dentro de doble paréntesis)

(("$a" > "$b"))

* **>=**  
  es mayor que o igual a (dentro de doble paréntesis)

(("$a" >= "$b"))

**Comparación de cadenas**

* **=**  
  es igual a

if [ "$a" = "$b" ]

* **==**  
  es igual a

if [ "$a" == "$b" ]

Nota: Aunque es un sinónimo de = el operador *==*se comporta diferente cuando se usa dentro de corchetes dobles que simples, por ejemplo:

[[ $a == z\* ]] # Verdadero si $a empieza con una "z" (expresión regular coincide).

[[ $a == "z\*" ]] # Verdadero si $a es igual a z\* (coincide literalmente).

[ $a == z\* ] # Ocurre división de palabras.

[ "$a" == "z\*" ] # Verdadero si $a es igual a z\* (coincide literalmente).

* **!=**  
  no es igual a / Distinto

if [ "$a" != "$b" ]

NOTA: este operador usa coincidencia de patrón dentro de doble corchete.

* **<**  
  es menor que (en orden alfabético ASCII)
* if [[ "$a" < "$b" ]]

if [ "$a" \< "$b" ]

Nota: el operador "<" necesita ser escapado dentro de corchetes.

* **>**  
  es mayor que (en orden alfabético ASCII)
* if [[ "$a" > "$b" ]]

if [ "$a" \> "$b" ]

Nota: el operador ">" necesita ser escapado dentro de corchetes.

* **-z**  
  La cadena está vacía (nulll), tiene longitud cero.
* cadena='' # Variable de longitud cero (null)
* if [ -z "$String" ]
* then
* echo "\$cadena está vacía."
* else
* echo "\$cadena no está vacía."

fi

* **-n**  
  cadena no está vacía (contiene algo) nota: El operador *-n* exige que la cadena esté entre comillas entre paréntesis. Aunque el uso son comillas puede funcionar es altamente recomendable usar comillas.

**Comparaciones lógicas**

* **-a**  
  Y lógico (and)

exp1 -a exp2

devuelve verdadero si ambas exp1 y exp2 son verdaderas.

* **-o**  
  O lógico (or)

exp1 -o exp2

devuelve verdadero si alguna de las expresiones exp1 y exp2 son verdaderas.

Éstos últimos operadores son similares a los operadores de Bash **&&** (and) y **||** (or) cuando se usan con doble corchete:

[[ condition1 && condition2 ]]